PROYECTO FINAL DE CARRERA

UNIVERSITAT ROVIRA I VIRGILI

Investigación y Análisis de Ontologías para el manejo de información en el World Wide Web

Javier Sánchez Alonso, jsa.ge@estudiants.urv.es
Ingeniería Técnica en Informática de Gestión

Directora: **Arantza Aldea**

Escola Tècnica Superior d'Enginyeria (ETSE)
Universitat Rovira i Virgili (URV)
http://www.etse.urv.es

Curs 2002-03

Índice principal

1. INTRODUCCIÓN

Este proyecto final de carrera está comprendido dentro del proyecto H-TechSight. El proyecto H-TechSight que se explicará en detalle a lo largo del documento, consiste en la creación de un buscador web basado en Ontologías, para facilitar a las empresas tecnológicas el manejo de la información que reside en un entorno tan amplio como es Internet. Este buscador esta constituido por un sistema multi-agente el cual ayudándose de las especificaciones de dominios concretos definidos en las Ontologías, extraen las páginas webs relacionadas pertenecientes a este dominio, para poder en un futuro analizar esas páginas y extraer conocimiento que puede ser beneficioso para las empresas.

El objetivo de mi proyecto final de carrera es el de ayudar en el proyecto H-TechSight en los siguientes conceptos:

- Estudio de las Ontologías (editores, lenguajes y metodologías).
- Comparación de dos editores de Ontologías (OntoEdit,WebODE), para poder evaluar que editor es el que se adecua mejor a nuestro proyecto.
- Creación de una Ontología en el dominio de los Biosensores.
- Depuración de la Ontologías, para obtener un búsqueda más eficiente.
- Razonamiento de la InformationOntology, con todas las páginas webs encontradas por los agentes.
- Utilización de repositorios (Sésame) para poder almacenar las InformationOntology resultadas de las búsquedas, y en un futuro poder aplicar razonamientos sobre ellas como el análisis de la información que contienen estas Ontologías pudiendo ver como cambia a través del tiempo.

La primera parte del proyecto, está comprendida en los puntos 1 y 2, esta parte es una parte muy teórica ya que esta enfocada a entender que es una Ontología, como se puede crear, y que necesitamos para crearla. Para explicar todo esto, he empezado desde cero

empezando por que es una Ontología y acabando por cual es el uso que le damos en el proyecto H-TechSight . Una vez vistos estos dos puntos podremos entender porque la definición de una Ontología es tan importante cuando tratamos con grandes volúmenes de información.

La segunda parte del proyecto es la parte práctica, en esta parte se aplica toda la teoría estudiada de las Ontologías. En la parte práctica de mi proyecto, he ayudado al proyecto de H-Techsight, creando la Ontología de biosensores, depurándola (por dos motivos, el primero es porque la estructura de la Ontología influye directamente a la eficiencia de las búsqueda realizada por los agentes, y el segundo es por la interoperabilidad entre los programas que he utilizado, los cuales generán la Ontologías en un mismo lenguaje aunque se han de hacer pequeñas modificaciones en éste para poder utilizar la misma Ontología en todos los programas), también he tenido que razonar con las InformationOntology, para encontrar nuevas relaciones entre conceptos de la propia Ontología ayudando así a poder extenderla y hacer las búsquedas aún mas competitivas. Y por último analizar una herramienta para utilizarla como repositorio de Ontologías (Sésame), para que en un futuro se pueda comparar la información que contiene cada Ontología y poder analizar por ejemplo como cambia esa información dinámicamente.

Esta última parte práctica, se explica con detalle en el punto 3 de este documento, donde no solo expongo lo que he hecho sinó que también hago una descripción de las herramientas que he utilizado, y que uso he hecho con ellas.

2. ONTOLOGÍAS

2.1. Introducción: La necesidad de Ontologías en la sociedad de la información

En las últimas décadas el volumen de información de que disponemos se ha multiplicado exponencialmente, gracias sobretodo al World Wide Web (WWW), un

entorno donde el gran volumen de información disponible hace que el manejo de ésta sea también altamente laborioso.

Cuando se dispone de un gran volumen de información , ésta tiene que estar clasificada y categorizada, para poder extraer información eficientemente. En el presente la mayoría de información que se presenta en el WWW no dispone de estos requisitos por lo que hace mucho mas difícil la extracción de información procesable de cualquier página web.

El futuro de la información que se distribuye en éste entorno, es una información clasificada y categorizada, gracias al uso de lenguajes de programación web como XML o RDF que no solo se dedican a la estructuración del contenido de una página web, sino que también dotan de significado a la información que en ella reside, de esta forma nosotros podemos extraer con diferentes herramientas, información procesable por diferentes lenguajes, y de esta manera poder relacionarla con otros tipos de información e incluso razonar con ella sabiendo si dicha información relevante o no.

Dicho de otra manera dotar al la web de semántica, para facilitar el uso de herramientas por ejemplo que se dedican a la extracción de información. Poniendo un ejemplo de una de estas herramientas, nos basamos en el mejor buscador de Internet como es GOOGLE, si nosotros queremos encontrar información acerca de biosensores, escribiremos la palabra biosensor en la máquina de búsqueda, pero esta búsqueda nos encontrará todas las páginas que contengan la palabra biosensor, y no todas ellas estarán relacionadas con el campo específico en el que el usuario realiza la búsqueda (ej. Compañías de biosensores), puesto que muchas páginas pueden nombrar dicha palabra sin que su tema principal trate sobre este campo. Aunque luego tenga herramientas para filtrar páginas que no estén muy relacionadas, nunca encontrará páginas en donde aunque no aparezca explícitamente esta palabra su contenido este implícitamente relacionado con los biosensores como por ejemplo compañías de biosensores o procesos de producción.

En conclusión para poder mantener un entorno de información dotado de significado, necesitamos una herramienta que nos proporcione poder clasificar la información, que

sirva para el manejo de información eficiente entre entidades, compañías y también entre sistemas automatizados que necesitan un lenguaje y un dominio estándar para comunicarse entre ellos.

2.2. Definición

Antes de empezar con definiciones de ¿que es una Ontología?, y antes de empezar a diseñarlas nos hemos de formular unas preguntas básicas que nos ayudarán a entender el funcionamiento de éstas y su posterior uso:

¿Qué metodologías puedo usar para construir mi Ontología, si quiero empezar una a partir de un dominio específico o si por lo contrario necesito reutilizar otra Ontología y extenderla?.

¿Qué actividades se llevan a cabo en las diferentes metodologías existentes para el diseño de Ontologías?.

¿Existen metodologías o herramientas de soporte que permitan construir una Ontología colaborativamente?.

¿Cual es el ciclo de vida de una Ontología desarrollada con una cierta metodología?,

¿Por que etapas pasará mi Ontología hasta estar creada correctamente?

¿Qué tipo de herramientas existen para facilitar el desarrollo en la creación de Ontologías?

¿Utilizando este tipo de herramientas mi Ontología como quedará almacenada? en ficheros, bases de datos...

¿Estas herramientas existentes están provistas de algún mecanismo de razonamiento?

¿Todas estas herramientas añaden traductores de Ontologías en diferentes tipos y formatos de lenguaje? ¿Estas traducciones, mantienen la consistencia de la información o hay algún tipo de pérdida?

¿Qué lenguajes puedo usar según para diseñar una Ontología? ¿Qué tipo de expresividad me da cada lenguaje?

¿El lenguaje que utilice será compatible con otros lenguajes usados para representar el conocimiento y la información en Internet?

Todas las respuestas a estas preguntas las veremos a continuación, estudiando que tipos de Ontologías hay, que metodologías existen para el diseño de estas Ontologías, que tipos de herramientas dan soporte a la creación de Ontologías, y que lenguajes permiten exportar estas herramientas.

¿Qué es una Ontología?

Podemos definir una Ontología como el resultado de seleccionar un dominio y aplicar sobre él mismo un método para obtener una representación formal de los conceptos que contiene y las relaciones que existen entre los mismos.

Por tanto una Ontología clasifica toda la información de un dominio específico en axiomas lógicos, esta información se clasifica como lo haríamos si utilizásemos una metodología orientada a objetos, por tanto se definen una serie de conceptos que se representan en clases con sus correspondientes atributos. Esta jerarquización de la información, nos permite aplicar conceptos relacionados con la programación orientada a objetos como por ejemplo la herencia, donde un concepto puede tener una relación padre-hijo con otro similar. La propia metodología orientada a objetos nos permite hacer razonamiento con estos conceptos puesto que un concepto herede de otro significará implícitamente q todos sus atributos también se heredarán y esto es un factor fundamental en la clasificación de la información.

Una Ontología juega un papel crucial tanto en la extracción de información de la WWW como en la comunicación entre agentes que componen un Sistema multi-agentes, puesto que se necesita una conceptualización de un dominio para que sirva de estándar de comunicación tanto para agentes humanos como de software.
La definición que hemos dado es una definición universal sobre el significado de la palabra Ontología, hoy en día hay un acuerdo mutuo entre todos los Organismos que se dedican al estudio del conocimiento, en el significado de Ontología, por tanto esa definición es totalmente correcta, aunque si es verdad que no fue la primera.

Una de las primeras definiciones la dio Neches [1], el cual definió una Ontología como: **"Una Ontología define los términos básicos y las relaciones entre ellos de un tema en concreto como también la reglas para combinarlos y extender otros términos y relaciones del vocabulario"** Esta definición nos dice que tenemos que hacer para construir una Ontología dándonos unas pautas: En la definición habla de términos y relaciones entre términos, identifica reglas para combinarlos y extender mas definiciones de términos y relaciones a los ya existentes.

Unos años más tarde Gruber [6] definió una Ontología como: **"Una especificación explícita de una conceptualización"** esta definición pasó a ser la más aceptada en la literatura y en toda la comunidad Ontológica. A partir de esta definición aparecieron muchas más las cuales eran modificaciones de esta pero la base seguía siendo la misma, como ejemplos tenemos a Borst [3] que modificó ligeramente la definición de Gruber: **"Las Ontologías se definen como una especificación formal de una conceptualización compartida",** posteriormente Studer [4] combino las definiciones de Gruber y Borst para obtener: **"La conceptualización hace referencia a un modelo abstracto de algun fenómeno en el mundo el cual se destacan los conceptos más importantes de éste"**

A partir de estas tres definiciones aclararemos el significado de tres palabras claves que utilizan para describir el significado de Ontología.

Con **explícito** se quiere dar a entender que el tipo de conceptos usados y sus restricciones se tienen que definir explícitamente. Cuando Utiliza la palabra **formal** se

refiere al hecho de que la Ontología debe estar expresada formalmente para que puede ser interpretada por diferentes mecanismos automáticos como por ejemplo las herramientas de soporte. Y la palabra **compartido**, en el hecho de que la conceptualización del dominio ha de estar compartida, se refiere a que una Ontología no debe privatizarse a algo individual sino debe estar accesible para que pueda servir de soporte, como información formalizada, y pueda ser utilizada o para especializarla en información concreta de ese dominio o para generalizarla en información más abstracta del mismo.

Posteriormente en 1995, Guarino [5] colecciono y analizo 7 definiciones mas de Ontologías y como conclusión definió una Ontologías como **"Una teoría lógica la cual aporta una explícita y parcial parte de una conceptualización sobre un dominio específico"** donde la conceptualización la entiende como una idea del mundo que una persona o grupo de personas pueden tener. Aunque a priori la idea de conceptualización aportada por Guarino es la misma que aportaba Studer, éste da un paso más estableciendo como requisito a la construcción de una Ontología crear una teoría lógica de un dominio. Aunque estrictamente hablando esta definición solo debería ser aplicable a aquellas Ontologías que basan su diseño en lógica de primer orden.

Existen otro tipo de definiciones sobre Ontologías basadas en el proceso que se lleva a cabo para la construcción de la misma.
Por ejemplo Bernaras [6] en el proyecto Kactus definió una Ontología como: **"Una Ontología provee el significado para describir explícitamente una conceptualización del conocimiento extraído de una base de conocimiento (KB)"**, la Ontología se construye siguiendo una estrategia Bottom-up ya que la información sufre un proceso de abstracción de una base de conocimiento.

Otra estrategia para la construcción de Ontologías como ya hemos comentado anteriormente es el reuso de otras Ontologías, como es el caso de Ontologías como SENSUS, con más de 70.000 conceptos representados, de la cual se pueden especializar Ontologías de dominios específicos a si como bases de conocimiento, si el conocimiento no se puede extraer de estas Ontologías una vez extraída la información de otras fuentes y formalizada ésta se puede extender la Ontología SENSUS con el

objetivo de poder abarcar más campo de conocimiento. A partir de este tipo de estrategia se basa la siguiente definición de Ontología: **"Una Ontología es una estructura jerárquica de un conjunto de términos que describen un dominio que puede ser usada como esqueleto para la creación de mas bases de conocimiento"**. Con esta definición se sobreentiende que una misma Ontología puede crear varios KBs con el cual se beneficiarían por utilizar el mismo esqueleto de conocimiento y por lo tanto la interoperabilidad entre diferentes KBs resultaría más fácil.

A veces la noción de Ontología no es tan trivial como aparenta, en este sentido las Taxonomias también se consideran Ontologías, por ejemplo UNSPC, e-cl@ss y RosettaNet, los cuales son estandards del dominio del comercio electrónico, y también el directorio de Yahoo que es una taxonomía que se utiliza para buscar información en la Web, son consideradas Ontologías porque proveen una conceptualización consensuada de un cierto dominio. La comunidad ontológica distingue entre Ontologías que son principalmente taxonomías, de Ontologías que modelan un cierto dominio más profundamente y aplican mas restricciones en la semántica del dominio.
Por tanto la comunidad ontológica distingue entre lo que son ***lightweight ontologies*** (Ontologías ligeras) *y* ***heavyweight ontologies*** (Ontologías pesadas).
Por una parte las *lightweight ontologies* incluyen conceptos, taxonomias de conceptos, relaciones entre los conceptos y propiedades que describen características de los conceptos. Por otro lado las *heavyweight ontologies* añaden a lo visto anteriormente axiomas y restricciones, y por lo tanto razonamiento.

Por este motivo, con la definición de los tipos de Ontologías que existen, éstas son utilizadas por muchos y diversos fines:

- Procesar el lenguaje natural
- Manejo del conocimiento.
- Comercio electrónico.
- Integración de información inteligente.
- Web semántica.

Teniendo en cuenta lo dicho anteriormente, todo lo que hace referencia al diseño de bases de datos y también claro está los lenguajes orientados a objetos, modelan un dominio con conceptos, relaciones, y propiedades, pero ninguna de las dos comunidades impone una restricción semántica tan severa como lo hacen las *heavyweight ontologies.*

De este concepto sacamos la definición que propusieron Uschold y Jasper que decían : **"Una Ontología puede tomar varias formas, pero necesariamente debe incluir un vocabulario de los conceptos usados y algunas especificaciones de su significado, Esto incluye definiciones y una explicación de cómo están inter-relacionados estos conceptos entre sí, los cuales representan la estructura de un dominio, y además han de restringir las posibles interpretaciones de los términos"**.

Por tanto como hemos visto, las funciones que puede desempeñar una Ontología son muchas desde poder dar soporte a la creación de bases de conocimiento (KBs) hasta la utilización de estas para aplicaciones en el mundo del comercio electrónico.

Lo que no hemos visto aún es como ha de ser una Ontología para poder decir que está diseñada correctamente. Como toda aplicación una Ontología debe cumplir una serie de requisitos para poder considerarla una Ontología eficiente, porque como consecuencia de su gran utilización como soporte de información la eficiencia en la creación de esta supondrá a posteriori la eficiencia de muchas aplicaciones.

Requisitos de una Ontología

1. **Claridad:** Una Ontología debe poder comunicar de manera efectiva el significado de sus términos. Las definiciones deben ser objetivas y comentadas en lenguaje natural.

2. Coherencia: Una Ontología debe de expresar un significado que sea consistente con las definiciones de los conceptos, y si a una Ontología se le realiza algún tipo de razonamiento o inferencia éste no debe trastocar el significado de los conceptos que inicialmente se tenía.

3. Extendible: Una Ontología debe anticipar usos puesto que una Ontología se puede reutilizar fácilmente y por tanto debe estar abierta a la inserción de nuevo conocimiento en ella.

4. Sesgo de codificación mínimo: Debe de especificar a nivel de conocimiento, sin depender de una codificación particular a nivel de símbolo, por tanto la Ontología debe de ser independiente al lenguaje en el cual se exporte y la base de conocimiento de be ser igual para cualesquiera de los lenguajes.

Una vez hemos visto que se entiende por Ontología, con un recopilatorio de definiciones, hemos de pasar a estudiar que tipos de métodos o metodología hemos de seguir para crear nuestra Ontología, dependiendo del uso que hagamos con ella.

2.3. Métodos y Metodologías para construir Ontologías

A partir del uso más generalizado de las Ontologías empezaron a aparecer métodos para la construcción de éstas, así pues en 1990 Lenat y Guha [7] publicaron los pasos generales y algunos puntos interesantes sobre el método **Cyc** para el desarrollo de KBs. Algunos años más tarde, en 1995 basándose en la experiencia acopiada desarrollaron la **Enterprise Ontology** [8] y **Tove** (Toronto Virtual Enterprise) [9], ambos destinados al dominio empresarial, se publicaron las primeras pautas de ambos métodos y tiempo mas tarde se refinaron.

La metodología **METHOLONGY** [10] apareció al mismo tiempo, la cual más tarde fue también ampliada. En 1997 se creó un nuevo método para construir Ontologías

basadas en la Ontología SENSUS [11], y años más tarde, apareció la metodología **On-To-Knowledge** como resultado de un proyecto que llevaba el mismo nombre [12].

Sin embargo todos estos métodos y metodologías no permitían una construcción de la Ontología de forma distribuida y colaborativamente, el único método que incluía la propuesta de una construcción colaborativa era el CO4 [13].

Este método incluía un protocolo para aceptar nuevos fragmentos de conocimiento al resto de la arquitectura de conocimiento, la cual ésta había sido aceptada previamente.

De este modo una Ontología podía ser extendida y mejorada por varios usuarios de forma colaborativa, es decir agregando conceptos o relaciones que pudieran completarla, y de forma distribuida, sin depender de un lugar específico para modelar esta Ontología.

Todos estos métodos y metodologías de los que hemos hablado, fueron propuestos para la construcción de Ontologías, pero hay muchos métodos y metodologías que no solo se encargan de esta parte del diseño sino abarcan temas como: Ingeniería inversa de una Ontología, es decir de un texto dado construir la Ontología, la combinación o fusión de Ontologías, el aprendizaje de una Ontología, la evaluación de una Ontología y la evolución de una Ontología.

Hay muchos tipos de métodos y metodologías que se inspiran muchas veces en el tipo de Ontología que van a construir ya pueda ser una Ontología creada con lógica de primer orden, una Ontología creada a partir de un dominio concreto especificando los conceptos más relevantes y a partir de allí especializando el resto de conceptos, o bien sea utilizando enormes Ontologías como SENSUS y especializándolas para crear Ontologías específicas.

A continuación describiremos los métodos más relevantes para crear Ontologías teniendo en cuenta los factores descritos anteriormente.

El método usado para construir el **Cyc KB** [7] consistía en tres fases: La primera fase consistía en la codificación manual de articulos y trozos de información que llevaban conocimiento implícito sobre algún campo en concreto, toda esta información se extraía manualmente. La segunda fase se adquiría mas información que se añadiría a la ya obtenida usando herramientas de soporte para la extracción de información y formalizándola (Herramientas para la creación de Ontologías como Protege2000, WebODE, o OntoEdit), y la tercera fase y última era parecida a la segunda pero la información se adquiría automáticamente sin la ayuda de ningún humano, y esta automáticamente se añadía a la ya obtenida extendiendo la Ontología.

Otro método importante es el descrito por **Uschold y King** [8], los cuales proponen 4 actividades que se han de realizar para la construcción de la Ontología: la primera actividad consiste en evaluar el objetivo de la Ontología, para que queremos crear esta Ontología, la segunda actividad sería construirla, la tercera evaluarla, mirando si la Ontología final cumple los requisitos que habíamos determinado en la primera actividad, y la última documentarla.

Durante la actividad de construcción de la Ontología, los autores proponen capturar el conocimiento, formalizarlo en código, y integrar otras Ontologías si éstas aportasen más conocimiento. Los autores también proponen 3 estrategias para identificar los principales conceptos de una Ontología: la primera llamada **Top-down**, consiste en extraer de un dominio o identificar los conceptos mas abstractos y luego ir especializándolos en conceptos más concretos, la segunda llamada **Bottom-up** consiste en identificar los conceptos mas específicos y luego generalizarlos a conceptos mas abstractos, y por último tenemos la estrategia llamada **Middle-out** que consiste en identificar los conceptos más importantes y luego a partir de ellos especializarlos y generalizarlos a otros conceptos.

Grüninger y Fox [9], propusieron también una metodología inspirada en los sistemas basados en el conocimiento que utilizaban como herramienta de inferencia la lógica de primer orden. Primero lo que se hacía era identificar los escenarios principales, que son las posibles aplicaciones que tiene la Ontología una vez desarrollada, posteriormente se formulaban una serie de preguntas en lenguaje natural para ver el alcance que tenia la

Ontología, estas preguntas se llamaban **Competency Questions.** Estas preguntas y sus respuestas se formulaban para poder extraer de forma más fácil los principales conceptos que se representarían en la Ontología, las propiedades de que estaría compuesto cada concepto, y como no las relaciones y axiomas que tendría la Ontología creada. Todos los componentes de la Ontología al igual que ésta estaban formalizados en lógica de primer orden y partían de la ventaja que les daba la robusteza de la lógica clásica, ya que ésta les servía de guía para poder transformar escenarios informales en modelos computables.

Otro método también a tener en cuenta es el desarrollado en el proyecto **KACTUS** [6], la Ontología se construye a partir de varios KBs, cuantos mas KBs haya más general será la Ontología, es decir si hay un determinado KB de un dominio específico y se quiere construir otro KB de otro dominio similar al primero, lo que se hace es transformar el primer KB en una Ontología, y ésta se adapta para representar todos los conceptos que contiene el segundo KB y de esta forma la Ontología contiene información de los 2 KBs y hace más fácil la creación de mas bases de conocimiento similares a las anteriores porque ya arrastra todo el conocimiento representado formalmente y relacionado de las otras dos. Por tanto esta metodología lo que hace es partir de un dominio específico abstraer la información para poder relacionarla con otro similar, por tanto utiliza un estrategia **Bottom-up.**

Otro método diferente a los ya vistos se basa en la Ontología **SENSUS** [32], utiliza una estrategia **Top-down** porque lo que hace es especializar Ontologías específicas de enormes Ontologías como es **SENSUS.** Los autores lo que proponen es primero de todo identificar un conjunto de términos relevantes de un dominio en particular, entonces esos términos son linkados a este tipo de Ontologías como SENSUS que tiene más de 70000 conceptos relacionados, al linkarlos los términos relevantes aparecen relacionados con otros conceptos de la Ontología grande que a su vez estaban relacionados con otros conceptos, lo que se hace posteriormente es extraer la parte de la Ontología que se ha relacionado con nuestros conceptos, y la añadimos a nuestra Ontología específica. Si un concepto no existiese en la Ontología grande se añade manualmente y se extrae la parte de la Ontología relacionada, además si en un subárbol hay varios nodos relacionados con un término relevante se extrae todo el subárbol, por

la teoría de que si varios nodos de un subárbol tienen relación con otro nodo los demás nodos del subárbol tendrán una probabilidad muy alta de estar también relacionados con éste. Por tanto esta metodología provee compartición de conocimiento ya que varios dominios estarán modelados con un mismo esqueleto de conocimiento.

Una de las metodologías que también tenemos que dar más énfasis es la denominada **METHONTOLOGY** [14] esta metodología se creo en el Laboratorio de Inteligencia artificial de la **Universidad Politécnica de Madrid** (UPM), para crear Ontologías tanto partiendo de cero y identificando uno por uno los conceptos más relevantes como arañando partes de grandes Ontologías al igual que el método anterior. La metodología **METHOLONGY** incluye métodos para el proceso de desarrollo de una Ontología, para el ciclo de vida basado en etapas diferenciadas con prototipos como outputs y técnicas particulares para cumplimentar eficientemente cada actividad.

El proceso de desarrollo de una Ontología identifica que tareas se han de ejecutar para construir una Ontología (Organización, Control, Seguridad, Especificación, Adquisición del Conocimiento, Conceptualización, Integración, Formalización, Implementación, Evaluación, Mantenimiento, Documentación y Gestión de la configuración).

El Ciclo de vida identifica las etapas por las que pasa una Ontología en su vida, así como las relaciones con los otros ciclos de vida de las otras Ontologías.

La principal fase de **El proceso de desarrollo** es la Conceptualización, y herramientas como WebODE, OntoEdit etc dan soporte a esta etapa del desarrollo. Muchas otras herramientas pueden usar esta metodología para el desarrollo de sus Ontologías.

Otra metodología que mencionaremos es la **On-To-Knowledge methodology** [12], la cual incluye una identificación de metas, objetivos, que son logrados con herramientas de soporte para el manejo de conocimiento.

Las principales fases que describe esta metodología son:

- **Kick-off phase (Fase inicial):** En esta fase se adquieren los requisitos específicos de la Ontología y se relatan en el libro de requisitos específicos (ORSD), describiendo en que dominio se ha de diseñar la Ontología, y que campos ha de abarcar.

  En esta fase se diseñan también dos elementos relacionados entre ellos, primero se determinan los conceptos con los que va a trabajar la Ontología según el dominio específico y después se determinan las relaciones entre estos conceptos que normalmente son de generalización-especialización, por tanto se describe ya la jerarquización de conceptos con todas sus relaciones.

- **Refinement phase (Fase de refinamiento):** Esta fase consta de 2 etapas, en la primera etapa se chequea las relaciones y axiomas entre los conceptos, formalizando gráficamente el árbol de la Ontología, con todos los conceptos y sus atributos definidos así como las relaciones entre éstos.

  En la seguna etapa se edita la Ontología con un editor de Ontologías como puede ser WebODE u Ontoedit, y através de éstos se exporta a un lenguaje formal como XML, RDF u otros.

- **Evaluation phase (Fase de Evaluación):** En esta última etapa se evalúa que la Ontología cumpla los requisitos especificados, y también que el lenguaje utilizado para exportar la Ontología mantenga la misma jerarquización de conceptos y relaciones entre ellos. Por tanto la Ontología una vez diseñada tiene que estructurar de forma eficiente toda la información comprendida en un dominio específico, sino fuese así esa Ontología no sería útil a la hora de utilizarla como soporte en otras aplicaciones.

- **Maintence phase (Fase de mantenimiento):** En esta fase se controla la Ontología bien sea manualmente o ayudándose de herramientas de soporte, de los cambios que se le han de aplicar a causa del rápido y dinámico cambio en la información que se produce en entornos distribuidos como es el caso de Internet.

Conclusiones

Una vez vistas diferentes tipos de metodologías, podemos decir que cada una de ellas no es mejor ni peor que las otras, sino que están diseñadas para cubrir la construcción de un tipo en concreto de Ontologías. Por ejemplo si comparamos **KACTUS** con **SENSUS,** la primera se parte de una base de conocimiento (KB) y se ejecuta un proceso de abstracción para determinar los conceptos principales que queremos representar y relacionar, la segunda en cambio se basa en el esqueleto extraído de una gran Ontología y a partir de allí se especializa en la Ontología que quieres crear.

El tipo de metodologías no solo se puede comparar por el tipo de Ontologías del cual tratan sino también por el grado de dependencia entre la Ontología desarrollada y la aplicación en la cual la queremos utilizar. Por ejemplo el método **KACTUS** y la metodología **O-To-Knowledge** son aplicaciones dependientes, ya que la Ontología se construye en base a la aplicación la cual la va a usar. En cambio la metodología de **Grüninger y Fox** y el método basado en **SENSUS,** son aplicaciones semi-dependientes, y finalmente el método **CYC** , el método **Ushold y King,** y la metodología **METHONTOLOGY** son aplicaciones totalmente independientes de la construcción de la Ontología y su posterior uso.

Por último hay que decir que ninguno de los métodos o metodologías presentados, están lo suficiente avanzados para compararlos con métodos aplicados en otros campos como la Ingeniería del Software, la metodología más completa nos la da **METHONTOLOGY,** la cual es recomendada por la **FIPA** para la construcción de Ontologías.

Además una de las causas por las que este tipos de métodos y metodologías no tienen el desarrollo que tienen otros, es la no unificación de éstos, cada grupo de investigación aplica su método para construir metodologías y no hay un estándar unificado que utilice toda la comunidad ontológica.

2.4. Ciclo de Vida de una Ontología

Ciclo de Vida en el diseño de Ontologías

El ciclo de vida de una Ontología es el mismo que el que pueda tener el diseño de cualquier software.
Aquí especificaremos un poco más al detalle las fases de diseño que hemos descrito anteriormente

Identify purpose and scope (Identificación del objetivo) : En esta etapa se identifican los requisitos igual que en la fase inicial o **Kick off phase.**

Knowledge Acquisition (Adquisición del conocimiento) : En esta etapa se recopila la información que la Ontología ha de estructurar. Esta información se puede conseguir de libros, documentos científicos o incluso de otras Ontologías.

Conceptualisation (Conceptualización) : En esta fase se identifican de manera informal los conceptos clave que tratará la Ontología y sus relaciones. Trata de describir informalmente como se estructurará toda la información extraida en la etapa anterior, para que ésta quede representada eficientemente.

Integrating (Integración) : Una Ontología se puede diseñar desde 0 partiendo de una dominio específico y extrayendo su información, pero gracias a la estructura de las Ontologías y su característica de "**extendible**" una Ontología se puede especializar de otra, y en esta etapa de estudia si la Ontología que queremos diseñar puede o no especializar de otra ya existente.

Encoding (Progamación): En esta etapa, se escoge el lenguaje que se utilizará para exportar una ontotología, ya puede ser en un sistema basado en frames como es nuestro caso o a nivel de entidades lógicas relacionadas, así como se describirán formalmente la terminología del dominio el cual queremos formalizar la información.

Evaluation (Evaluacíon) : Del mismo modo que en la fase de evaluación que hemos descrito anteriormente, en esta etapa se evalúa que la Ontología final ya exportada en un lenguaje concreto mantenga las mismas relaciones que se habían descrito en la etapa de **Conceptualisation,** también se evalúa si la Ontología presenta redundancias en la información representada.

Maintence (Mantenimiento): Como ya hemos visto antes, no solo basta con crear un Ontología, ésta se ha de ir actualizando dinámicamente conforme la información del dominio que modela también cambie, sobre todo si la Ontología se utiliza como soporte para la extracción de conocimiento de la web como es nuestro caso.

2.5. Uso y Aplicaciones

Las Ontologías como ya hemos comentado son utilizadas para varias tareas, a continuación describo algunas de ellas para ver el amplio abanico de posibilidades que ofrece esta estructuración de la información.

- Como **repositorios** para la organización de conocimientos e información, tanto de tipo corporativo como científico.
- Como herramienta para la **adquisición de información**, en situaciones en la que un equipo de trabajo la utiliza como soporte común para la organización del dominio.
- Como herramienta de **referencia** en la construcción de SBC (Sistemas Basados en el Conocimiento).
- Para permitir la **reutilización** del conocimiento existente en nuevos sistemas.

Estos cuatro ejemplos demuestran la funcionalidad de una herramienta capaz de modelar toda la información existente en un dominio. Pero una funcionalidad que no hay que dejar escapar de las Ontologías es la de dar soporte a pequeñas y medianas empresas para que mantengan un conocimiento estructurado de la información que manejan, y este conocimiento pueda ser actualizado y renovado por consecuencia de los cambios en el entorno.

En las últimos años y gracias al desarrollo de las grandes tecnologías y a la distribución de un gran volumen de información contenido en entornos como es el WWW las empresas necesitan un soporte para el manejo de información que se adapte a sus necesidades y sobre todo para innovación eficiente que pasa primero de todo por tener una estructura consistente de información del dominio específico en el que trabajan.

Un manejo eficiente de la información puede:

- **Mejorar la utilización** del conocimiento empresarial, científico y técnico para crear nuevas oportunidades de mercado y así hacer crecer de forma notable sus beneficios.
- **Descubrir nuevas relaciones** con otros dominios heterogéneos y crecer potencialmente su base de conocimiento.
- **Conocer mejor el desarrollo** de mercado y guiar de forma más eficiente sus investigaciones para cubrir sus necesidades de mercado.
- **Fomentar una mejor preparación** de sus empleados y estructurar su negocio jerárquicamente basándose en la utilización que hace cada uno de sus empleados de la información disponible.
- **Ayudar a los jóvenes profesionales** teniendo herramientas de soporte, a hacer frente a los cambios bruscos que se originan hoy en día en sus profesiones.

Por tanto, disponer de un común y compartido entendimiento de un dominio específico puede ayudar a comunicar personas y empresas y también sistemas de aplicaciones (como por ejemplo la comunicación de agentes inteligentes). Este

conocimiento es crucial para poder automatizar el manejo de conocimiento sobre un dominio sin necesidad de la intervención humana.

2.6. Lenguajes de Ontologías

A principios de los años 90 se empezaron a crear las primeras bases de lenguajes de Ontologías, los cuales estaban basados en lógica de primer orden como el lenguaje KIF en combinación de lógica de primer orden y frames como es ONTOLINGUA [15,11] o en DL(Description Logics) como es el caso del lenguaje LOOM.

KIF es un lenguaje basado en lógica de primera orden creada en 1992 como un intercambio de conocimiento entre sistemas de razonamiento (KR). **ONTOLINGUA** el cual se construyo basándose en KIF fue desarrollado en 1992 por el laboratorio de sistemas de conocimiento (KSL) en la Universidad de Stanford. Éste combina los sistemas de frames y los cálculos de predicados de primer orden.

En cambio, **LOOM** [16] fue desarrollado a la vez que ONTOLINGUA en el Information Science Institute (ISI) en la Universidad de California, inicialmente no se pensó como un lenguaje de Ontologías sino de KBS (sistemas basados en el conocimiento). LOOM se basa en DLS y reglas de producción, y está provisto de clasificaciones automáticas de conceptos. Éste lenguaje también permite representar los siguientes conceptos: conceptos, taxonomías de conceptos, relaciones n-arias, funciones, axiomas, hechos y reglas de producción.

OCML [17] fue desarrollado posteriormente en 1993, en el **KMI** de la Open University, este lenguaje es muy parecido al **ONTOLINGUA**, de hecho la mayoría de definiciones que se pueden expresar en **OCML** son iguales que la forma de expresarlas en **ONTOLINGUA**, aunque también incluye algunas mejoras como, deducción y producción de reglas, y definiciones operacionales para funciones. **OCML** fue creado principalmente para construir Ontologías y modelar métodos que solucionen ciertos problemas.

Flogic [18], fue desarrollado en 1995, en la **Kalsruhe University**. Flogic (**F**rame **L**ogic), combinan Frames y lógica de primer orden, permitiendo representar conceptos, taxonomías sobre conceptos, relaciones binarias, funciones, instancias, axiomas y deducción de reglas.

El boom de Internet llego posteriormente e hizo necesario la aparición de lenguajes de Ontologías que explotasen las características del entorno web. A Estos lenguajes generalmente se les denomina **Web-based ontology languages** (lenguajes de Ontología basados en la web), o **Ontology markup languages** (lenguajes de Ontologías basados en etiquetas). Estos lenguajes están todavía en fase de desarrollo, están continuamente evolucionando, teniendo en cuenta que Internet es un medio que dinámicamente cambia y a pasos agigantados no es de extrañar que los lenguajes tengan que amoldarse a estos cambios para sacar el máximo provecho de este entorno.

Un ejemplo de estos tipos de lenguajes es **SHOE** [19] desarrollado en 1996 como una extensión de HTML, en la universidad de Maryland. Éste usaba etiquetas diferentes a las especificadas por HTML, y de esta forma podía insertar Ontologías en los documentos HTML. **SHOE** combina frames y reglas, y permite representar conceptos, taxonomías sobre estos conceptos, relaciones n-arias, instancias y deducciones con reglas, las cuales eran usadas por el motor de inferencia para razonar y obtener nuevo conocimiento.

Entonces apareció **XML** [20], el cual nada más crearse se mostró como un lenguaje estándar para el intercambio de información en la web, viendo este hecho la sintaxis de SHOE se modifico para poder usa XML, y en consecuencia otros lenguajes de Ontología adoptaron una medida similar y empezaron a edificar su sintaxis basándose en la de XML .

Ya en 1999 se creo **XOL** [21], el cual fue desarrollado por el centro de Intelogencia Artificial de SRI internacional, como una XMLización de un pequeño subconjunto de primitivas del protocolo OKBC, llamado OKBC-Lite. Es un lenguaje con muchas

restricciones, donde solo se especifican: conceptos, taxonomías de conceptos, y relaciones binarias. Este lenguaje no esta provisto de ningún tipo de mecanismo de inferencia, y su principal función era la de intercambiar información con Ontologías en el dominio de la medicina.

Posteriormente se creo **RDF** [22] que fue desarrollado por el W3C (World Wide Web Consortium) como un lenguaje basado en la web semántica para describir recursos web. En consecuencia a éste lenguaje salió RDF Shema [23], una extensión de RDF basado en primitivas de frames. No es muy expresivo, ya que solo admite representación de conceptos y relaciones binarias, y además no esta provisto, como otros lenguajes, de un motor específico de razonamiento.

Estos lenguajes habían establecido los fundamentos de la Web Semántica, en éste contexto 3 lenguajes más aparecieron como extensiones de RDF Shema, éstos eran OIL, DAM+ OIL y OWL.

OIL [24], fue desarrollado en el entorno del proyecto europeo On-To-Kkowledge. Éste añadía primitivas de RDF(s), y semántica de DLs. Este lenguaje incorporaba una máquina de razonamiento (**FaCT**), la cual se usaba para clasificar conceptos automáticamente.

Posteriormente, en diciembre del 2000, se creó **DAML+OIL** [15], que fue creado conjuntamente entre U.S y E.U, en el proyecto DARPA. Tanto **OIL** como **DAML+OIL,** permitían representar conceptos, taxonomías, relaciones binarias, funciones y instancias. Hoy en día se sigue poniendo un cierto interés para proveer a éste lenguaje de un mecanismo propio de razonamiento.

Finalmente, en el año 2001 el W3c formó un grupo llamado Web-Ontology (WebOnt), el objetivo de este grupo era crear un nuevo lenguaje de etiquetas, para tratar Ontologías en la futura Web semántica, este lenguaje se llamó **OWL** (Web Ontology Language). El input de este lenguaje fueron las características de **DAML+OIL,** las cuales ayudaron a crear la primera especificación de este lenguaje. Por último hay que resaltar que OWL esta dividido en dos estratos: **OWLlite** y **OWL.**

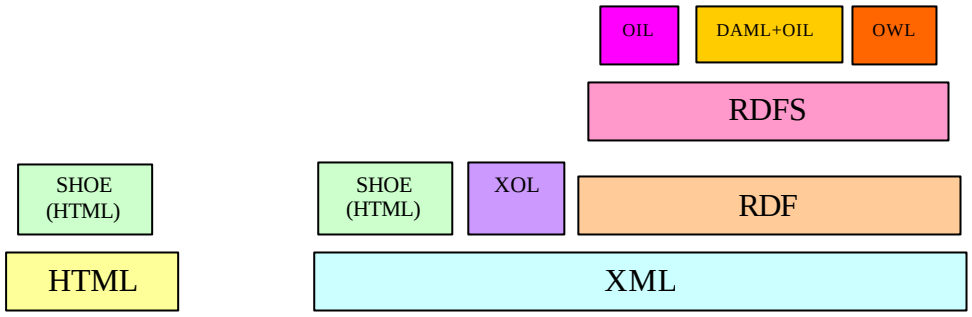


Fig. 1 Pila de lenguajes de Ontologías

Conclusiones

Una vez vistos los diferentes tipos de lenguajes de Ontologías que existen, haremos un breve resumen clasificatorio de éstos, según las aplicaciones que soporten y sus funcionalidades. Al final de las conclusiones, éstas quedaran plasmadas en una tabla, donde se representarán cada uno de los lenguajes con sus características más destacables.

Hay varias características las cuales comparten todos los lenguajes que hemos mencionado anteriormente, éstas son: ***Conceptos, organización de conceptos en taxonomías,, relaciones binarias, y instancias.*** Aunque todos los lenguajes comparten estas características no todos lo hacen de la misma forma, por ejemplo LOOM, OCML, OIL, DAML+OIL, y OWL son más expresivos que el resto de los lenguajes, ya que permiten la creación de particiones de un concepto en diferentes tipos de subclases.

Una característica que solamente comparten Ontolingua y SHOE, son las relaciones n-arias, donde en el resto de lenguajes para expresar un tipo de relación así tienes que descomponerlas en varias relaciones binarias.

Ontolingua, LOOM, OCML, OIL, DAML+OIL, y OWL, son lenguajes donde la creación y manejo de funciones se pueden representar de forma muy sencilla.

Los axiomas formales son la herramienta más poderosa para representar conocimiento en Ontologías. Y son usados para representar conocimiento que no puede ser representado con otro tipo de primitivas que contenga el lenguaje. Hay varios lenguajes que están provistos de axiomas formales para representar el conocimiento que tratan, éstos son: Ontolingua, LOOM, OCML, y Flogic.

Finalmente las reglas pueden ser definidas en LOOM y OCML y las acciones solo se pueden definir en Ontolingua (Aunque no se pueden ejecutar), LOOM, y OCML.

Los mecanismos de inferencia o razonamiento que aporta cada lenguaje son diversos. A excepción del mecanismo de inferencia que incorpora el lenguaje OIL llamado (FaCT), estos mecanismos se usan para deducir nuevo conocimiento a partir de una Ontología o incluso para chequear las relaciones entre los conceptos que puedan estar representados en un a Ontología. En los lenguajes LOOM y OIL, el mecanismo modifica automáticamente la clasificación existente entre los conceptos que se representan si se encontrase alguna inconsistencia entre las relaciones de éstos.

En resumen si nosotros queremos construir una Ontología, primero tenemos que estudiar para que aplicación queremos esta Ontología, en términos de expresividad y de servicios de razonamiento e inferencia, puesto que no todos los lenguajes nos proporcionan en mimo nivel de estas características. Por ejemplo un lenguaje que solamente aporte una representación y razonamiento básico, con taxonomías, conceptos y relaciones binarias, no será suficiente si trabajo con **heavyweight Ontologies**, y necesito hacer grandes razonamientos, o por ejemplo la existencia de traductores entre diferente lenguajes, no es buena si quieres asegurarte que no se va a perder información al pasarlo a un lenguaje determinado, ya que cada lenguaje tiene su forma de representar el conocimiento y muchas veces la traducción del conocimiento de un lenguaje a otro puede implicar que haya o bien pérdida de información, o que la clasificación de la información varíe de un lenguaje a otro y esa representación se aproxime mas o menos a la representación inicial que había determinado el usuario experto.

Tabla Resumen

	Onto-lingua	OCML	LOOM	FLogic	XOL	SHOE	RDF(S)	OIL	DAML
Conceptos Generales									
Metaclases	S	S	S	S	S	N	S	N	N
Particiones	S	X	S	X	N	N	N	S	S
Documentación	S	S	S	X	S	S	S	S	S
Atributos									
Atributos de instancia	S	S	S	S	S	S	S	S	S
Atributos de clase	S	S	S	S	S	N	N	S	S
Atributos locales	S	S	S	S	S	S	S	S	S
Atributos globales	X	X	S	N	S	N	S	S	S
Facets									
Valores de slots por defecto	N	S	S	S	S	N	N	N	N
Restricciones de tipo	S	S	S	S	S	S	S	S	S
Restricciones de cardinalidad	S	S	S	X	S	N	N	S	S
Documentación de slots	S	S	S	N	S	S	S	S	S
Taxonomías									
Subclase de	S	S	S	S	S	S	S	S	S
Particiones de subclases	S	X	S	X	N	N	N	S	S
Especializaciones	S	X	S	X	N	N	N	S	S
No subclase de	X	N	X	N	N	N	N	S	S
Relaciones/Funciones									
Relaciones n-arias	S	S	S	X	X	S	X	X	X
Restricciones de tipo	S	S	S	S	S	S	S	S	S
Restricciones de integridad	S	S	S	S	N	N	N	N	N
Definiciones operacionales	N	S	S	S	N	N	N	N	N

Axiomas									
Lógica 1er orden	S	S	S	S	N	X	N	X	X
Lógica 2nd orden	S	N	N	N	N	N	N	N	N
Axiomas declarados	S	S	N	N	N	N	N	N	N
Axiomas acoplados	S	S	S	N	N	N	N	N	N
Instancias									
Instancias de conceptos	S	S	S	S	S	S	S	S	S
Hechos	S	S	S	S	S	S	S	S	S
Claims	N	N	N	N	N	S	X	X	X

Leyenda

S: Sí contiene.

N: No contiene.

X: No contiene directamente pero ayudándose de otras herramientas puede desempeñar la misma función.

2.7. Editores de Ontologías

En los últimos años el número de medios y herramientas para la creación de Ontologías ha crecido exponencialmente. Estas herramientas tienen como objetivo dar soporte al proceso de desarrollo de Ontologías y como consecuencia a su uso. A continuación les hablaremos de los editores más relevantes.

La primera herramienta de Ontologías que se creó fue el ***Ontolingua Server***, [25] ésta se desarrollo en el laboratorio de sistemas de conocimiento (KSL) en la Universidad de Stantford. El ***Ontolingua Server*** apareció a principios del año 1990 y fue desarrollado

para facilitar el desarrollo de Ontologías implementadas en el lenguaje Ontolingua. Inicialmente la herramienta principal de la que proveía este editor era el editor de Ontologías aunque también incluía otros tipos de módulos. Como todos los editores de Ontologías definía varios lenguajes disponibles para poder exportar la Ontologías editadas, estos lenguajes eran ***LOOM, PROLOG ,CORBA'S IDL, CLIPS***, etc.

En la misma época apareció ***Ontosaurus*** [11] que fue desarrollado por el Instituto Científico de la Información (ISI), en la Universidad de California. ***Ontosaurus*** estaba provisto de dos módulos principales: el primero era un servidor de Ontologías el cual usaba LOOM como lenguaje para representar el conocimiento, y el segundo era como buscador web el cual utilizaba Ontologías implementadas en LOOM. Éste editor podía exportar sus Ontologías en ***Ontolingua, KIF,KRSS, y C++.***

En 1997 el Knowledge Media Institute(KMI) desarrollo en la Universidad Abierta el ***Tadzebao*** y el ***WebOnto*** [26]. WebOnto es un editor de Ontologías OCML. La principal ventaja de este editor sobre otros es que soporta la edición de Ontologías con varios usuarios a la vez, cosa que pueden trabajar con ella síncrona o asíncronamente.

En los últimos años, ha surgido una nueva generación de editores de Ontologías. Éstos han sido creados para integrar las Ontologías en los sistemas de información actuales. Todos estos editores permiten extender, especializar y modificar una Ontología, así como importarla y exportarla, o crearla gráficamente definiendo los conceptos que se representarán junto con sus relaciones. Por tanto todos estos editores permiten modelar una Ontología de forma que ésta cumpla los requisitos necesarios para el buen diseño de una Ontología.

Entre estos últimos editores de Ontologías nosotros podemos citar 3: ***Protege2000, WebODE*** y ***Onto Edit.***

Protege2000 [27] fue desarrollado por el **Standford Medical Informatic** (SMI), en la Universidad de Standford, y es la última versión que han sacado de **Protege.** Es una aplicación *standalone*, por tanto una aplicación que la instalas en un ordenador y no permite que varios usuarios puedan acceder a la vez y colaborativamente a una cierta

Ontología. También al igual que Ontoedit se caracteriza por ser una arquitectura extensible, la cual se basa en un conjunto de plug-ins donde tu puedes crear una aplicación para este tipo de herramienta y acoplarla a ella. La principal función que desempeña esta herramienta como todas las otras que hemos visto es la de editor de Ontologías. Pero como hemos comentado tiene una librería de plug-ins en la cual uno puede añadirle mas funcionalidades. Actualmente esta herramienta está provista de plug-ins , para poder importar y exportar una Ontología en los siguientes lenguajes: **Flogic, Jess, OIL, XML y Prolog,** acceder a OKBC que es un servidor de bases de conocimiento, restricciones de creación y ejecución con una herramienta llamada (PAL) y linkage de Ontologías con PROMPT [28].

WebOde y OntoEdit, son 2 de los últimos editores de Ontologías que han surgido. Estos dos editores han sido los que hemos escogido para utilizar con nuestras Ontologías, intentando hacer una comparación con la información exportada por ambas.

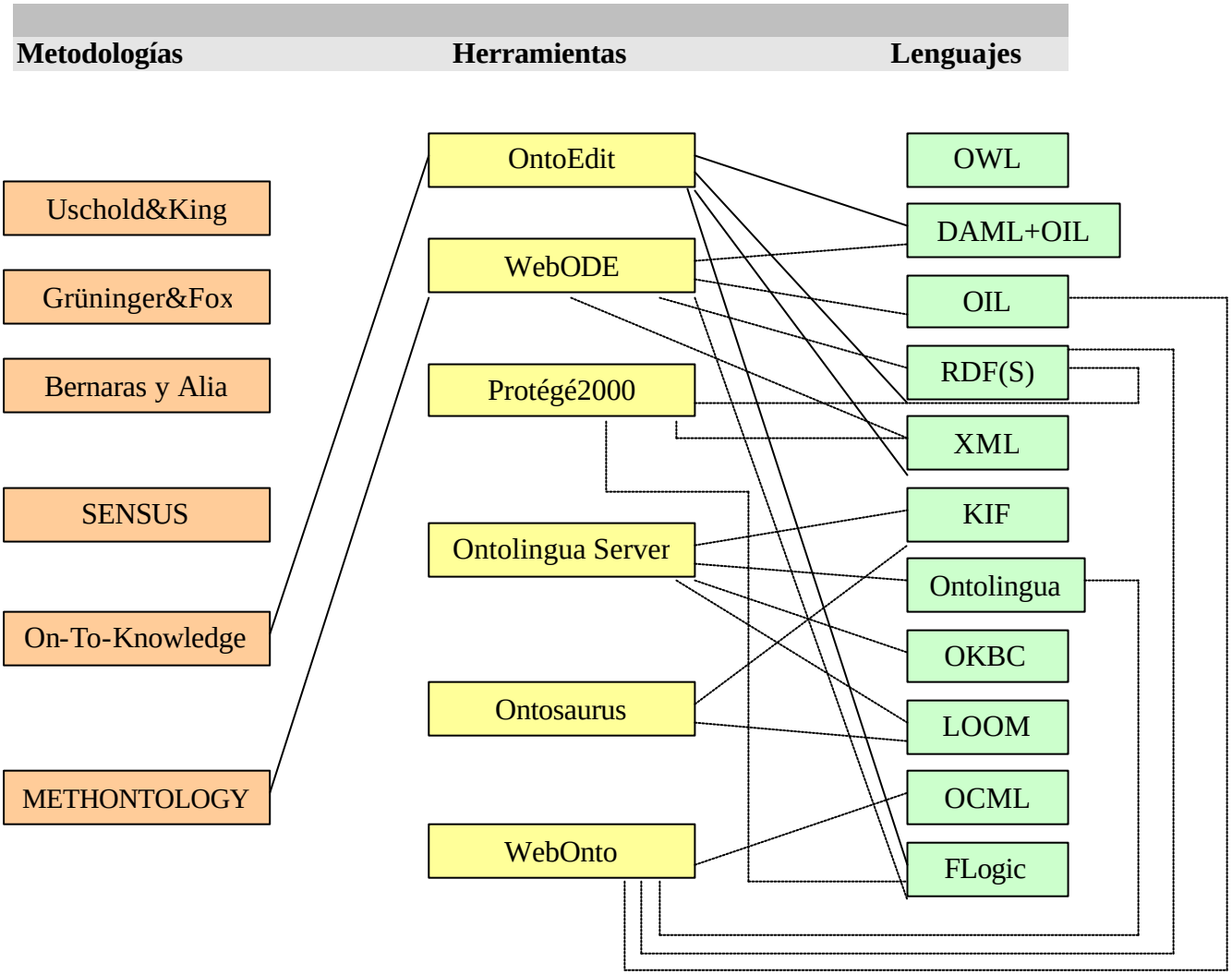
El objetivo de esta comparación entre estas dos herramientas era comprobar que sin depender de un lenguaje en concreto, el concepto de la Ontología se mantenía inalterable exportándola con uno y otro editor.
Pero antes de hacer alguna comparativa hablaremos un poco de cada una de estas herramientas.

WebODE [29] es el sucesor de ODE (Ontology Design Enviorement) [30], y fue desarrollado en el laboratorio de inteligencia artificial de la Universidad Politécnica de Madrid (UPM).
WebODE no es una aplicación standalone sino que esta en un servidor web instalado en el cual un usuario se puede conectar y desde allí trabajar con su Ontología. Por eso más de un usuario, siempre que tengan autorización, puede estar trabajando a la vez con una misma Ontología, cosa que hace mucho más flexible el trabajo sobre esta. Del mismo modo que muchos otros editores ***WebODE*** tiene la posibilidad de importar y exportar una Ontología en unos lenguajes definidos, estos lenguajes son: ***XML, RDF(s), OIL, DAML+OIL, CARIN, Flogic, JESS, Prolog***. También se pueden editar axiomas que son sentencias de lógica de primer orden, con WebODE Axiom Builder (WAB) [31], otra funcionalidad es la de documentar Ontologías, evaluarlas y también poder

entremezclarlas o integrarlas para poder crear otra Ontología ya que una de las mayores ventajas que tienen las Ontologías es que son fácilmente reutilizables, lo que facilita la categorización y clasificación de cierta información, teniendo Ontologías que ya clasifican parte de ésta.

OntoEdit [32], es el segundo editor de Ontologías que hemos estudiado, fue desarrollado por AIFB en la Universidad de Karlsruhe. Es una herramienta muy parecida a la que hemos estudiado anteriormente ***WebODE.*** Trabaja sobre un entorno extendible y flexible basado en una arquitectura de plug-ins, y sus dos principales utilidades son como editor y buscador de Ontologías. Como los demás editores con ***OntoEdit*** se pueden crear tus propias Ontologías y exportarlas a un lenguaje específico o así mismo importar una Ontología implementada en un lenguaje aceptado por ***OntoEdit*** y poder trabajar sobre ella. Los lenguajes que ***OntoEdit*** nos permite utilizar para el proceso de importación/exportación son: ***Flogic , XML ,RDF(s) , DAML+OIL*** etc.

GRÁFICO METODOLOGÍAS-HERRAMIENTAS-LENGUAJES



Leyenda

Metodología ---------- **Herramienta**: La herramienta soporta esta metodología.

Herramienta ---------- **Lenguaje**: Esta herramienta permite **exportar** una Ontología en este lenguaje.

Herramienta ---------- **Lenguaje**: Esta herramienta permite **importar** una Ontología en este lenguaje.

Herramienta ---------- **Lenguaje**: Esta herramienta permite **importar** y **exportar** una Ontología en este lenguaje.

3. USO DE ONTOLOGÍAS EN EL PROYECTO H-TECHSIGHT

3.1. Definición

H-TechSight (A Knowledge management platform with intelligence and insight capabilities for technology intensive industries), es un proyecto financiado por la Unión Europea y comenzó en Mayo del 2002 con una duración de dos años. En este proyecto colaboran los departamentos de Ingeniería Química y Ingeniería Informática de nuestra universidad.

Este proyecto pretende ayudar a pequeñas y medianas empresas a dominar la información con la cual ellas trabajan y a acceder a la información publicada en la red facilitándole herramientas para que puedan buscar de forma exhaustiva y eficiente esa información y facilitar herramientas que dinámicamente encuentren información relacionada que puede ser muy útil para su propia innovación.

En este proyecto distinguimos dos etapas. En la primera etapa se desarrollará una herramienta capaz de buscar inteligentemente información en la red a través de una Ontología previamente definida por un usuario experto, y representar dicha información al usuario para que pueda ver los resultados obtenidos.

La segunda etapa consiste en, una vez encontrada dicha información, procesarla y razonar con ella, eliminando links redundantes y encontrando relaciones entre los links encontrados de clases que no guardan una estrecha relación, de esta forma podemos obtener relaciones entre clases que el usuario a priori no había encontrado.

3.2. Objetivos del proyecto

3.2.1. Objetivo principal

El principal objetivo del proyecto h-TechSight es mejorar las capacidades de las empresas e industrias tecnológicas para poder guiar, evaluar, predecir y responder a los cambios y tendencias tecnológicos. Éste proyecto quiere hacer que estas empresas lleguen a ser grandes competidores en su dominio específico, gracias a herramientas que les sirvan de soporte para manejar eficientemente la información de la cual disponen.
Para lograr este objetivo, el proyecto desarrollará tecnologías capaces de actualizar la información con la cual trabajan, sistemática y automáticamente y creará mapas de conocimiento dinámico sobre un dominio intensivo específico.
Estos sistemas ayudaran a las empresas haciéndolas avanzar inteligentemente, dándoles un soporte para el mercado, el cual está sometido a muchos cambios que dinámicamente se pueden predecir.

Los objetivos de este proyecto, están dirigidos a crear KMPs (Knowledge Management Platform), es decir plataformas de gestión de conocimiento, las cuales ayudarán a las empresas en una eficiente dirección de ésta, la cual permitirá planificar y pronosticar las evoluciones que sufre el dominio específico al cual se dedican. El objetivo de estas plataformas de conocimiento, es convertir en conocimiento toda la información que se extrae de una web, portales industriales, intranets locales, o documentos guardados en repositorios, pero no solo la información se puede sacar electrónicamente sino que además la información se puede extraer de revistas dedicadas al sector industrial, documentos de investigación , reportajes y trabajos industriales.

Sobre todo, el mayor énfasis se lo hemos de dar a los recursos webs, ya que en éstos se almacena un mayor número de información , que aparece dispersa, y como tal es difícil de analizar y evaluar manualmente, y por eso nos tenemos que ayudar de herramientas de soporte que nos faciliten esa labor. Un KMP debe de ser validado y evaluado por el usuario experto previamente y los principales sectores profesionales.

En las industrias tecnológicas , la gestión de conocimiento requiere un estudio previo de los dominios técnicos a los cuales se dedica la empresa, este estudio de la información pretende predecir como ese dominio va a cambiar a lo largo del tiempo, si van a haber fusiones, si hay sectores que van a madurar, prosperar, o al contrario declinar. Poder pronosticar como va a cambiar el sector, objetivo de la empresa, facilita la gestión en general de una empresa, ya que puede rectificar en innovaciones que vea que no tiene futuro, y por lo contrario puede innovar hacia donde el mercado se engrandezca más.

Para todo ello, la empresa necesita disponer de la información de su dominio específico, pero la información “a secas” no es válida, sino que necesita poder hacer algún tipo de clasificación de esa información, abstraer la información mas general y especializar el resto, consiguiendo así árboles de conocimiento (Ontologías), los cuales representan, un soporte imprescindible para estas plataformas de conocimiento.

H-techSight introduce la gestión del conocimiento como herramienta indispensable para una gestión más eficiente de todos los campos de la empresa, esta gestión de la información puede determinar un perfil de dominio, puede predecir las tendencias de este dominio, con respecto a las demanda, y puede:

(i) Ayudar a las empresas a explotar sus oportunidades de mercado y a beneficiarse de su conocimiento.
(ii) Descubrir dependencias que antes no se sabían entre dominios heterogéneos y potenciar la transferencia de conocimiento a través de esos dominios.
(iii) Guiar a los investigadores a mejorar sus investigaciones sobre las necesidades de mercado.
(iv) Mejorar la preparación de los profesionales, y prepararles para poder reponerse a cambios, incrementar su eficiencia y reducir el desempleo.
(v) Ayudar a jóvenes científicos y ingenieros a perfilar mejor sus aptitudes respecto a sus profesiones.
(vi) Guiar a las universidades y institutos, a revisar su guía académica, con el objetivo de introducir a los jóvenes universitarios una educación que les

ayude a comprender como gestionar la información de la cual disponen y como utilizar herramientas que lo faciliten.

El objetivo de h-techSight es promover la practica de gestión del conocimiento a una nueva era donde se puedan lograr los requisitos arriba citados.

Para lograr estos objetivos h-techSight propone combinar agentes inteligentes que junto a Ontologías específicas de un dominio en particular puedan sistemática y automáticamente extraer los conocimientos deseados, para poder razonar con él. Estos desarrollos podrán servir de soporte, nunca suplantar, al las decisiones de toman los ingenieros especializados y así incrementar su potencial de forma inteligente dentro el mercado. El proyecto trabajará para crear plataformas de conocimiento basadas en una solución técnica avanzada, que tendrá que ser evaluada y validada siempre por un usuario experto. Los resultados del proyecto estarán empaquetados adecuadamente, y ofrecerán herramientas software que ayudarán a gestionar la información de una forma eficiente y contendrá servicios de consulta para poder facilitar el uso de la información disponible.

3.2.2. Objetivos Técnicos

Como ya hemos visto anteriormente, el objetivo principal de este proyecto es crear una plataforma de gestión de conocimiento, para facilitar a las empresas la gestión de la información de la que disponen, y además conseguir beneficiarse de lo que se ofrece en la web pudiendo clasificar la información y convirtiendo ésta en conocimiento inteligente.

Para lograr este objetivo, el proyecto se propone seguir los objetivos técnicos que se describen a continuación:

1. Investigar y promover...

- ...Ontologías dinámicas que sean capaces de auto-actualizarse cuando se producen cambios en el entorno. Promover la investigación de cómo se podrían modificar periódicamente los componentes de dichas Ontologías. Estudiar la posibilidad de crear mapas de conocimiento para productos, el mercado, los servicios y las tecnologías. Poner un cierto énfasis en la indagación de dependencias que aparecen a priori ocultas entre dominios heterogéneos, y encontrar también dependencias entre empresas y tecnologías.

- ...Agentes que puedan explotar automáticamente la información que esta disponible en las Ontologías. Los agentes extraen la información de Internet ayudándose del soporte de las Ontologías, y de esta forma subministrarán y constatarán la evolución que sufre la información depositada en entornos dinámicos como es Internet. El área de la química industrial es un claro ejemplo, ya que es un área selecta, debido a el gran número de relaciones que hay entre disciplinas individuales (ej. Petroquímica, Industria Farmacéutica, Biotecnología, tecnología medioambiental).

2. Desarrollar e implementar...

- ...Una plataforma de gestión de conocimiento para implementar la investigación de conceptos y permitir un grado efectivo de evaluación, proceso, análisis, organización y traducción de la información en conocimiento.

- ...Herramientas genéricas de gestión de conocimiento capaces de explotar la información lo más eficientemente posible una vez extraída de la web (sitios webs, portales, librerías electrónicas etc), y así aumentar las capacidades de la empresa en este aspecto y mantenerla competitiva. Estas herramientas deberán incluir motores de búsqueda para encontrar la información antes de convertirla en conocimiento y encontrar relaciones de dependencias que se desconocían entre dominios, con el propósito de relacionar mercados que puedan ayudarse mutuamente, o abrir nuevos mercados para una empresa que puedan darle beneficios mayores.

- ...Una interfaz donde se muestren los resultados que se han analizado para que el usuario experto pueda observar gráficamente, los resultados obtenidos y su evolución a lo largo de tiempo. En el desarrollo de los productos se utilizará feedback (“Marcha atrás”) para poder refinar y mejorar tanto el entorno como las herramientas.

3. Validar y evaluar...

- …probando y testando el software con problemas reales de la industria. La Ontología dinámica junto con la información que almacena debe ser evaluada utilizando una evaluación iterativa donde en cada etapa se debe comprobar que no existan redundancias en la información, que los conceptos plasmados mantengan las relaciones iniciales, y que en definitiva mantenga una cierta consistencia con lo que inicialmente se quería representar. La Ontología, la función que desempeñan los agentes, y en general la plataforma de gestión de conocimiento, ha de ser testada y comparada con la tecnología estándar y con los programas actuales con el mismo objetivo. El objetivo de estas pruebas es el de evaluar los beneficios que puede tener este nuevo desarrollo, evaluar las nuevas características que añade este nuevo software, y aclarar el potencial de estos nuevos desarrollos.

- …el entorno en el cual va a trabajar la nueva plataforma de conocimiento al igual que las herramientas de ayuda a las tomas de decisiones. También se ha de evaluar la habilidad con la que se puede determinar nuevos cambios o tendencias, descubrir nuevos conceptos, y dar a conocer dependencias entre diferentes dominios en un entorno determinado.

4. Distribuir…

- …el nuevo software al publico con una versión limitada de la plataforma de conocimiento, un motor de búsqueda y funciones de soporte.
- …demostrando y promocionando el programa integrado a las principales industrias europeas, y internacionalizándolo mediante ofertas que combina herramientas que proporciona el software, y servicios de consulta.

- …a través de grupos de trabajo, que tengan un foco de influencia y interacción con industrias intensivas en un dominio tecnológico concreto.

3.2.3. Análisis del proyecto

Las industrias altamente tecnológicas, así como muchas profesiones asumen un particular perfil dinámico, que se ve frecuentemente y dramáticamente afectado por los cambios. La nueva dinámica está enfocada hacia nuevos desarrollos científicos e ingenieros que rediseñan la forma de actuar de las industrias para guiarles en los cambios que sufre la información o para predecirles esos cambios, creando nuevos puestos de trabajo y especializando los ya existentes o incluso alejar a las profesiones de los mercados con una bajo índice de demanda..

El dinamismo que han de tomar las empresas e industrias respecto a los frecuentes cambios en la información disponible presenta interacciones entre estas, y la evaluación

de las tendencias que tomará el mercado muchas veces pueden representar problemas. Aunque muchas veces el uso de plataformas de conocimiento en disciplinas concretas descuida la razón que guía a una empresa a prosperar que es el dinamismo: saber actuar a cambios en el entorno y saber encontrar y abrirse paso en mercados que aparentemente no tenían ningún tipo de relación.

Éste es el principal objetivo de las industrias tecnológicas, el representar en conocimiento la información que disponen para poder responder a todo tipo de cambios en el entorno.

Como hemos comentado anteriormente, el proyecto h-techsight, no solo pretende extraer información y representarla en conocimiento, sino que quiere poder determinar la dinámica que sigue este conocimiento, si a lo largo del tiempo en un mismo dominio aparecen nuevos conceptos que puedan representar un beneficio para empresas que se dedican a ese dominio, estudiar esos cambios favorece, a investigar como evolucionan los dominios y responder a esas evoluciones. Las Ontologías deben también poder actualizarse dinámicamente para poder representar el conocimiento a medida que este cambia.

Todo usuario experto puede crear una Ontología con información relevante de su empresa para poder analizar los cambios que se originan en la información y poder actuar a tiempo. En definitiva el proyecto h-techsight quiere ayudar a las industrias tecnológicas a saber gestionar la información que procesan, facilitándole herramientas que automáticamente extraigan esa información la conviertan en conocimiento, razonen con ella y puedan sacar conclusiones futuras.

Una parte importante de este proyecto, como ya hemos comentado es la extracción de información de la Web, para lograr esto, el proyecto h-techsight ha diseñado una plataforma multi-agentes, que implementa un buscador Web basado en Ontologías, para poder extraer de forma eficiente páginas relacionadas con un dominio en concreto, y razonar con ellas pudiendo obtener con ello más conocimiento. A continuación se muestra gráficamente la estructura de este sistema multi-agentes y se explica la función de cada uno de los componentes que la integran:

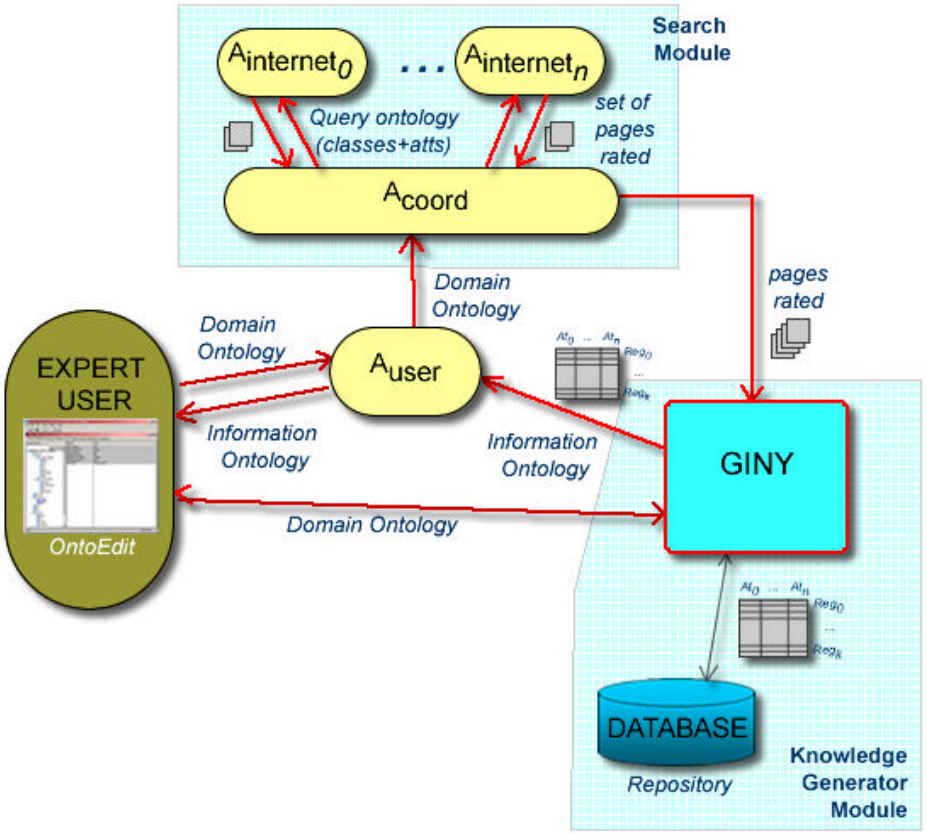


Fig. 2 Esquema una plataforma de gestión de conocimiento basada en agentes

En este esquema se representan los componentes que describen una plataforma basada en el conocimiento.

El ***USER EXPERT*** (usuario experto), es el usuario que define su propia Ontología, para definir su Ontología puede utilizar algunas de las pautas que se recogen en las diferentes metodologías que hemos estudiado, dependiendo del tipo de uso y aplicación para la cual va a utilizar esta plataforma elegirá una u otra. Una vez, el usuario experto haya definido su Ontología ayudándose de las herramientas que también hemos visto, la cargará en la plataforma a través de una interfície de usuario que puede ser una aplicación standalone o una aplicación web, una vez cargada, el ***USER AGENT*** (agente de usuario) accederá a la Ontología, y se la transferirá al ***COORDINATOR AGENT***

(agente coordinador), el cual dependiendo del número de ***Ainternets*** (agentes de internet) que haya distribuirá la información que contiene la Ontología. Como la Ontología estará representada mediante clases, las cuales estarán relacionadas formando jerarquías de padre-hijo, el agente coordinador, distribuirá a cada agente de internet un número de clases en concreto, una vez los agentes de internet hayan recibido esas clases, se conectarán a la red y buscaran páginas relacionadas con las clases que contengan. Una vez se hayan filtrado las páginas más importantes o que aparentemente contienen más relación con las clases objeto de búsqueda, se introducen dentro de la Ontología, con lo cual cada clase tendrá una serie de links encontrados. Una vez la Ontología dispone para cada clase de los links encontrados se aplica razonamiento sobre ella, con lo cual las páginas redundantes se eliminan y se intentan encontrar relaciones entre clases que a priori el usuario experto no hubiese podido encontrar.

El ***GYNI*** es un mecanismo de razonamiento, el objetivo del cual es analizar las páginas que se han encontrado para transformar la información que éstas contienen en conocimiento, pudiendo con ello tanto encontrar relaciones entre dominios heterogéneos, como encontrar información que las empresas no habían tenido en cuenta y que puede ser relevante para ellas etc.

Y por último el ***REPOSITORY,*** es como el propio nombre indica, un repositorio de Ontologías, donde guardamos las Ontologías que hemos recuperado de las diferentes búsquedas para luego ser objeto de estudio, con esto podríamos entre otras cosas analizar como va cambiando la información de un dominio específico a lo largo del tiempo. El usuario experto también puede utilizar este repositorio como base de datos para almacenar la información que ha ido adquiriendo, y estudiar las tendencias del mercado o las fusiones entre dominios que se puedan producir.

4. PROYECTO FINAL DE CARRERA

4.1. Descripción del proyecto

Una gran parte de mi proyecto ha estado enfocado, como ya habéis visto, al estudio de las Ontologías, así como todos los elementos que rodean a la creación de una Ontología como lenguajes de Ontologías, editores, y metodologías usadas para crear Ontologías.

Pero a parte de profundizar en el tema del análisis de las Ontologías, mi proyecto está enfocado a ayudar en una pequeña parte al proyecto h-techsight. Esa ayuda esta repartida en los siguientes puntos:

- Creación de una Ontología enmarcada en el dominio de los biosensores, utilizando el OntoEdit y el WebODE como editores.

- Razonamiento de la información que contienen la Information Ontology, para mirar la posibilidad de encontrar relaciones entre conceptos de la Ontología que el usuario a primera vista no hubiese podido encontrar.

- Utilización de un repositorio de Ontologías para guardar la Information Ontology, en vista a estudiar como cambia la información que contiene ésta dinámicamente.

Cada uno de estos puntos está desarrollado en los siguientes apartados, los cuales están ordenados por la secuencia lógica que seguí a la hora de desarrollar el proyecto.
Primero de todo el estudio de la Ontología concreta con la que iba a trabajar, después el razonamiento que he tenido que aplicar sobre ésta, y por último el estudio de algún tipo de repositorio RDF, que permitiera almacenar Ontologías en este formato para luego extraerlas y analizarlas.

4.2. Ontología utilizada

La Ontología que he tenido que modelar para este proyecto en los dos editores de Ontología propuestos (OntoEdit , WebODE), esta relacionada con el dominio Industrial. Esta Ontología pretende conceptualizar, más concretamente el dominio de los Biosensores. Como ya sabemos una Ontología pretende conceptualizar un dominio para tener conocimiento sobre unos conceptos que a priori pueden estar relacionados.
Esta Ontología sirve de base fundamental para el proyecto H-techsight, donde teniendo como elemento de partida una clasificación lógica de los conceptos que pueden abarcar este dominio, se puede extender a más conocimiento que a priori no se conocía, pudiendo encontrar relaciones con otros dominios y así ayudar a las empresas a extender su mercado, o incluso a beneficiarse de otros sectores que aparentemente no tienen relación alguna.

Esta Ontología es utilizada por los agentes del sistema, como soporte para poder encontrar páginas en Internet que puedan aportar conocimiento a una empresa en concreto, estas páginas se seleccionan teniendo en cuenta la estructura de la Ontología, por lo que la forma de conceptualizar la información determinará la búsqueda final.

Dos Ontologías jerárquicamente diferentes aunque representen el mismo dominio y los mismos conceptos, darán resultados diferentes, por tanto la forma de representar un dominio en una Ontología no es trivial, y la extensión que tenga ésta, determinará el grado de búsqueda del sistema, cuantos más conceptos se representen en la Ontología más grande será el abanico de posibilidades para encontrar relaciones con otros dominios o conceptos o directamente ayudará a encontrar un volumen de información mayor, que se convertirá “teóricamente” en una mayor conocimiento adquirido.

Ontología de Biosensores

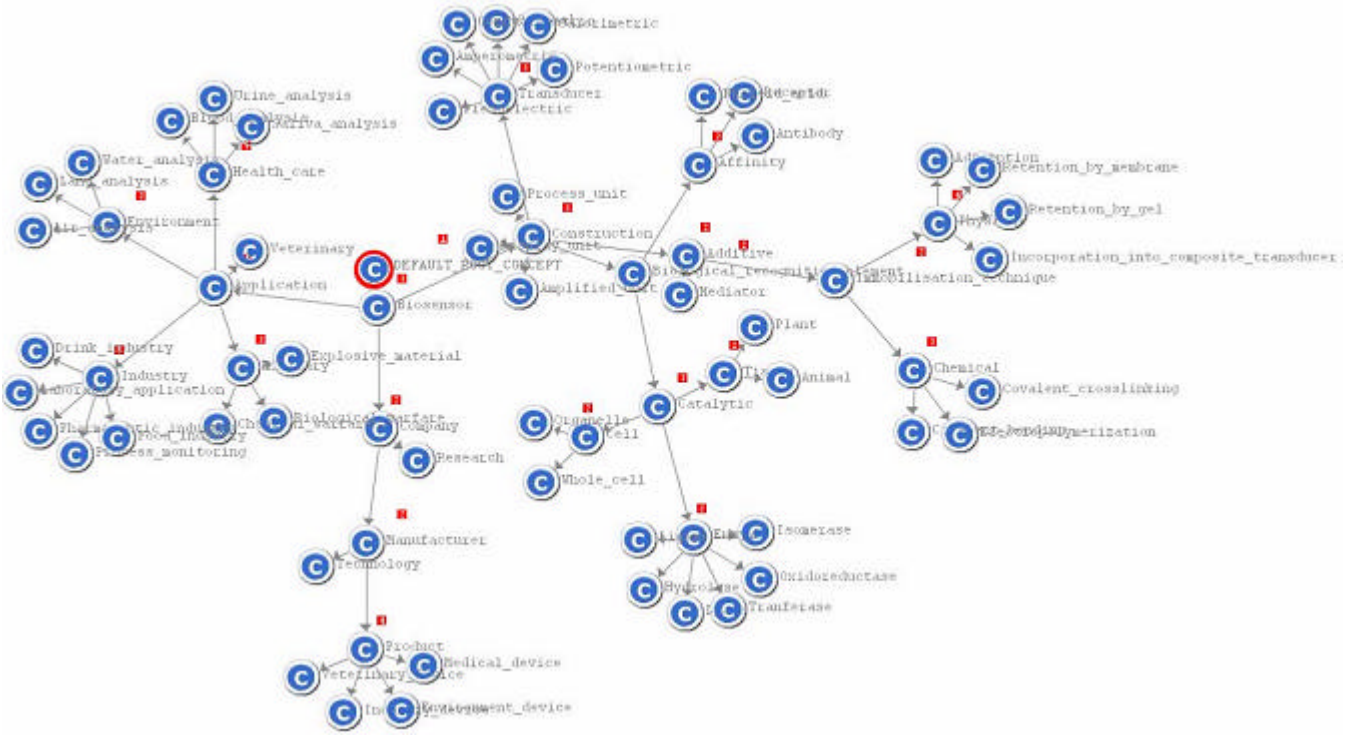


Fig. El concepto representado con una pelota rodeada de un circulo rojo, es la clase raíz Default_Root_Concept, el resto son clases donde unas heredan de otras. Las instancias se representarían con pelotas verdes, en esta Ontología no se ha definido ninguna.

En nuestra Ontología no se han definido instancias, porque el objetivo de estas Ontologías es generalizar un dominio entero, para encontrar conocimiento sobre todo el dominio, las instancias implicarían personalizar las búsquedas a compañías concretas, o tipos de biosensores concretos, pero ese no es el objetivo de H-techSight.

Teniendo en cuenta el tipo de Ontologías con las que los agentes del sistema multiagentes trabajan podemos definir tres tipos:

- **Domain Ontology**
- **Query Ontology**
- **Information Ontology**

Domain Ontology

La Domain Ontology, es la Ontología que se genera con alguno de los editores de Ontologías que hemos estudiado, el gráfico que he mostrado donde aparece representada la Ontología de biosensores es una Ontología del tipo domain ontology, donde se representan conceptos, propiedades y relaciones sin ningún tipo de información adicional.

Query Ontology

La Query Ontology, es la domain ontology una vez se ha repartido entre los agentes de internet para realizar la búsqueda. El user agent contabiliza el número de agentes que hay en el sistema y dependiendo de éste extrae trozos de la domain ontology para repartirlos entre éstos que lo utilizarán de input para realizar la búsqueda.

Information Ontology

La Information Ontology, es la domain ontology una vez se le ha añadido información adicional, esta información no es más que las páginas webs encontradas por los agentes. Por tanto todas las clases ahora tendrán una propiedad más en forma de lista para guardar las páginas webs que están relacionadas con ésta.

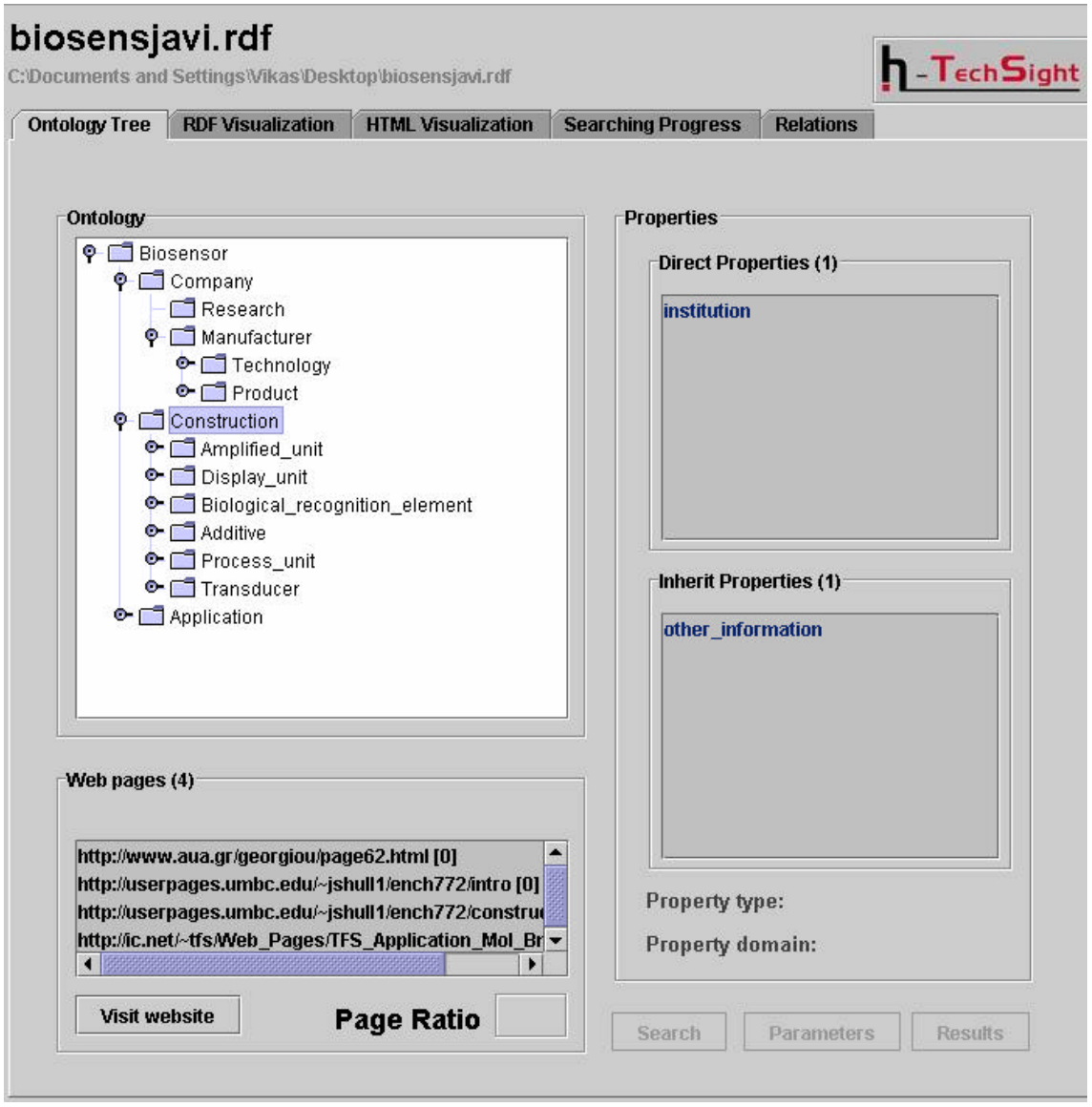
Una página web, se representa como una pseudoinstancia de la clase a la que pertenece, ya que no se puede considerar una instancia real porque no contiene los mismos atributos que la clase padre, sino que contiene atributos propios de la búsqueda como:

- **Links:** Que es una lista donde se representan los links que contiene esta página web
- **Rate:** Representa un rátio de puntuación que se le da a cada página, lo podríamos considerar el peso de la página.
- **Url:** Es la dirección web de la página.

- **Title:** En este atributo se representa el nombre de la clase a la que pertenece dicha página.

Las páginas webs heredan de la clase **InformationURL,** que es la que contiene estos atributos, por lo que las clases representadas en la domain Ontology contendran un atributo que será una lista de elementos del tipo InformationURL.

En la siguiente ventana vemos una parte de la interfaz de usuario, donde se representa la Information Ontology y se pueden apreciar las páginas webs encontradas por los agentes.



En esta figura se representa la Information Ontology, en la ventana principal podemos ver la Domain Ontology, y más abajo vemos como situándonos encima de cada clase nos saldrán las páginas webs pertenecientes a ésta, en este caso las páginas webs que se aprecian pertenecen a la clase **Research.** En la parte derecha de la figura nos aparecen además, los atributos propios (arriba) y heredados (abajo) que contiene cada clase.

En la interfaz de usuario se pueden cargar todo tipo de Ontologías, si quisiéramos cargar una Domain Ontology, se apreciarían las mismas cosas excepto el recuadro con las páginas webs encontradas, que al no contener información se mantendría en blanco.

4.3. Análisis de los editores utilizados: OntoEdit y WebODE

OntoEdit

OntoEdit, es uno de los dos editores que he utilizado para crear mi Ontología, esta Ontología representa conceptos sobre el dominio de los biosensores. La Ontología la creé utilizando esta herramienta y luego la exporté en formato RDF, ya que los agentes que hemos descrito ya anteriormente, utilizan este formato para leer una Ontología. A continuación describiremos que tipo de herramienta es OntoEdit, cual es su arquitectura, como se importa y exporta una Ontología determinada y sus principales funcionalidades.

OntoEdit fue desarrollado por AIFB en la Universidad de Karlsruhe, OntoEdit es un entorno de ingeniería de Ontologías, en el cual se desarrollan, editan y mantienen todo tipo de Ontologías usando editores gráficos.

Ontoedit se basa en una arquitectura flexible de plug-ins. Este tipo de arquitectura permite que esta herramienta sea fácilmente extensible por el usuario experto, que según sea el entorno para el que quiera utilizar Ontoedit, o según sean las funcionalidades que desea adquirir, puede añadir unos u otros plug-ins, pudiéndose hacer una herramienta hecha a medida.

Aún así OntoEdit esta disponible en tres tipos de versiones:

OntoEdit libre

Esta versión de OntoEdit recoge todas las características necesarias para modelar una Ontología. La importación y exportación de Ontologías en los estándares RDF y DAML también están disponibles en la visualización del modelo.

Funcionalidad	Tipo
Conceptos jerárquicos,Atributos,Relaciones	Base
Multilenguajes(Export/Import)	Base
Manejo multiple de Ontologías	Base
RDF(S)	Export&Import filter
OXML	Export&Import filter
DAML+OIL	Export&Import filter
Flogic	Export&Import filter
Excel	Import filter
Directorios	Import filter
Editor de instancias	Plug-in
Conceptos con relación disjoint	Plug-in
Visualizador	Plug-in

OntoEdit libre es una versión de OntoEdit limitada ya que como máximo puedes manejar 50 conceptos, 50 relaciones, y 50 instancias.

OntoEdit

Ontoedit, contiene todas las características de OntoEdit libre, mas un plug-in adicional llamado **Domainlexicon,** y no tiene limitación en el manejo y relaciones de conceptos.

Funcionalidades	Tipo
OntoEdit libre	Base
Domainlexicon	Plug-in
No hay limitaciones en el manejo de conceptos y relaciones	

OntoEdit Professional

OntoEdit profesional esta basado en OntoEdit, y permite a los usuarios añadir reglas, consultar la información que contiene la Ontologías, utilizar el mecanismo de inferencia para chequear la consistencia de una Ontología, importar una Ontología de una base de datos y exportarla en la misma ubicación.

Todas estas funcionalidades que añade se añaden con cada uno de los plug-ins que describiremos en la siguiente tabla. El plug-in de importar y exportar de una base de datos soporta todas las bases de datos con una interfaz JDBC, como por ejemplo **MSSQL Server, Oracle, DB2 y MySQL.**

Funcionalidades	**Tipo**
OntoEdit	Base
Editor Textual de reglas	Plug-in
Editor gráfico de reglas	Plug-in
SQL-Shema	Export filter
SQL-Shema	Import filter
Axiomas básicos	Plug-in
Inferencia	Plug-in
Query Tool (Herramienta para consultas)	Plug-in
OntoMap	Plug-in

Plugins Externos

Ya hemos visto los tres tipos de versiones de OntoEdit que han salido al mercado, pero como hemos dicho gracias a la arquitectura basada en plug-ins un usuario experto puede decidir que tipo de funcionalidades le quiere dar a su herramienta y cuales no, y así crearse una herramienta hecha a medida que se adapte a todo tipo de entornos y usos.

OntoEdit habilita una opción en su interfaz para poder importar plug-ins de una forma cómoda y rápida, en la cual podrás importar un plug-in o eliminarlo.

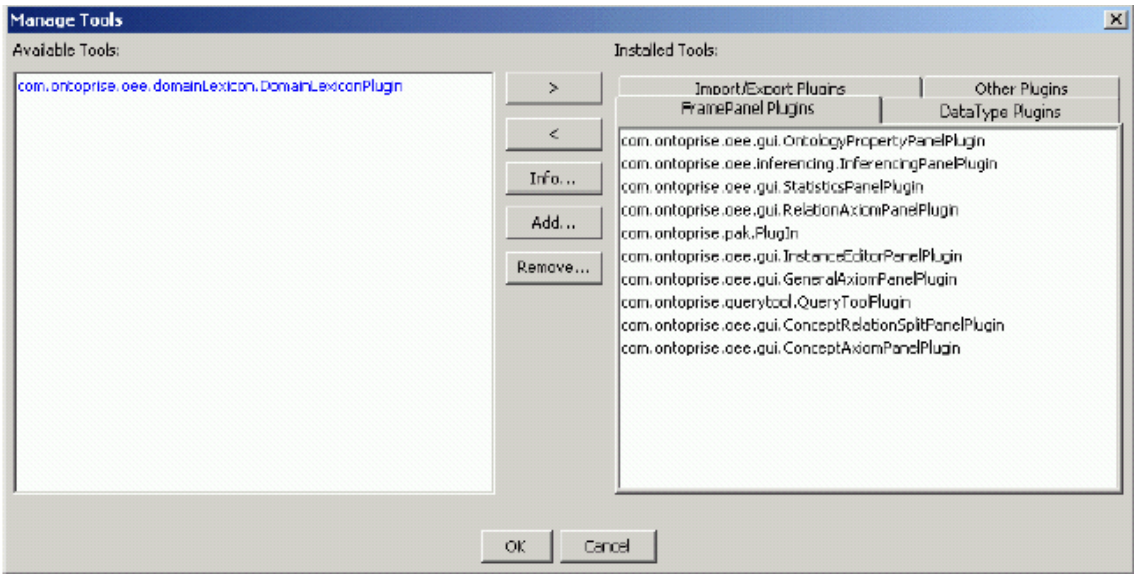


Fig. Ventana donde se puede cargar un plug-in o eliminarlo

En la ventana de la izquierda se listan los plug-ins conocidos, que son los plug-ins que ya están disponibles en las versiones de OntoEdit antes citadas. En el lado derecho de la ventana , aparecen el resto de plug-ins divididos en tres tipos: **Plug-ins de importación y exportación, FramePanel Plug-ins y otros plug-ins.**

Plug-ins para un modelage básico

De los plug-ins del tipo FramePanel plug-in podemos destacar seis que son básicos, estos son:

- Concept & relations
- Instances
- Relation axioms
- Disjoint concepts
- Identification
- Metadata

Concept & relations

Utilizando este plug-in, un usuario, puede editar crear y eliminar conceptos de su Ontología, así como relaciones.

Los conceptos están representados formando un árbol jerárquico, ya que mantienen una relación del tipo **is-a,** aunque un usuario puede crearse un tipo de relación entre dos conceptos en concreto. Los atributos de los conceptos también se definen aquí ya que no son más que relaciones con tipos estándares del lenguaje (String,Integer,Float, etc)

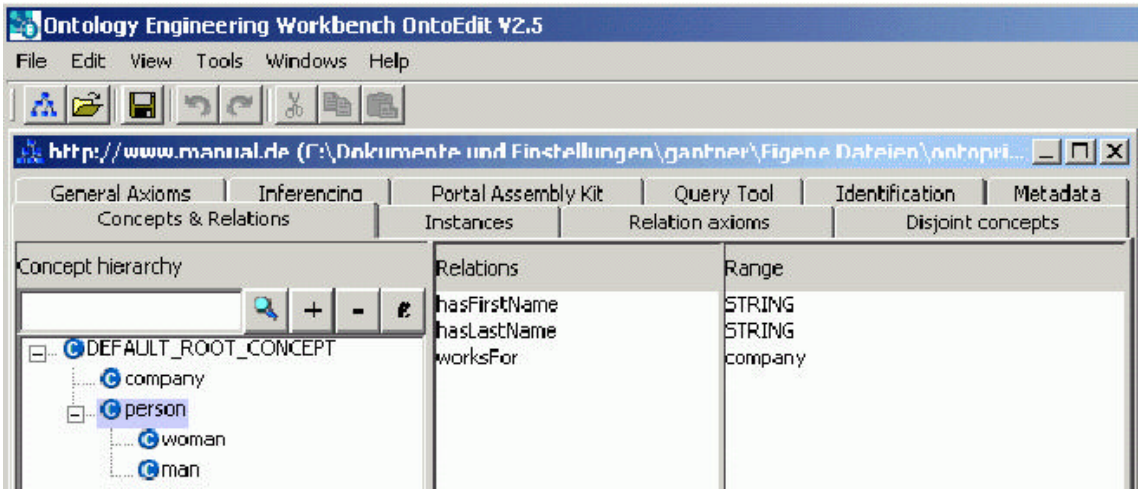


Fig. Ventana donde se muestran la disposición jerárquica en forma de árbol de los conceptos representados en la Ontología.

Instances

Utilizando el plug-in de instancias uno puede definir, editar y eliminar instancias. Una instancia se puede crear seleccionando un concepto, por ejemplo persona y creando una instancia, en ese momento la instancia que se cree será una instancia de persona, y por tanto heredará todos los atributos que esta contenga. Un ejemplo de cómo se visualiza lo vemos en la siguiente ventana de OntoEdit.

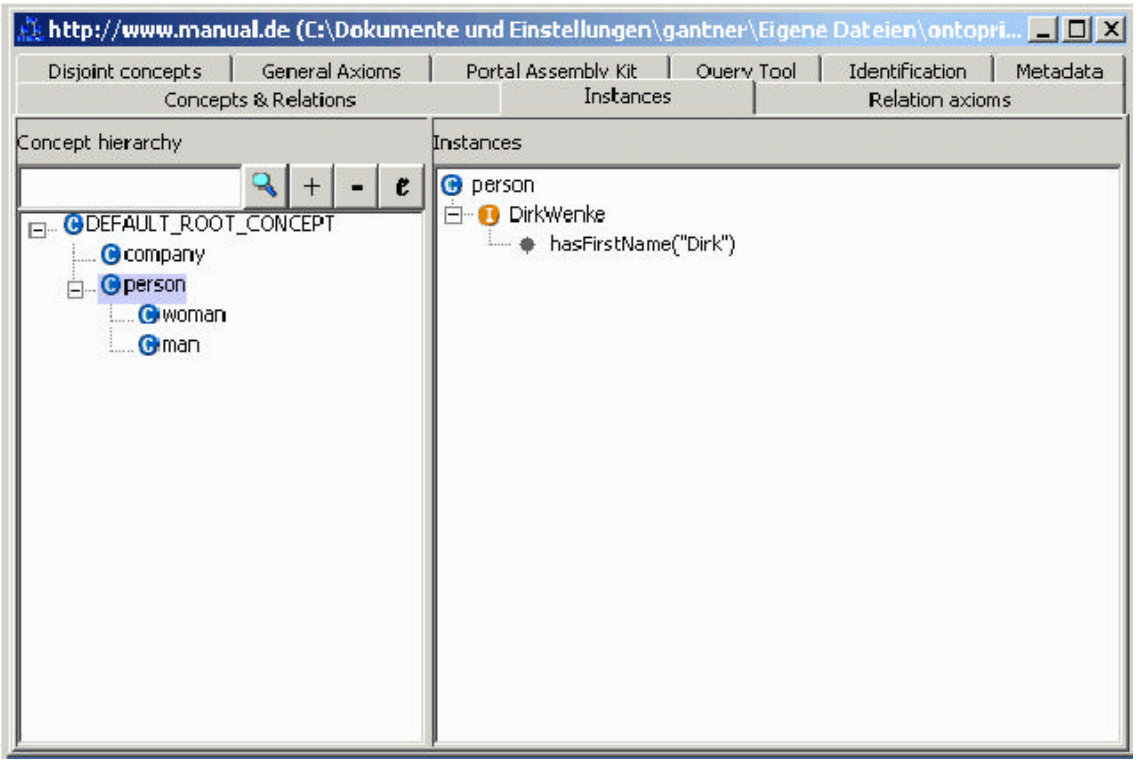


Fig. Ventana donde se aprecia la creación de instancias. En la parte izquierda aparece la Ontología representada en forma de árbol, y en la parte derecha aparece el resultado de seleccionar un concepto, donde aparecen las instancias del concepto seleccionado representadas por el icono 

Relation Axioms

Un axioma es una regla que se aplica sobre un dominio específico, los axiomas se utilizan para crear una serie de reglas sobre una Ontología que se han de cumplir para que la Ontología sea consistente.

OntoEdit tiene un editor textual de axiomas así como también dispone de un editor gráfico, en los cuales puedes definir unas reglas específicas sobre tu Ontología.

El plugin Relation axiom, contiene ya una serie de axiomas predefinidos, que se pueden aplicar a lo largo y ancho de la Ontología, estos son:

- Symmetric axioms
- Transitive axioms
- Inverse axioms

Un usuario, puede definir los conceptos de su Ontología y las relaciones entre estos y aplicar por ejemplo un tipo de axioma sobre una relación. Imaginemos que tenemos el concepto Persona y el concepto Empresa, y tenemos la relación una Persona **trabaja** en una Empresa, si yo a esta relación le aplico el Inverse axiom entonces automáticamente aparece otro tipo de relación en sentido inverso que se llama emplear, Una Empresa **emplea** a una Persona. Parece una cosa sin mucha transcendencia, pero si utilizamos un plug-in de consultas a la Ontología y preguntamos si en una empresa trabaja tal persona si no hubiésemos habilitado para la relación **trabaja** el Inverse relation, no hubiésemos obtenido ningún resultado, en cambio habilitándolo se crea esa nueva relación obteniendo así el nombre de las personas que trabajan en la empresa.

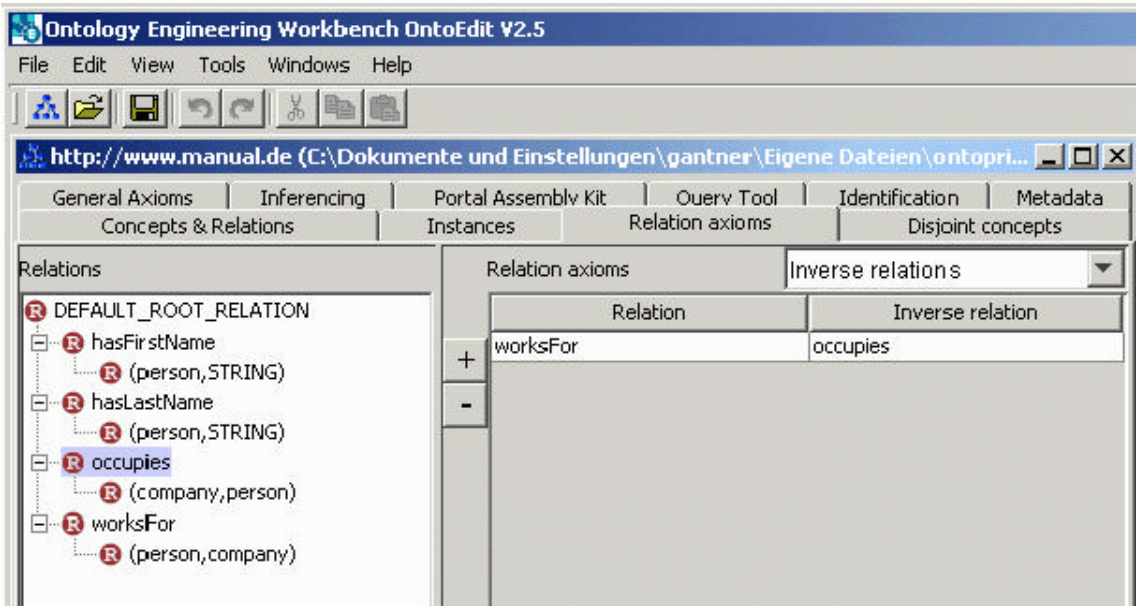


Fig. ventana donde se muestra la relación WorksFor y la nueva relación Occupies resultado de aplicar el relation axiom plug-in Inverse axioms.

Disjoined Concepts

En este plug-in el usuario puede definir conceptos del tipo disjoint. Este plug-in no es realmente un axioma pero puede ser usado para chequear la consistencia de una Ontología. Si tu defines dos conceptos del tipo disjoint, no se podrán crear instancias que pertenezcan a la vez a dos conceptos diferentes. Actualmente OntoEdit dispone en cada una de sus versiones de este plug-in con lo cual una instancia no puede instanciar a dos conceptos a la vez.

Identification

En el plug-in de Identification, se especifica meta información sobre la Ontología.
Aquí es donde se puede dar un título a tu Ontología, definir un dominio y una área de aplicación concreta. La **Uri** también se muestra aquí, pero no puede ser editada, la Uri (**U**niform **R**essource **I**dentifier), es un identificador de tu Ontología y sirve para que otros usuarios tengan acceso a ella y puedan consultarla o trabajar sobre ella. Con este

plug-in también puedes definir o especificar Ontologías que tengan relación con la Ontología creada, ya sean Ontologías que modelen un mismo dominio o área o Ontologías que has utilizado para crear tu Ontología.

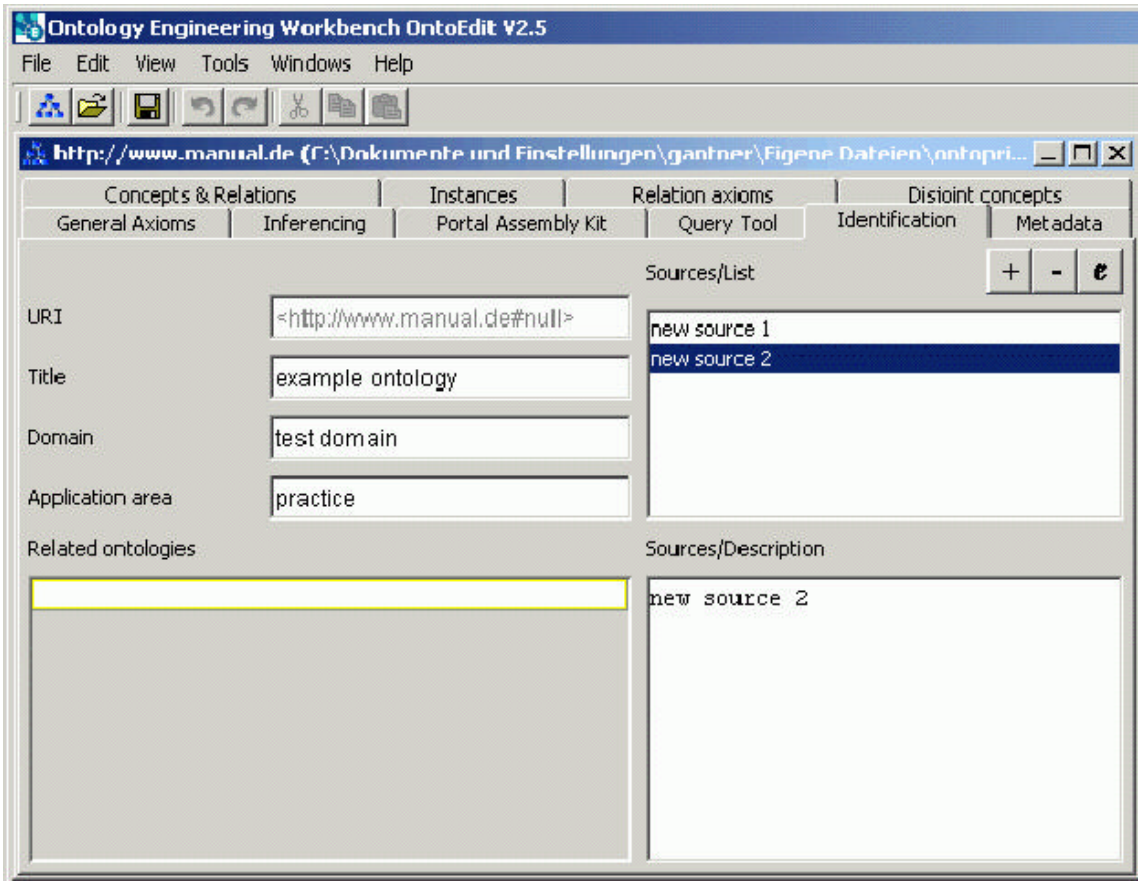


Fig. Ventana donde se realizan las funciones anteriormente citadas.

Metadata

En este plug-in se puede ver información estadística sobre tu Ontología, y especificar, quién es el desarrollador de la Ontología así como crear la documentación en cualquier lenguaje. Las estadísticas reflejan el número de conceptos, relaciones, instancias y axiomas que contiene tú Ontología, la profundidad del árbol ontológico, y la fecha de creación así como la de última modificación.

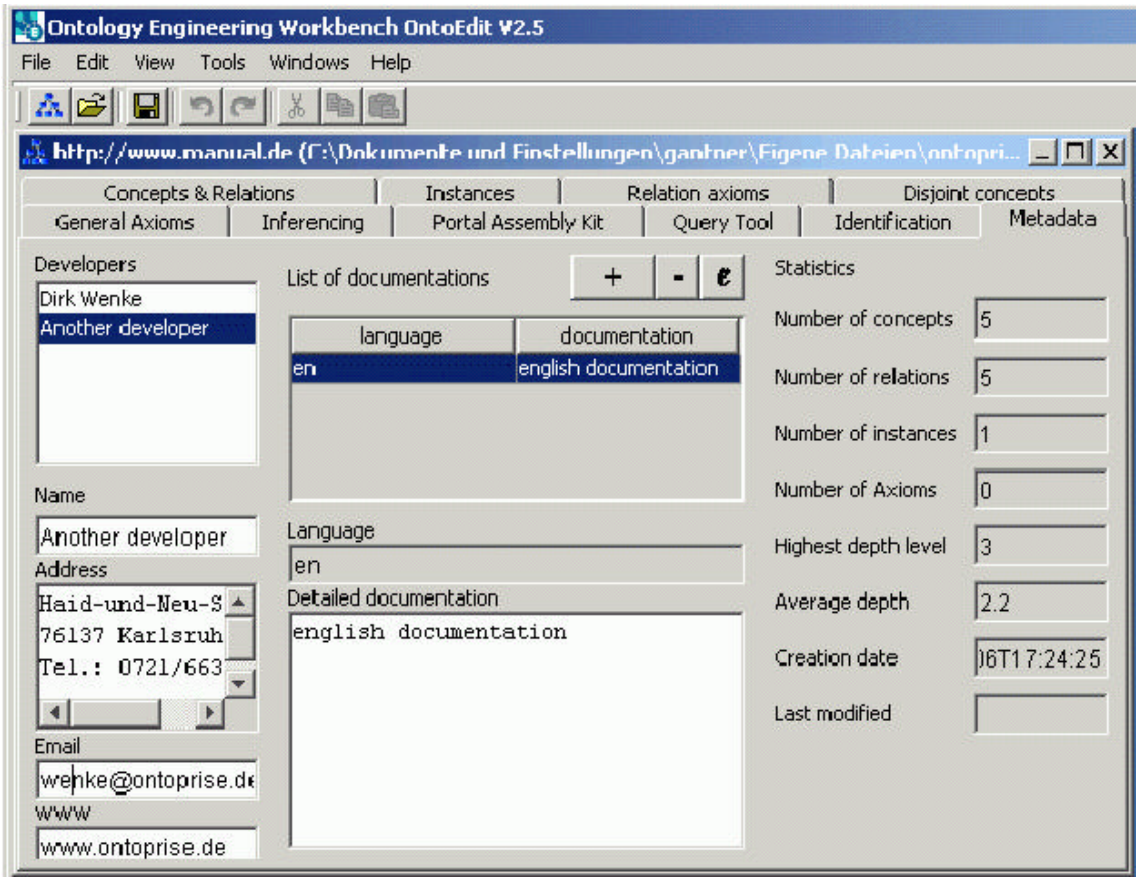


Fig. Ventana donde se reflejan los datos estadísticos de la Ontología modelada.

Plug-ins para un modelado avanzado

Hemos visto una serie de plug-ins para modelar Ontologías de forma eficiente, estos plug-ins contienen las funciones básicas que todo editor de Ontologías debe tener para crear tu propia Ontología y poder manejar la información que contiene. Los plug-ins que se describen a continuación contienen funciones avanzadas sobre Ontologías, que un usuario experto puede utilizar para analizar esa información más fácilmente.

Visualizador

El plug-in visualizador permite al usuario editar su Ontología gráficamente.

Los conceptos y las instancias se muestran como pelotas azules y verdes respectivamente. El usuario puede navegar gráficamente para comprobar que la Ontología mantiene la estructura que inicialmente quería representar, y puede ver las instancias de cada concepto seleccionando el concepto, y aparecerán el número de instancias que tiene el concepto seleccionado representado con bolitas rojas.

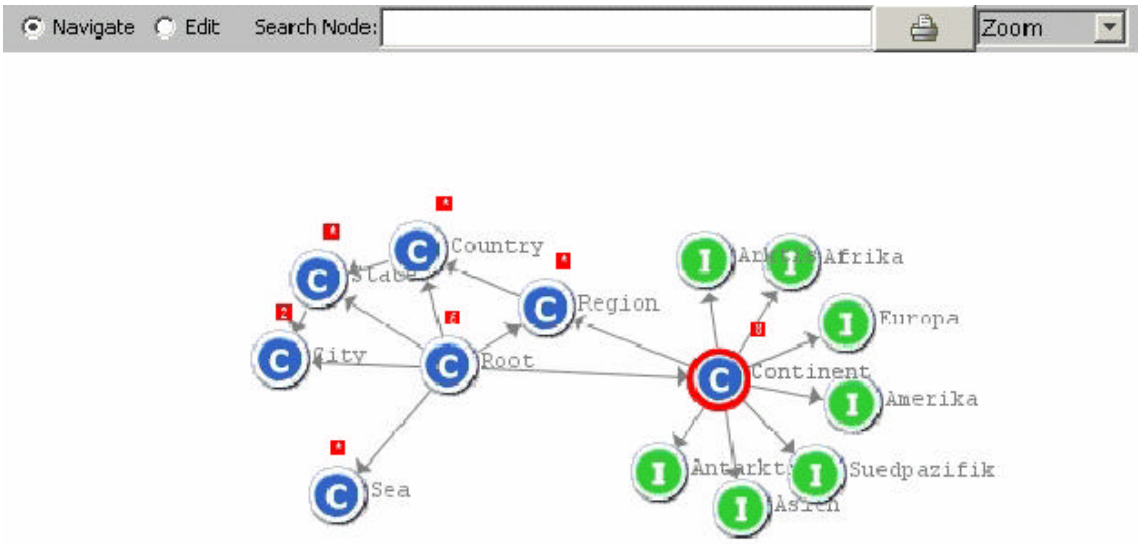


Fig. Representación el árbol ontológico gráficamente

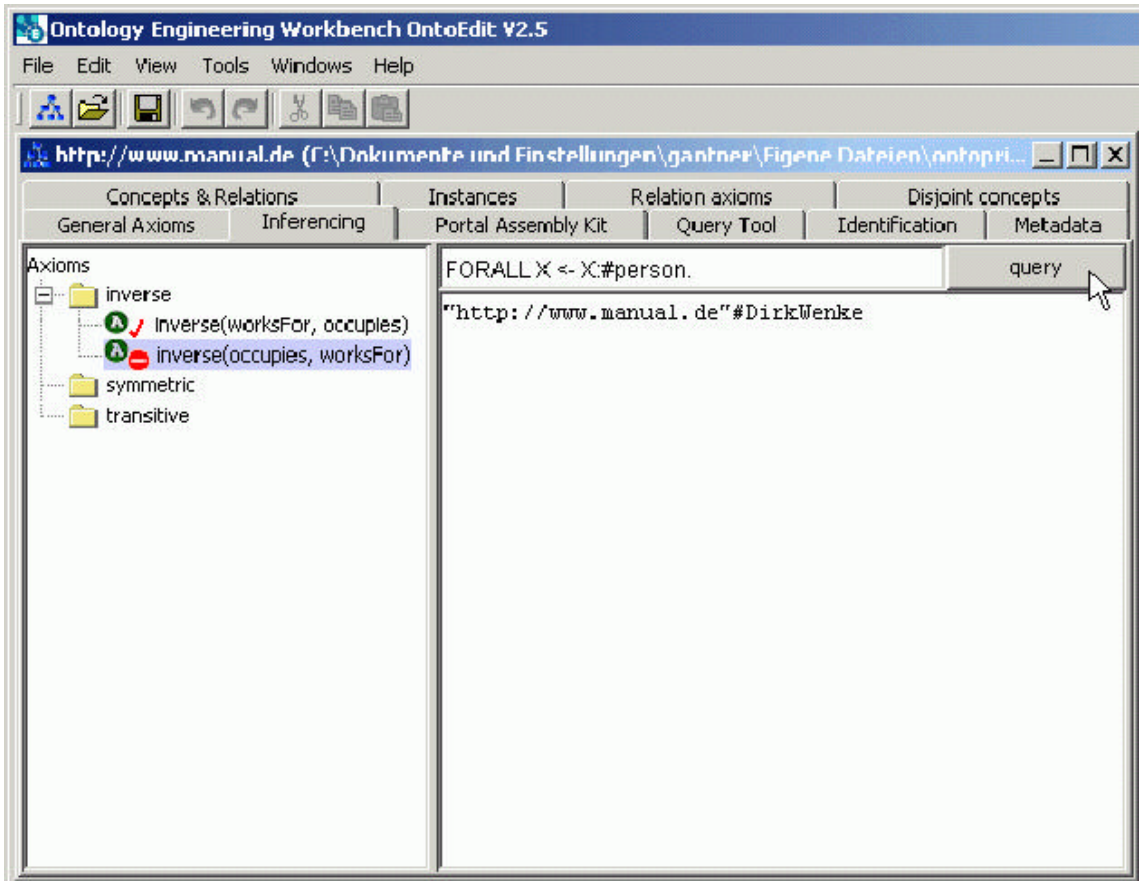
Query Tool

Este plug-in se utiliza para consultar la base de conocimiento que representa la Ontología. Con esta herramienta el usuario puede formular consultas sobre conceptos o relaciones representadas en la Ontología.

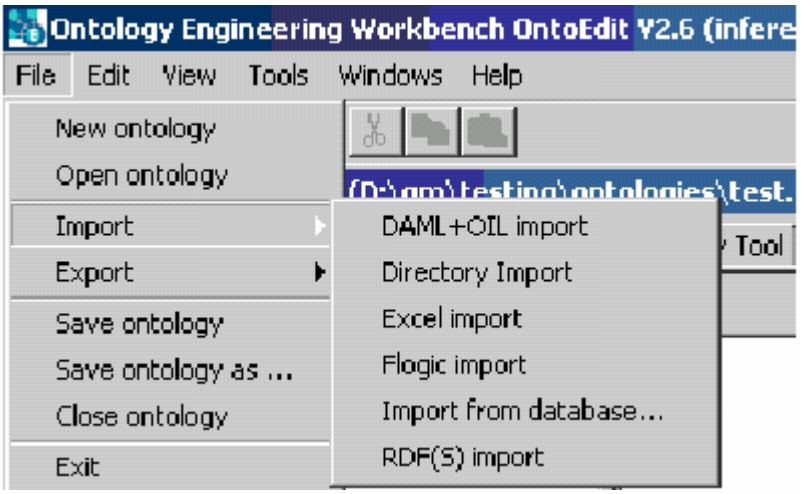
Inferencing

Este plug-in puede ser usado para testar una Ontología y sus axiomas. Tú puedes formular consultas de entre los datos modelados, la sintaxis de estas consultas viene dada por el lenguaje F-Logic.

Otra funcionalidad es la posibilidad de habilitar o deshabilitar axiomas, que se utiliza para testera el correcto funcionamiento tanto de los **Relations Axioms** (axiomas que OntoEdit integra), como **General Axioms** (axiomas definidos por el usuario).



Data Import/Export



OntoEdit soporta los formatos RDF(S), DAML+OIL,OXML y FrameLogic(F-Logic). También puede importar de una plantilla Excel, y en la versión de OntoEdit professional

también se puede importar y exportar a una base de datos, entre las que soporta OntoEdit está: **MSSQL Server, Oracle, DB2, MySQL.**

En mi proyecto, después de modelar la Ontología con OntoEdit la exporté a RDF, ya que los agentes de Internet encargados de buscar las páginas Webs leían las Ontologías en formato RDF. El documento que hay a continuación es el resultado de exportar la Ontología **"Biosensores"** en el formato RDF, en el cual se ve como las relaciones de parentesco padre –hijo **"is-a" relations** se mantienen, haciendo posible que la estructura principal, que es la clasificación y estructuración de los conceptos quede inalterable, de esta forma el modelo conceptual representado mantiene toda la información relevante necesaria para poder utilizarla como soporte de conocimiento.

A continuación les mostramos un extracto de la Ontología en formato RDF

Ontología exportada a formato RDF: Dominio Biosensores

```
<RDF:RDF
  xmlns:RDF='http://www.w3.org/1999/02/22-RDF-syntax-ns#'
  xmlns:NS0='http://www.newOnto.org/1034673349140'
  xmlns:NS1='http://schema.ontoprise.com/oxml/RDF/1.0'
  xmlns:RDFs='http://www.w3.org/2000/01/RDF-schema#'>
  <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#Biosensor'>
    <RDF:type RDF:resource='http://www.w3.org/2000/01/RDF-schema#Class'/>
    <RDFs:label xml:lang='en'>http://www.newOnto.org/1034673349140#Biosensor</RDFs:label>
    <RDFs:subClassOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_CONCEPT'/>
  </RDF:Description>
  <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#measurement'>
    <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
    <RDFs:subPropertyOf RDF:resource='http://www.newOnto.org/1034673349140#measurement'/>
    <RDFs:domain
RDF:resource='http://www.newOnto.org/1034673349140#Biological_recognition_element'/>
    <RDFs:range RDF:resource='http://www.w3.org/2001/XMLSchema#STRING'/>
  </RDF:Description>
  <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#support_surface'>
    <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
    <RDFs:subPropertyOf RDF:resource='http://www.newOnto.org/1034673349140#support_surface'/>
    <RDFs:domain RDF:resource='http://www.newOnto.org/1034673349140#Covalent_bonding'/>
    <RDFs:range RDF:resource='http://www.w3.org/2001/XMLSchema#STRING'/>
  </RDF:Description>
  <RDF:Description
RDF:about='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_RELATION'>
    <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
```

```
 </RDF:Description>
 <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#Air_analysis'>
  <RDF:type RDF:resource='http://www.w3.org/2000/01/RDF-schema#Class'/>
  <RDFs:label xml:lang='en'>http://www.newOnto.org/1034673349140#Air_analysis</RDFs:label>
  <RDFs:subClassOf RDF:resource='http://www.newOnto.org/1034673349140#Environment'/>
 </RDF:Description>
 <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#transducer'>
  <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
  <RDFs:subPropertyOf RDF:resource='http://www.newOnto.org/1034673349140#transducer'/>
  <RDFs:domain RDF:resource='http://www.newOnto.org/1034673349140#Additive'/>
  <RDFs:range RDF:resource='http://www.w3.org/2001/XMLSchema#STRING'/>
 </RDF:Description>
 <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#polymer_type'>
  <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
  <RDFs:label xml:lang='en'>http://www.newOnto.org/1034673349140#polymer_type</RDFs:label>
  <RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_RELATION'/>
  <RDFs:domain RDF:resource='http://www.newOnto.org/1034673349140#Electropolymerization'/>
  <RDFs:range RDF:resource='http://www.w3.org/2001/XMLSchema#STRING'/>
 </RDF:Description>
 <RDF:Description
RDF:about='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_CONCEPT'>
  <RDF:type RDF:resource='http://www.w3.org/2000/01/RDF-schema#Class'/>
 </RDF:Description>
 <RDF:Description RDF:about='http://www.newOnto.org/1034673349140#physical_phenomen'>
  <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
  <RDFs:label
xml:lang='en'>http://www.newOnto.org/1034673349140#physical_phenomen</RDFs:label>
  <RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_RELATION'/>
 </RDF:Description>
```

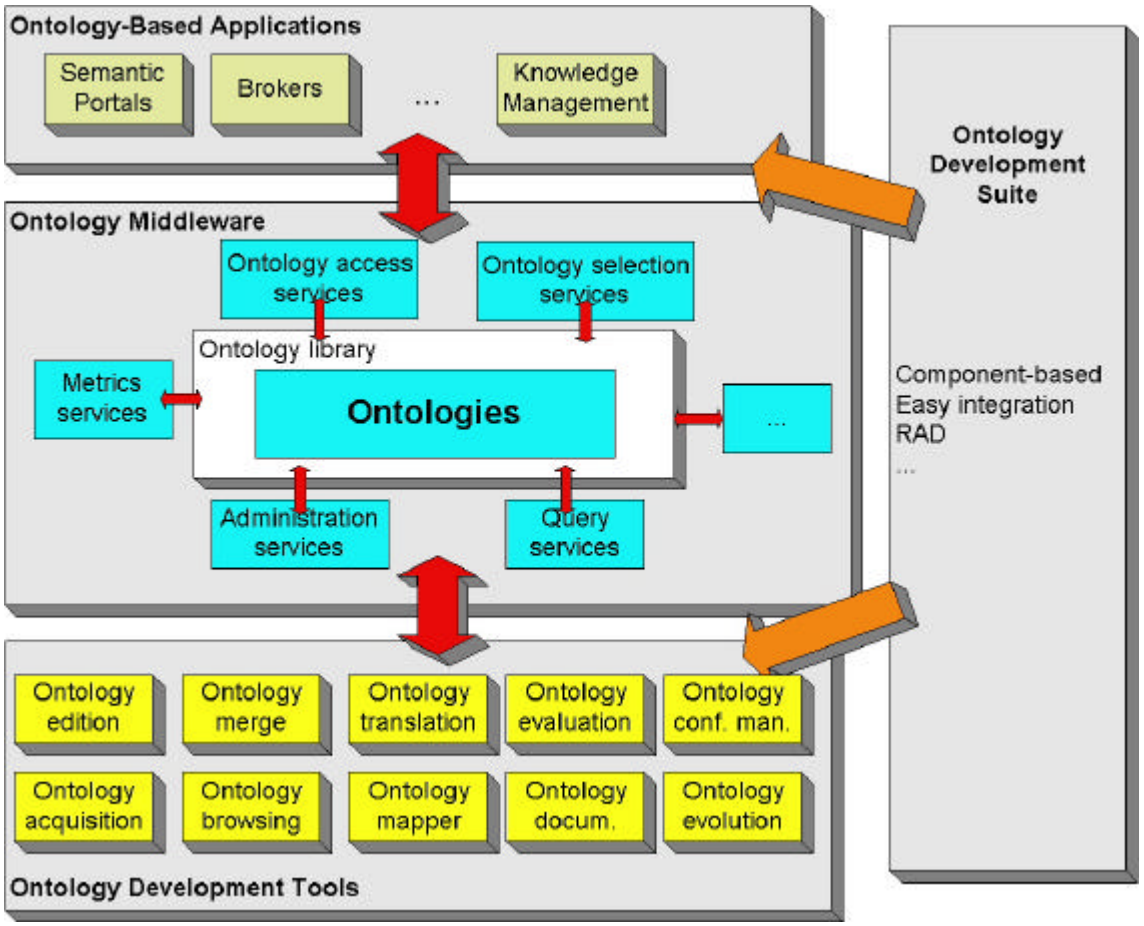
WebODE

WebODE, es la otra herramienta con la que he trabajado en el modelaje de Ontologías, las funciones que desempeña son similares a las de OntoEdit.

WebODE es el sucesor de ODE (Ontology Design Enviorement) , y fue desarrollado en el laboratorio de inteligencia artificial de la Universidad Politécnica de Madrid (UPM). Es al igual que OntoEdit una herramienta para modelar conocimiento a través de Ontologías. Esta herramienta utiliza para el desarrollo de sus Ontologías, la metodología METHONTOLOGY, una metodología que ya ha sido usada y testada

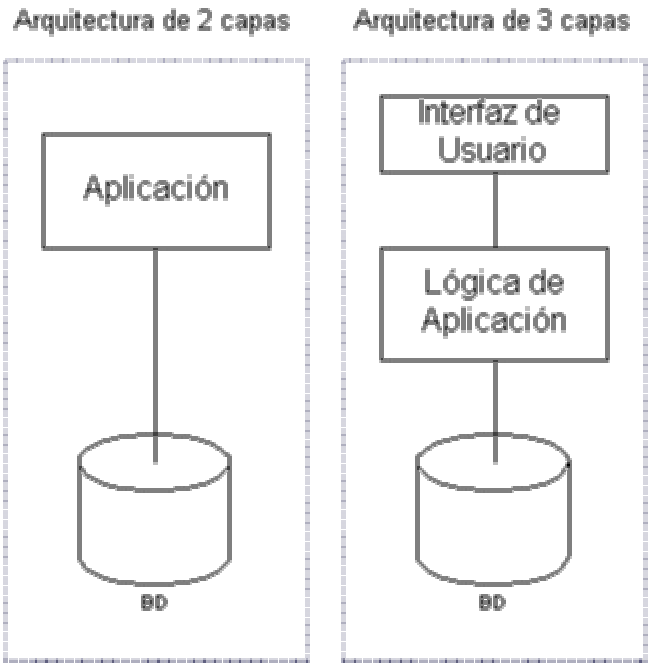
satisfactoriamente, y que fue construida en la Escuela técnica de ciencias de la computación en Madrid. WebODE es el equivalente a ODE (**O**ntology **D**esign **E**nvironment) con una interfaz web.

WebODE se ha implementado de acuerdo con las tecnologías más actuales y mas estandarizadas como **Java, RMI CORBA o XML,** esta implementación supone una máxima flexibilidad y interoperabilidad con otras aplicaciones necesarias en la empresas.



Arquitectura

WebODE has sido construido utilizando una arquitectura **3-tier,** a este tipo de arquitectura se le llama arquitectura de 3-capas, que completa los defectos que presentaban las arquitecturas de 2 capas.



La arquitectura de 2 capas tenía los siguientes defectos:

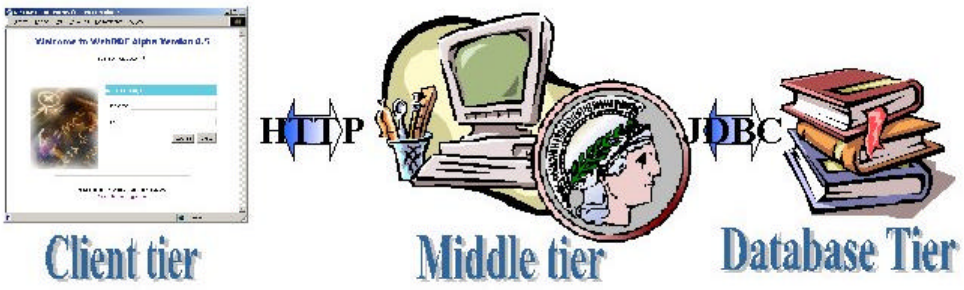
- Mucha carga en el cliente
- Poca carga en el servidor
- Mucho tráfico en la red
- Mantenimiento costoso en cada cliente
- Posibilidad de clientes desfasados

Características principales de una arquitectura de 3 capas:

Con una arquitectura de 3 capas todos estos inconvenientes que hemos visto tenían las arquitecturas de 2 capas desaparecen, gracias a la inserción de esta tercera capa. Y aparece una mejora fundamental:

- Posibilidad de crear diferentes interfaces para la misma lógica de negocio

WebODE, como ya hemos dicho, incorpora este tipo de arquitecturas, en la imagen siguiente veremos como se distribuyen las tres capas:

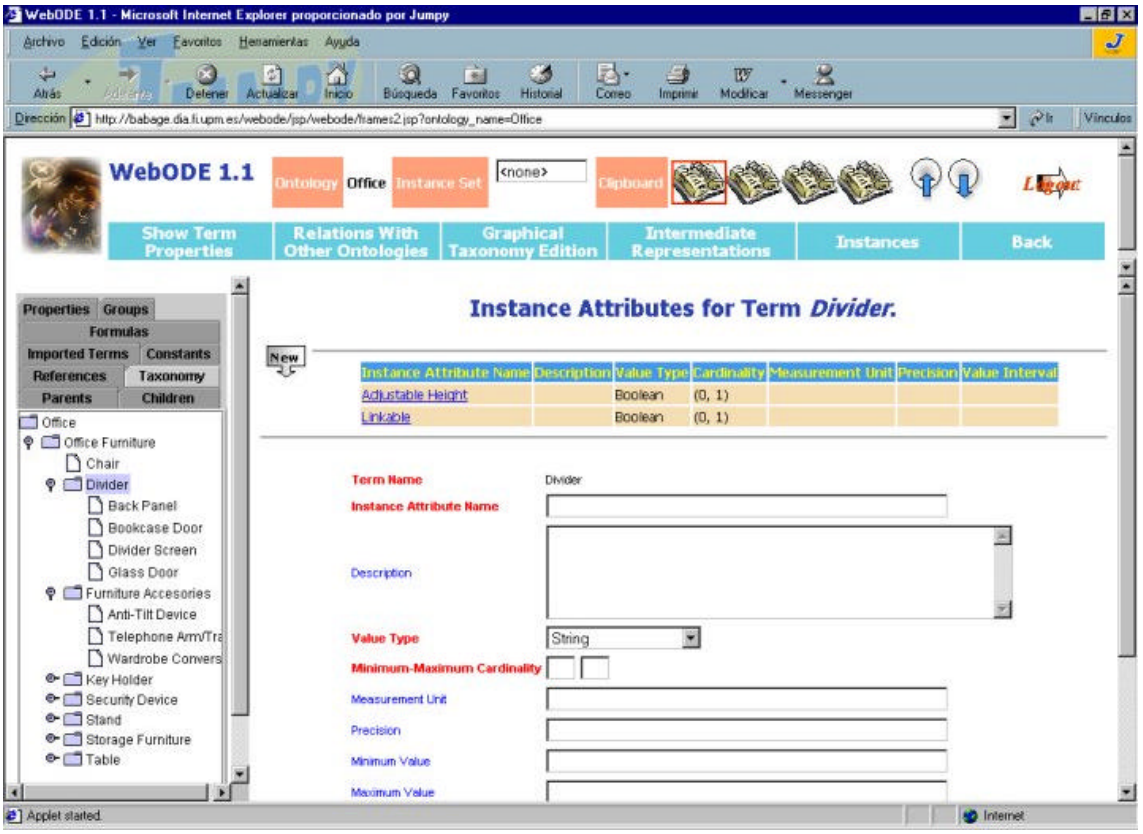


La primera capa es la interfaz de usuario. Para este capa WebODE ha utilizado tecnologías webs estándares, la capa de presentación fue implementada usando HTML, CSS (Cascading Style Sheets) y XML (Extended Mark-up Language) para permitir y facilitar la interoperabilidad con otras aplicaciones. Así el cliente puede trabajar mas rápidamente y facilita al servidor la faena de validar usuarios, este tipo de métodos utilizan tecnologías como JavaScript y Java.

La segunda capa es la lógica del negocio. Aunque de hecho esta capa esta constituida por otras 2 subcapas: Capa de presentación y Capa lógica.

La capa lógica hace posible el acceso a las Ontologías por medio de una API , que es invocada por una aplicación servidor desarrollada por el mismo equipo de WebODE (Minerva Aplicattion Server). Este servidor permite habilitar el acceso a servicios a través del RMI-IIOP (Remote Method Invocation-Internet Inter ORB Protocol), y además hace que el desarrollo y la integración de la aplicación resulten más sencillos.

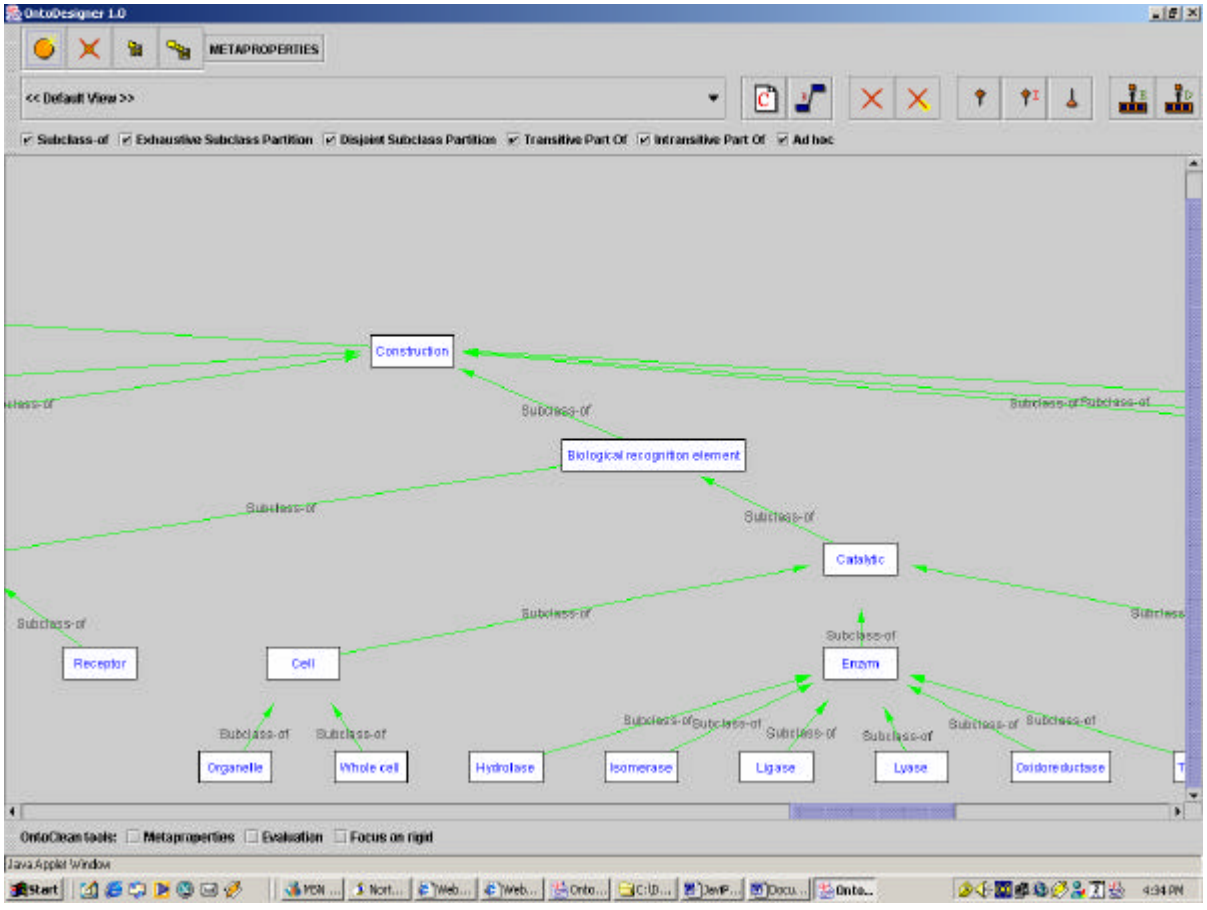
La capa de presentación es la responsable de generar el contenido que es presentado al usuario, también se encarga de manejar las peticiones que le llegan desde el cliente. Estas funciones las desempeña utilizando tecnologías estándares como son Servlets o JSP's (Java Server Pages).



Por último, la tercera capa es la que abarca el contenido de la información que esta disponible en las Ontologías, WebODE para almacenar esta información utiliza una basa de datos relacional, actualmente únicamente utiliza **ORACLE** para almacenar toda la información, y el acceso a esta base de datos se hace por medio de los JDBC (Java Database Connectivity) estándares, pero en el futuro es posible la incorporación de más bases de datos como OntoEdit para almacenar toda la información que contienen las Ontologías.

Características principales

- Soporte de multiusuarios con una misma Ontología.
- Capacidad de adaptar las funcionalidades a gusto del usuario experto por medio de “Templates”.
- Portapapeles de gran extensión y calidad para facilitar el manejo de información (copy-paste).
- Chequeo completo de la Ontología para asegurar que la Ontología mantiene una cierta consistencia y un conocimiento válido.
- Edición sencilla de la taxonomía, usando la interfaz de usuario, o utilizando el potente editor gráfico (OntoDesigner).



- Manejo de instancias independiente de la conceptualización de la Ontología.
- API para tener acceso a las Ontologías desde cualquier aplicación utilizando RMI o CORBA.
- Máxima operabilidad con cualquier otra aplicación gracias al uso de estándares actuales como XML.

Características de la Edición de Ontologías

- Uso de una interfaz de usuario gráfica para crear tus Ontologías.
- Administrador definido de vistas de una misma Ontología.
- Chequeo de la consistencia.
    - Restricciones de tipo.
    - Restricciones de valores numéricos.
    - Restricciones de cardinalidad.
    - Verificación de la consistencia de la Ontología (Instancias comunes de clases del tipo **disjoint**).
- Motor de inferencia.
- Construcción de axiomas.
- Servicio de documentación.
- Podas gráficas de las relaciones entre conceptos.

Importación Exportación de Ontologías

WebODE dispone de unas funciones para importar y exportar Ontologías en actuales y estandarizados formatos, para poder interoperar con demás aplicaciones.

Lenguajes de importación

- XML.
- X-CARIN.

Lenguajes de exportación

- XML.
- RDF(S).
- OIL.
- X-CARIN.
- DAML+OIL.
- PROLOG.
- JESS.

En el proyecto hemos escogido exportar la Ontología en el lenguaje RDF, porque ya hemos comentado que los agentes de Internet utilizaban este lenguaje para procesar la Ontología.
El código RDF que exporta WebODE y OntoEdit es similar, la cual cosa hace posible que cualquiera de los dos documentos RDF's sea válido para utilizarlo como input en el sistema multiagentes.

Para utilizar la Ontología exportada en RDF con OntoEdit, no se ha de hacer ningún cambio en el código, ya es válida para empezar a utilizarla. En cambio en la Ontología exportada en RDF por WebODE hay que hacer un pequeño cambio, y es que en OntoEdit la Ontología deriva de una clase raíz llamada **Root,** y el sistema multiagente dispone de una función de validación de la Ontología para analizar si ésta está o no bien formada para poder analizarla y extraer la información, que espera encontrar una Ontología colgando de esta clase Root. Por tanto solamente hemos de añadir 2 cosas:

1- **En la definición de la clase Biosensor hay que especificarle que ésta hereda de la clase root:**

```
<RDF:Description
RDF:about='http://webode.dia.fi.upm.es/ontology/Biosensor#B
iosensor'>
<RDF:type RDF:resource='http://www.w3.org/2000/01/RDF-
schema#Class'/>
<RDFs:subClassOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_
ROOT_CONCEPT'/>
 </RDF:Description>
```

2- **En todas las definiciones de las propiedades de una clase, hay que especificar que éstas heredan como propiedades de la clase DEFAULT_ROOT_RELATION:**

```
<RDF:Description
RDF:about='http://webode.dia.fi.upm.es/ontology/Biosensor#g
el_type'>
    <RDF:type RDF:resource='http://www.w3.org/1999/02/22-
RDF-syntax-ns#Property'/>
    <RDFs:domain
RDF:resource='http://webode.dia.fi.upm.es/ontology/Biosenso
r#Retention_by_gel'/>
<RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_
ROOT_RELATION'/>
    <RDFs:range
RDF:resource='http://www.w3.org/2000/01/RDF-
schema#Literal'/>
  </RDF:Description>
```

Conclusiones

Una vez utilizados los dos editores de Ontologías para modelar nuestra Ontología, podemos decir que prácticamente las funcionalidades que incorporan los dos editores son muy similares. En OntoEdit por ejemplo, las funciones se pueden ir acoplando según tus objetivos por medio de plug-ins, en cambio WebODE las incorpora en una versión estándar para todos los usuarios.

Una de las funciones que más he tenido que utilizar, aparte del modelaje de la Ontología, es la exportación de ésta en un lenguaje de programación en concreto. En mi caso el lenguaje utilizado como ya hemos visto ha sido RDF. El sistema multiagentes diseñado en el H-techSight, utiliza el formato RDF de OntoEdit por lo que si uno quisiera utilizar WebODE para modelar su Ontología debería hacer las transformaciones que hemos visto anteriormente en su documento RDF.

Pero esto no es todo, en la última parte del proyecto, que veremos a continuación, he utilizado Sésame, una aplicación standalone, que se comunica a través de un servidor (Apache) a una base de datos (mySQL). Esta aplicación permite guardar Ontologías codificadas en RDF en la base de datos utilizándola así como repositorio. El problema reside en que cualquier aplicación que trabaje con RDF, más concretamente que tenga que formar un RDF lo hará de una forma determinada, ya que si bien todos son documentos RDF, no todos ellos están diseñados de la misma manera. De este modo, no todos los RDF's que había creado con OntoEdit y WebODE son válidos para Sésame.

Los RDF's exportados en OntoEdit, son válidos para la interfície de usuario, ya que ésta ha sido desarrollada teniendo en cuenta el formato de OntoEdit, pero no son validos en un cierto sentido que ahora explicaré para Sésame.

El cierto sentido, es que si tu en OntoEdit modelas una Ontología y dos clases diferentes contienen un Slot con el mismo nombre, OntoEdit a la hora de crear el RDF, cogerá ese atributo y extenderá su nombre añadiéndole la clase a la que pertenece, para diferenciarlos por decirlo de alguna forma. De esta forma en OntoEdit la definición por

ejemplo del atributo ***measurement*** , para la clase Application y para la clase Bilogical_recognition_element seria:

Ej.

```
<RDF:Description
RDF:about='http://www.newOnto.org/1034673349140#measurement___Application'
>
   <RDF:type                RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-
ns#Property'/>
   <RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#measurement'/>
   <NS1:is_local_relation_of
RDF:resource='http://www.newOnto.org/1034673349140#measurement'/>
   <RDFs:domain
RDF:resource='http://www.newOnto.org/1034673349140#Application'/>
   <RDFs:range RDF:resource='http://www.w3.org/2001/XMLSchema#STRING'/>
  </RDF:Description>

<RDF:Description
RDF:about='http://www.newOnto.org/1034673349140#measurement___Biological_re
cognition_element'>
   <RDF:type                RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-
ns#Property'/>
   <RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#measurement'/>
   <NS1:is_local_relation_of
RDF:resource='http://www.newOnto.org/1034673349140#measurement'/>
   <RDFs:domain
RDF:resource='http://www.newOnto.org/1034673349140#Biological_recognition_elem
ent'/>
   <RDFs:range RDF:resource='http://www.w3.org/2001/XMLSchema#STRING'/>
  </RDF:Description>
```

Se aprecia que donde se define el atributo **measurement** se le extiende la clase a la que pertenece este atributo.

Por tanto utilizando una Ontología exportada en OntoEdit tendrías que modificar el RDF, eliminando la extensión del atributo, y la definición que está también en negrita, que te indica que es un atributo local de esa clase, o sino intentar directamente con el editor evitar que dos clases diferentes contengan dos Slots con el mismo nombre.

Exportando una Ontología en RDF con WebODE pasa algo similar. A parte de los cambios que hemos explicado que se han de hacer con una Ontología en WebODE para utilizarla en nuestro sistema multiagentes, habría que hacer otros:

Primero hay que decir que el problema que hemos explicado antes, que tenía OntoEdit con los slots con el mismo nombre en clases diferentes, WebODE los trata de otra forma y es que, un atributo normal se define de la siguiente manera:

Ej1.

```
<RDF:Description
RDF:about='http://webode.dia.fi.upm.es/ontology/Biosensor#response_time'>
<RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-ns#Property'/>
   <RDFs:domain
RDF:resource='http://webode.dia.fi.upm.es/ontology/Biosensor#Product'/>
<RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_RELAT
ION'/>
   <RDFs:range RDF:resource='http://www.w3.org/2000/01/RDF-schema#Literal'/>
  </RDF:Description>
```

En este caso el atributo response_time pertenece a la case Product, como se ve en la definición del dominio (RDFs:domain), en cambio si existe un slot que tiene el mismo nombre en dos clases diferentes WebODE lo definiría de la siguiente manera:

Ej2.

```
<RDF:Description
RDF:about='http://webode.dia.fi.upm.es/ontology/Biosensor#measurement'>
   <RDF:type RDF:resource='http://www.w3.org/1999/02/22-RDF-syntax-
ns#Property'/>
<RDFs:subPropertyOf
RDF:resource='http://www.newOnto.org/1034673349140#DEFAULT_ROOT_RELAT
ION'/>
  </RDF:Description>
```

Y de esta forma no podríamos saber a que clases pertenece este atributo, entonces la única solución que se podría hacer es cambiar el RDF manualmente y añadir dos definiciones expresando los dominios a los que pertenecen los slots, o como también

hemos dicho antes evitar crear dos slots con el mismo nombre que pertenezcan a clases diferentes.

Por tanto un documento RDF exportado con WebODE seria válido para la aplicación Sésame pero sin estos cambios que hemos mencionado, no podríamos utilizarlo para nuestro sistema multiagentes.

En conclusión hay que decir que la interoperabilidad de las aplicaciones que trabajan con Ontologías no es mala, pero siempre hay pequeños detalles que se han de modificar para que sea totalmente portable una Ontología de un entorno a otro.

Por tanto no nos podemos guiar en eso para evaluar un editor u otro porque como hemos visto utilizando cualquiera de los dos tendremos que hacer rectificaciones en el fichero RDF generado.

Evaluando los dos editores yo me quedaría con OntoEdit, por varias razones, una es que puedes ir adquiriendo funciones por medio de plug-ins conforme vayas avanzando en el campo del diseño de Ontologías, puedes utilizarlo como aplicación standalone sin necesidad de estar conectado a ningún servidor y pudiendo trabajar desde casa, y porque si quisieras compartir tu Ontología para que varios usuarios a la vez trabajasen con ellas, también podrías ya que OntoEdit (OntoEdit Profesional) dispone de un servidor donde se pueden guardar la Ontologías y conectándose a este servidor podrías hacer las mismas funciones que con el programa en casa, y por último tiene prácticamente las mismas funcionalidades que el WebODE, eso sí para el OntoEdit y el OntoEdit Professional no hay versiones libres.

Para diseñar la Ontología de Biosensores he utilizado finalmente el OntoEdit, ya que el diseño del sistema multiagentes estaba implementado para leer Ontologías codificadas en RDF en el formato de OntoEdit.

4.4. Análisis del razonamiento sobre la Ontología

Ya hemos comentado que la creación de una Ontología no es un proceso trivial, ya que la búsqueda de páginas web que realizan los agentes de Internet (Ainternet) depende directamente de cómo estén representados los conceptos en la Ontología y la relación de que haya entre estos conceptos.

Una vez que la Ontología esta a priori correctamente modelada, se distribuye a los agentes para que estos empiecen la búsqueda. Las páginas web encontradas por los agentes se insertan en la Ontología por lo que cada clase contendrá las páginas webs relacionadas con ésta. Pero una vez la Ontología contiene ese tipo de información, se ha de razonar con ella, ya que aplicando un cierto razonamiento uno puede encontrar relaciones entre conceptos de la propia Ontología que no había tenido en cuenta, y de esa manera se puede perfeccionarla o incluso extenderla. Sobre nuestra Ontología de biosensores he aplicado dos razonamientos y un filtrado de páginas.

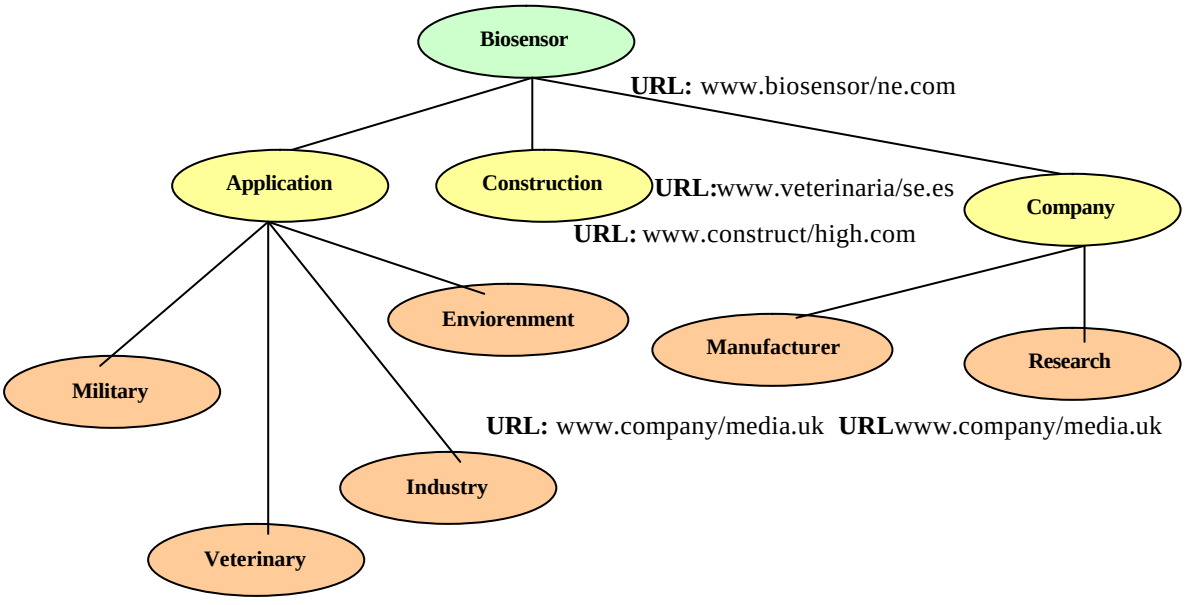
El filtrado de páginas no es más que recorrer la Ontología con las páginas webs insertadas y examinar que una misma clase no contenga dos URL's iguales, ya que eso significa duplicidad de información que no interesa. Si encontramos una clase que contenga dos o más URL's iguales lo que hacemos es eliminar todas ellas menos una ya que las demás son innecesarias.
"Teóricamente", los agentes de Internet deberían encargarse de no duplicar información innecesaria en la Ontología, pero después de varias búsquedas (eso si con poca frecuencia), he encontrado URL's iguales en una misma clase, y esto afecta directamente a los razonamientos que explico a continuación porque estos algoritmos no están diseñados para contemplar esta situación, por tanto curándome en salud he diseñado otro algoritmo que contemple éste caso y que se aplique antes de aplicar los 2 razonamientos que explico a continuación:

Eliminación de links redundantes.

El primer razonamiento consiste en encontrar links iguales entre diferentes conceptos donde haya una relación de "hermanos" entre ellos. Si encontramos que varios nodos (conceptos) que tengan un mismo padre, contienen links iguales, lo que hacemos es eliminar ese link de cada uno de los hijos y incluírselo al padre, porque lo normal, es que si todos heredan del mismo concepto, este link tenga una relación directa con el padre, y por herencia también con los hijos. Ya que la búsqueda que hacen los agentes actualmente, depende del nombre de los conceptos (que están representados en clases) y de los atributos de estas clases (slots), por tanto si todos los hijos contienen un link igual es muy probable que sea por la combinación de atributos heredados del padre y atributos propios de hijo, que a lo mejor con solo los del padre no obtiene resultados y por eso éste no tiene el link, pero la relación que pudiera tener sería la misma.

Para aplicar este razonamiento se ha utilizado un algoritmo de recorrido de árboles binarios en post-orden, donde primero se visita el nodo izquierdo, después el derecho y más tarde el padre, claro que este algoritmo se ha modificado mínimamente para que sea válido para un árbol n-ario, como es el caso de nuestra Ontología.

Razonamiento-1



En este gráfico, se aprecia la Ontología con las URL's encontradas por los agentes.
En este razonamiento intentamos eliminar los links redundantes, y en este caso nos damos cuenta que **Manufacturer** y **Research** contienen una misma URL, entonces la forma de razonar seria la siguiente:

Eliminamos la Url de los hijos y se la añadimos al padre, y esto lo hacemos porque si varios hijos contienen una misma URL, se supone que la han encontrado por los atributos que comparten todos ellos, y esos atributos son heredados del padre, por lo que el padre también tendrá relación con esa URL, aunque por cualquier motivo los agentes no la han encontrado para la clase padre. Entonces si la URL la ponemos en el padre se supone que tiene relación con el padre y con los hijos ya que éstos heredan de él.

Relación entre links de clases diferentes

Este razonamiento consiste en encontrar links iguales entre clases que no tengan ningún tipo de parentesco, es muy útil porque de esta forma se pueden descubrir relaciones entre clases que a priori no se habían tenido en cuenta, y puede servir para retocar la Ontología añadiendo por ejemplo otra relación entre esas dos clases.

El requisito indispensable es que no tengan ningún tipo de parentesco, porque que un nodo contenga un link determinado, y alguno de sus hijos también lo contenga es del todo lógico porque el hijo hereda los atributos del padre, pero encontrar links entre clases diferentes es mucho más difícil porque los atributos teóricamente son diferentes por tanto si esto pasara se podría decir que estas clases tienen algo en común y se podría analizar que relación las une y extender la Ontología, obteniendo así búsquedas más eficientes.

Por parentesco entendemos:

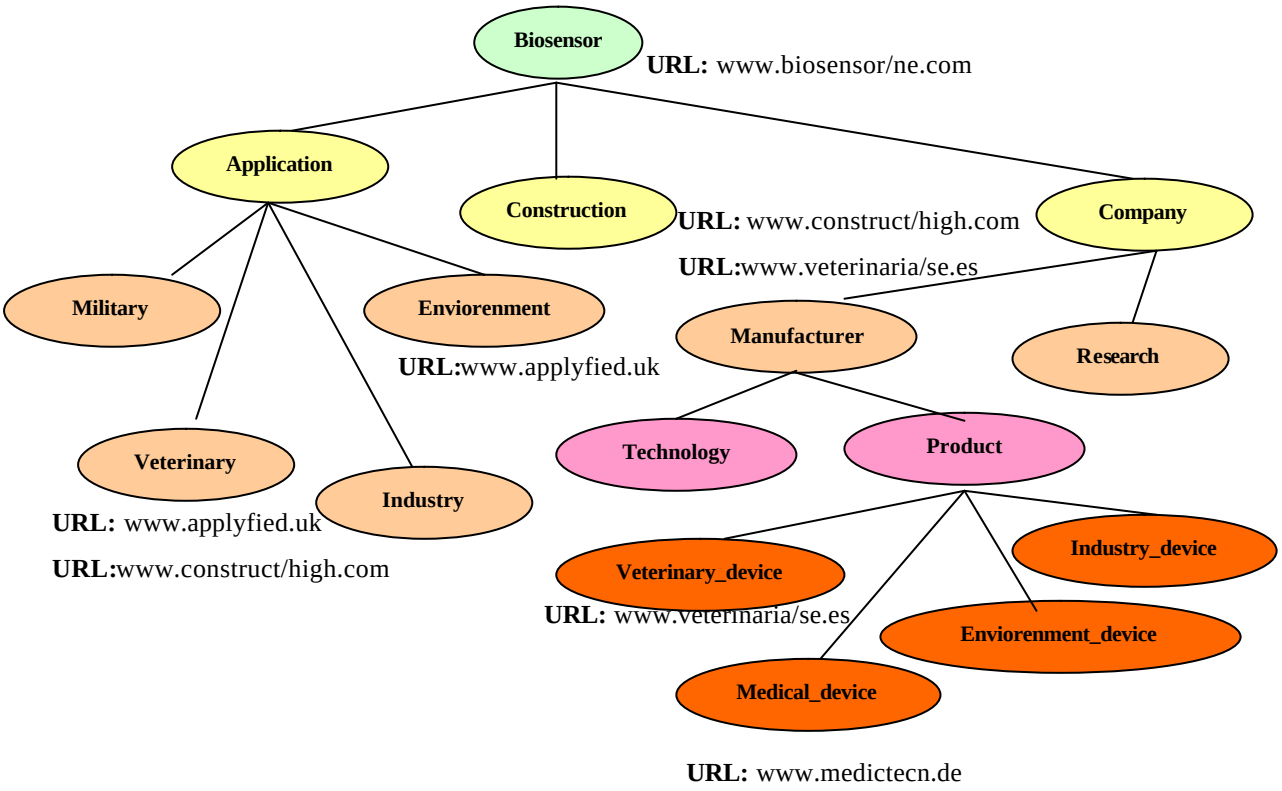
- Nodos que heredan atributos de otros nodos por un camino directo.

- Nodos con relación de hermanos.

Lo que se pretende evitar es que dos clases contengan la misma URL porque los atributos (slots) son iguales, ya que de esta forma a parte de ser del todo lógico no se encontraría ninguna relación nueva porque ya la hay.

Para este razonamiento se ha utilizado el mismo algoritmo de recorrido de árboles que en el anterior razonamiento, adptándolo mínimamente.

Razonamiento-2:



En este gráfico también representa la Ontología una vez se le han añadido las URL's encontradas por los agentes. Según la Ontología podemos observar las siguientes relaciones:

(Enviorenment,Veterinary: www.applyfied.uk).

(Construction,Veterinary_device: www.veterinaria/se.es).

(Veterinary,Construction: www.construct/high.com).

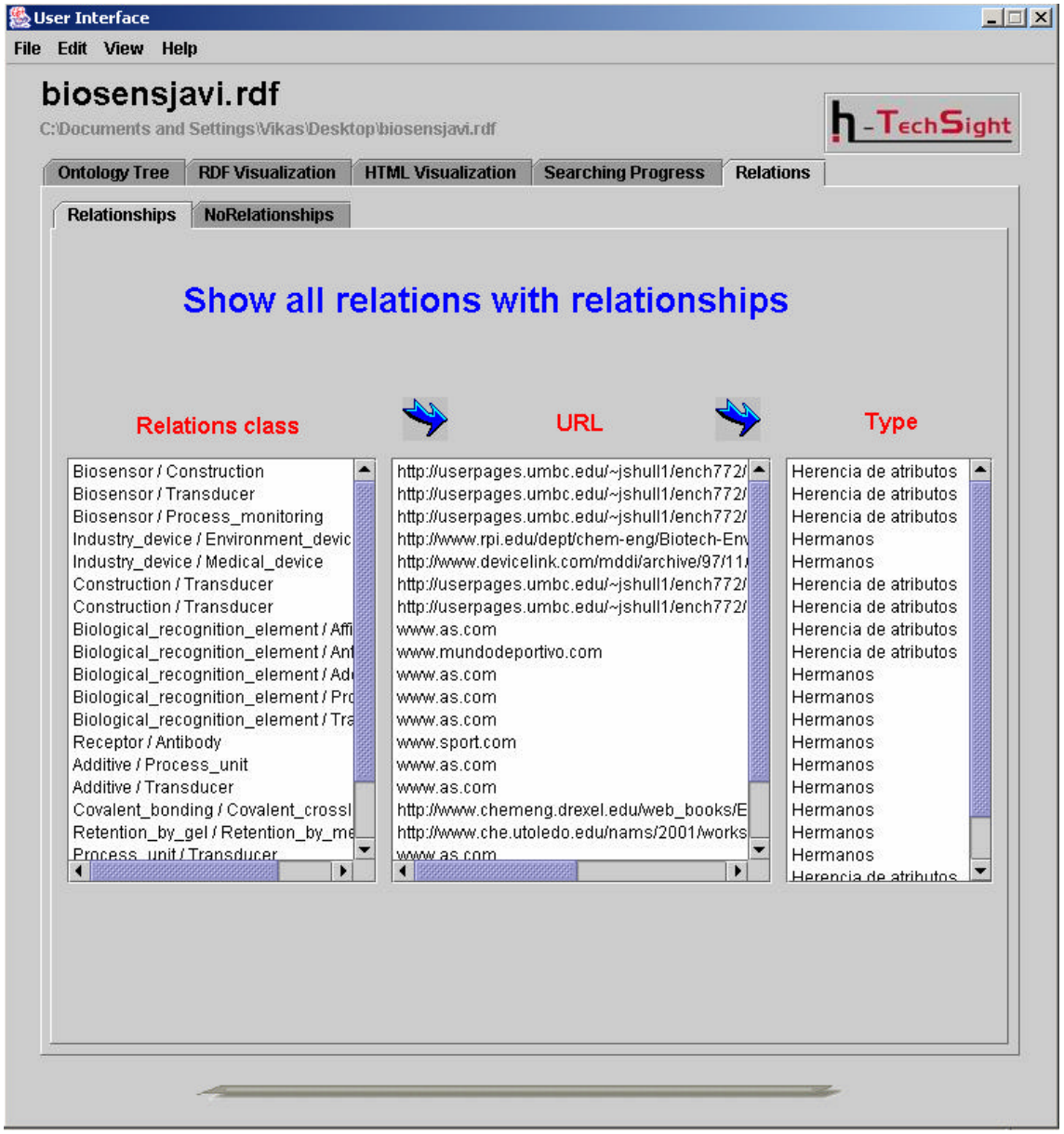
De estas tres relaciones entre URL's solo nos interesan las 2 últimas, ya que la primera nos muestra que dos clases hermanas entres si tienen una misma URL, pero es una cosa lógica porque comparten los un gran número de atributos.

En cambio las 2 últimas relaciones son entre clases que no tienen ningún tipo de relación directa, y es por eso que se seleccionan para ver si comparándolas con mas búsquedas se puede obtener algún tipo de relación que uno no podría haber determinado anteriormente.

El resultado de este segundo algoritmo de razonamiento se muestran también en la interfície de usuario, en una nueva pestaña llamada **relations,** en esta pestaña se muestran dos tipos de relaciones entre clases, la primera son las relaciones entre dos clases que tienen algún tipo de parentesco (**Relationships**), y la segunda muestra clases con URL's iguales que no tienen ningún tipo de parentesco (**NoRelationships**), y por tanto son las que realmente interesan.

Hay dos tipos de parentesco que hemos querido resaltar, como ya hemos comentado anteriormente, clasificados en la variable **type** y estos son **tipo hermanos**, y **tipo herencia de atributos**.

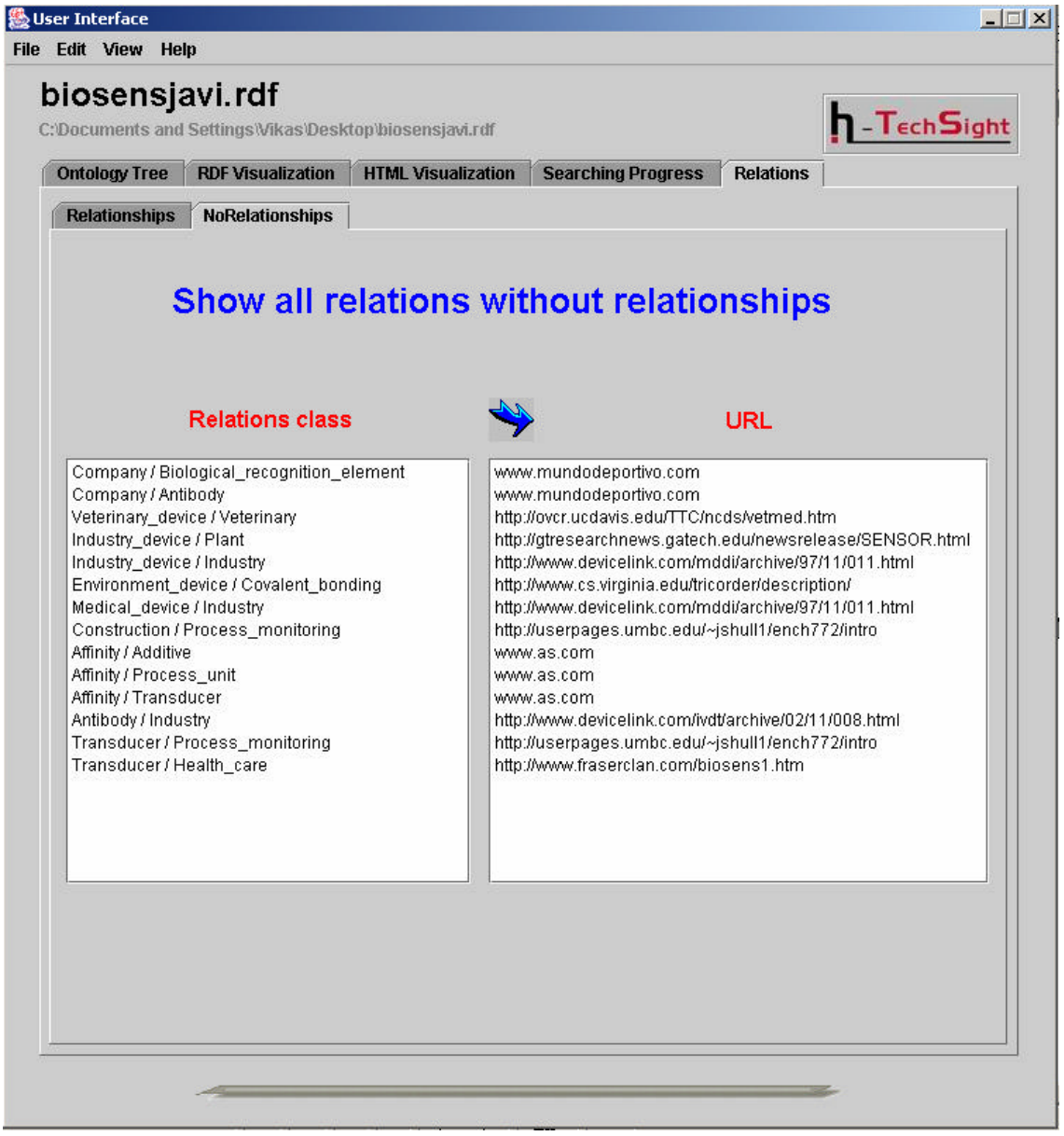
En la siguiente figura se muestra la pestaña **relations** de la interficie de usuario donde se pueden apreciar el resultado del algoritmo de razonamiento después de aplicar una búsqueda.



.

Esta figura se refiere a la ventana **Relationships** dentro de la pestaña **Relations**, como se puede ver tiene tres columnas, en la primera se especifican las clases que tienen relación, la segunda específica la URL encontrada por los agentes que pertenece a ambas clases, y la tercera especifica el tipo de parentesco que tienen las dos clases.

Por último la ventana que vienen a continuación se refiere a la otra pestañita llamada **NoRelationships,** donde solamente se muestran 2 columnas ya que como especifica clases que no tienen parentesco, no es necesaria



4.5. Sésame

4.5.1. Descripción

Sésame es una de las herramientas que he tenido que utilizar en el proyecto. Sésame es una arquitectura que provee al usuario de un eficiente almacenamiento de meta-datos expresados en RDF y RDF-Shema y un mecanismo de consulta sobre el RDF almacenado. Por tanto Sésame es una poderosa aplicación para poder comparar y consultar diferentes RDF's almacenados en una base de datos relacional.

Inicialmente esta herramienta fue desarrollada por **Aidministrador Nederland bv** [www.aidministrador.nl], como un prototipo de investigación y fue uno de los documentos clave en el **European IST proyect On-To.Knowledge** [www.ontoknowledge.org], posteriormente en el proyecto **OntoText** [www.ontotext.com], junto con el proyecto **On-To-Knowledge,** decidieron investigar las funcionalidades de Sésame y lo extendieron dotándole de modulos de seguridad para todos sus componentes. Cuando el proyecto **On-To-Knowledge** llegó a su fin, **Nlnet Foundation** [www.nlnet.nl], costeó el proyecto bajo la condición que el software se mostrase en forma de código abierto. Hoy en día, y gracias a ellos se puede encontrar Sésame como un software de código abierto.

Uno de los puntos fuertes de Sésame es su arquitectura donde para facilitar la interoperabilidad de Sésame con otras aplicaciones como por ejemplo la base de datos donde se almacenan los RDF's, el diseño y la implementación han sido construidos independientemente de éstas, por lo que independientemente de la base de datos que se utilice, Base de datos relacional, almacenamiento triple, base de datos orientada a objetos, Sésame se mantendrá invariable en su configuración y el mecanismo de consulta será válido cual sea la base de datos elegida.

Sésame se puede utilizar de dos formas diferentes una es mediante la interfaz web que se proporciona dándote de alta en www.aidministrator.nl, la cual se recomienda para empezar a familiarizarte con las funcionalidades que dispone esta aplicación o para manejar pequeñas cantidades de datos, y otra es bajándote el código como una

aplicación standalone, y configurándolo en tu ordenador, para eso necesitas una servidor web donde depositar los Jsp's de Sésame, y algún dispositivo de almacenaje, como una base de datos.

Para diseñar la primera aplicación Sésame se utilizó una api genérica de repositorios de RDF(Shema) conocida como RDF SAIL (Storage And Inference Layer), esta api provee funciones para almacenar, eliminar y consultar RDF desde un repositorio, con lo cual Sésame utilizó para diseñar sus módulos una abstracción de esta api. Si utilizas Sésame como una aplicación standalone desde tu propio ordenador, sea cual sea la base de datos que utilices para depositar tus RDF's, Sésame utilizará la misma api para ofrecerte las mismas funciones en el manejo de estos RDF's.

Actualmente Sésame dispone de 5 módulos funcionales para tratar la información en forma de RDF's que almacena en sus repositorios:

- Un modulo de administración de los datos para añadir y eliminar RDF's del repositorio.
- Un módulo para exportar los datos del RDF del repositorio a un documento con formato RDF (.RDF).
- Un mecanismo de razonamiento en RQL para evaluar consultas en RQL.
- Un mecanismo de razonamiento en RDQL para evaluar consultas en RDQL.
- Un módulo de seguridad para satisfacer la buena integridad de todos sus mecanismos.

El mecanismo de consulta utiliza como lenguaje el RQL, el único lenguaje que ofrece soporte a la semántica que aporta el RDF Shema. RQL está basado en la sintaxis de OQL, la idea que aporta este lenguaje, es que nosotros podemos ver un conjunto de modelos en RDF como un conjunto de grafos conectados, donde RQL nos ofrece

características para navegar a través de esos grafos y seleccionar especificas rutas del grafo y nodos para extraer la información.

4.5.2. Utilización y Uso

Como ya podemos suponer, en el proyecto he utilizado Sésame como repositorio de las **Information Ontology**, con el propósito de analizar la información que contienen estas Ontologías pudiendo así analizar como cambia la información que encuentran los agentes y encontrar relaciones entre los conceptos que representa la Ontología.

En este proyecto he utilizado Sésame, como aplicación standalone, bajándome el código disponible en la web de www.aidministrator.nl y siguiendo las instrucciones de configuración.
Para poder utilizar Sésame, se necesita primero que todo un servidor web, en mi caso he utilizado **Apache Tomcat,** ya que según he leído en referencias decían que habían testado Sésame con este servidor y les había dado buen resultado. Este servidor se utiliza para poder ejecutar los Jsp's que dispone Sésame, que hacen posible utilizar todos las funciones que esta aplicación incorpora. A parte del servidor, también necesitamos una base de datos donde guardaremos los RDF's, en mi caso he utilizado **Mysql,** por las mismas razones que el servidor, para interoperar con **mysql,** tan solo tienes que copiar los drivers de acceso a la base de datos **JDBC** en el directorio donde tendrás todo el código de Sésame.

Para utilizar Sésame como repositorio, primero has de crear una cuenta de usuario para poder acceder a las bases de datos creadas, para crear tu cuenta lo único que has de hacer es modificar el fichero **System.conf,** donde reside toda la configuración de Sésame. Una vez has creado una cuenta de usuario, tienes que crear las bases de datos que utilizarás para guardar las **Information ontology,** esta acción también se lleva a cabo en el mismo fichero, primero tienes que elegir con que base de datos trabajarás (MySQL), y después crear los repositorios dentro de esa base de datos, dando acceso al usuario que has creado. En mi caso tendré 3 repositorios, llamados **repository-1**,

repository-2, **repository-3**, los cuales los utilizaré para guardar en cada uno una **Information Ontology** para después poder analizar la información que contienen.

Código para la creación de una cuenta de usuario en Sésame

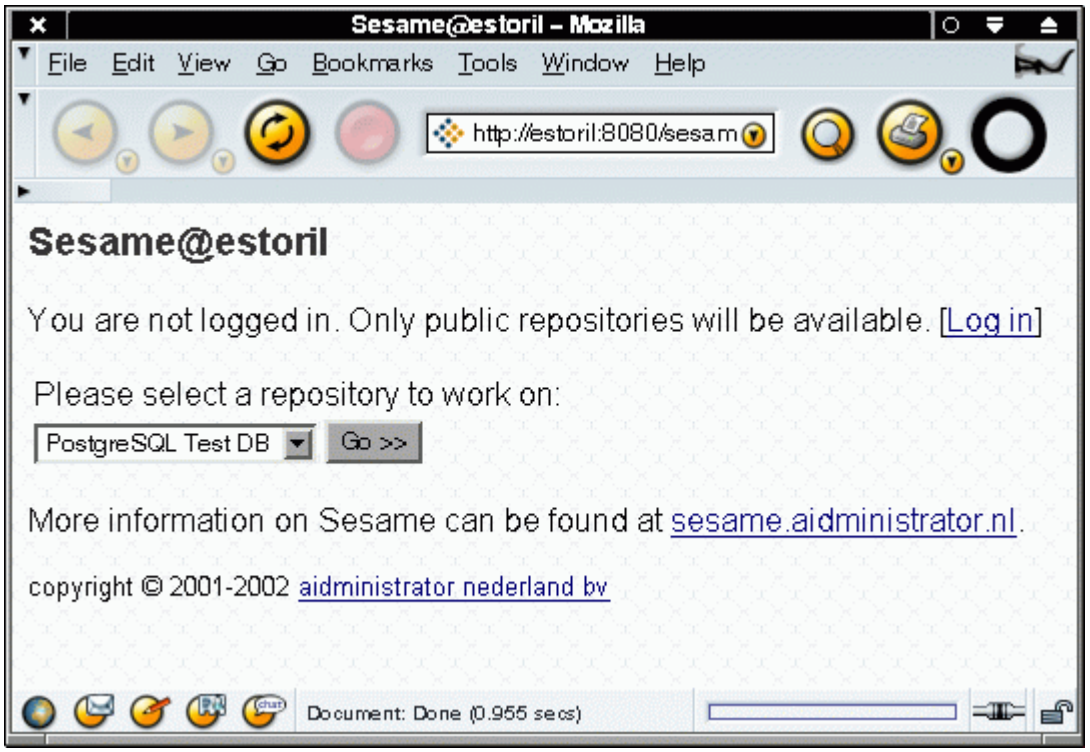
```
<user login="jsanchez" id="10">
     <fullname>Javier Sánchez</fullname>
     <password>chezsanj</password>
</user>
```

Código para la creación de una base de datos para almacenar RDF's en Sésame

```
<repository id="mysql-db1">
<title>MySQLRepository-1</title>

<sailstack>
<sailclass="nl.aidministrator.rdf.sail.sync.SyncRdfSchemaRe
pository"/>
<sail class="nl.aidministrator.rdf.sail.sql92.MySQLSail">
<param name="jdbcDriver" value="com.mysql.jdbc.Driver"/>
<param
name="jdbcUrl"value="jdbc:mysql://localhost:3306/repository
1"/>
<param name="user" value="administrador"/>
<param name="password" value="ljimenez"/>
 </sail>
</sailstack>

<acl worldReadable="false" worldWritable="false">
<user login="jsanchez" readAccess="true"
writeAccess="true"/>
</acl>

</repository>
```

¿Como usar Sésame?

Una vez has configurado Sésame, para que puede interactuar con el servidor web y la base de datos, tienes que acceder al directorio de Sésame vía web donde se cargará un Jsp de inicio **index.jsp.** En esta página de inicio te saldran las bases de datos que has creado, disponibles para cualquier usuario, en este caso solo saldrá **PostgreeSQL y TestDB** , porque las demás no son accesibles para todos los usuario, y para verlas deberás registrarte en la casilla [**Log in]** con el nombre y el password que habías puesto para el usuario que has creado.



Si accionas la casilla [Log in], saltará un Jsp llamado **login.jsp,** donde podrás registrarte como usuario.

Si el usuario no existe, la aplicación saltará a un Jsp de error de login, si en cambio el usuario ya esta registrado saltará al Jsp inicial en donde se podrán ver todas las bases de datos disponibles para ese usuario. En mi caso el usuario se llama "**jsanchez**", y el password "**chezsanj**", este usuario puede ver todas las bases de datos que se han creado en el fichero **System.conf,** aunque solo las de **Mysql** serán válidas, porque es la base de datos que utilizo y ya están creados los repositorios. Si accionase otra base de datos, el programa daría un error puesto que vería que quiero acceder a una base de datos que no está creada.

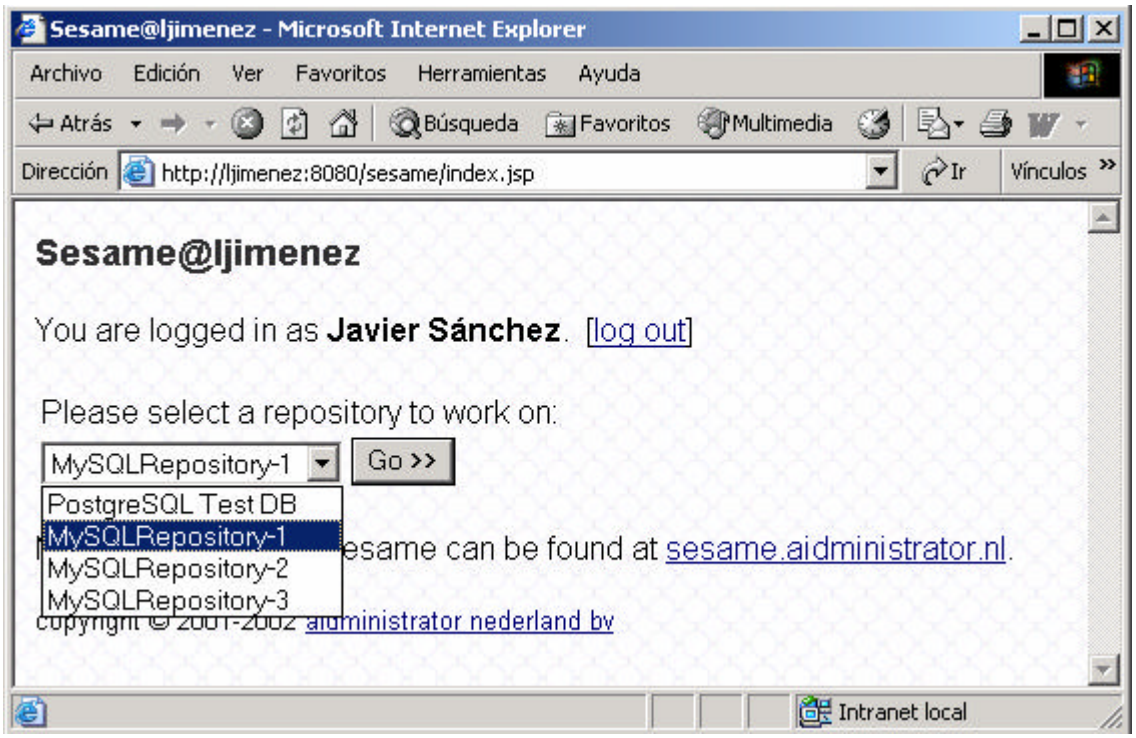
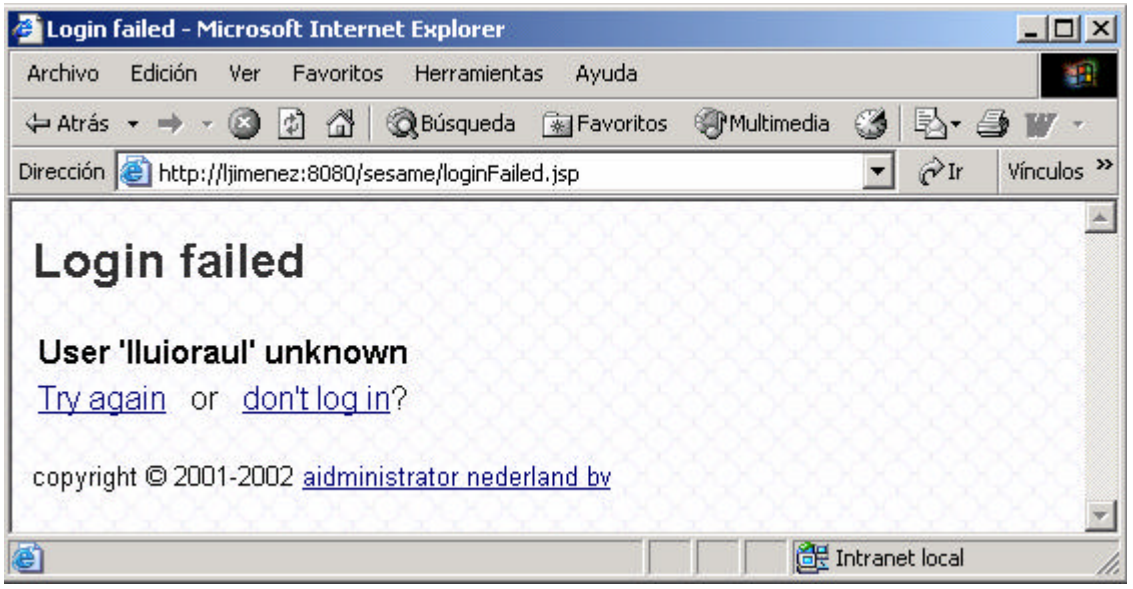
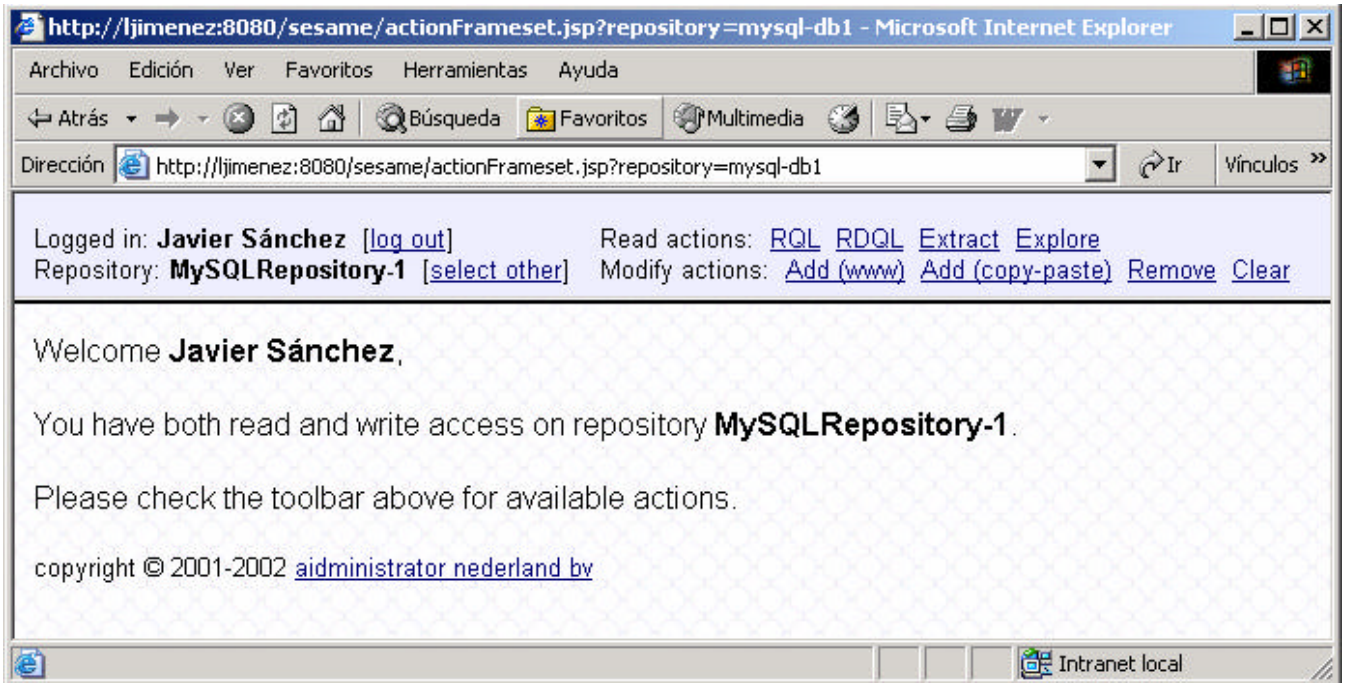


Fig. Ventana donde se muestra la aplicación una vez he dado de alta al usuario Javier Sánchez, y donde se muestran los tres repositorios que he creado para guardar la Ontologías.

En cambio, si el usuario no se hubiese creado, la aplicación hubiera saltado a un Jsp de error de login como muestra la siguiente ventana:



Una vez dado de alta, podemos seleccionar la base de datos que queramos para guardar nuestro primer RDF, en este caso podríamos elegir la primera que hemos creado, **MySQLRepositoy-1.** Una vez seleccionada esta base de datos la aplicación nos saltaría a otro Jsp donde en este ya se puede comenzar a manejar los RDF.



En esta ventana ya nos aparecen todas las funciones que se pueden hacer utilizando la aplicación Sésame. A la izquierda de la ventana tenemos dos funciones que son cerrar la sesión con el usuario **[Log out]** y seleccionar otra base de datos **[Select other]**.

Luego tenemos una seria de funciones clasificadas en 2 grupos diferentes:

i) Acciones para modificar los RDF's

Add(www): Esta función añade un RDF a partir de una URL, puedes indicar el Path donde esta ubicado este RDF y el mismo programa lo cargará en Sésame, guardándolo en la base de datos.

Add(copy-paste): Otra opción menos elegante, es copiar el código de una documento RDF y pegarlo directamente en un "Text Area" que contiene la ventana, y Sésame también de encargará de guardarlo en la base de datos.

Remove: Esta opción elimina cualquier tipo de recurso RDF como una clase o una propiedad, puedes indicarle por ejemplo que clase quieres eliminar de tu RDF y Sésame lo eliminará del repositorio.

Clear: Esta opción sirve para borrar literalmente el contenido del repositorio con el que estas trabajando.

ii) Acciones para consultar los RDF's

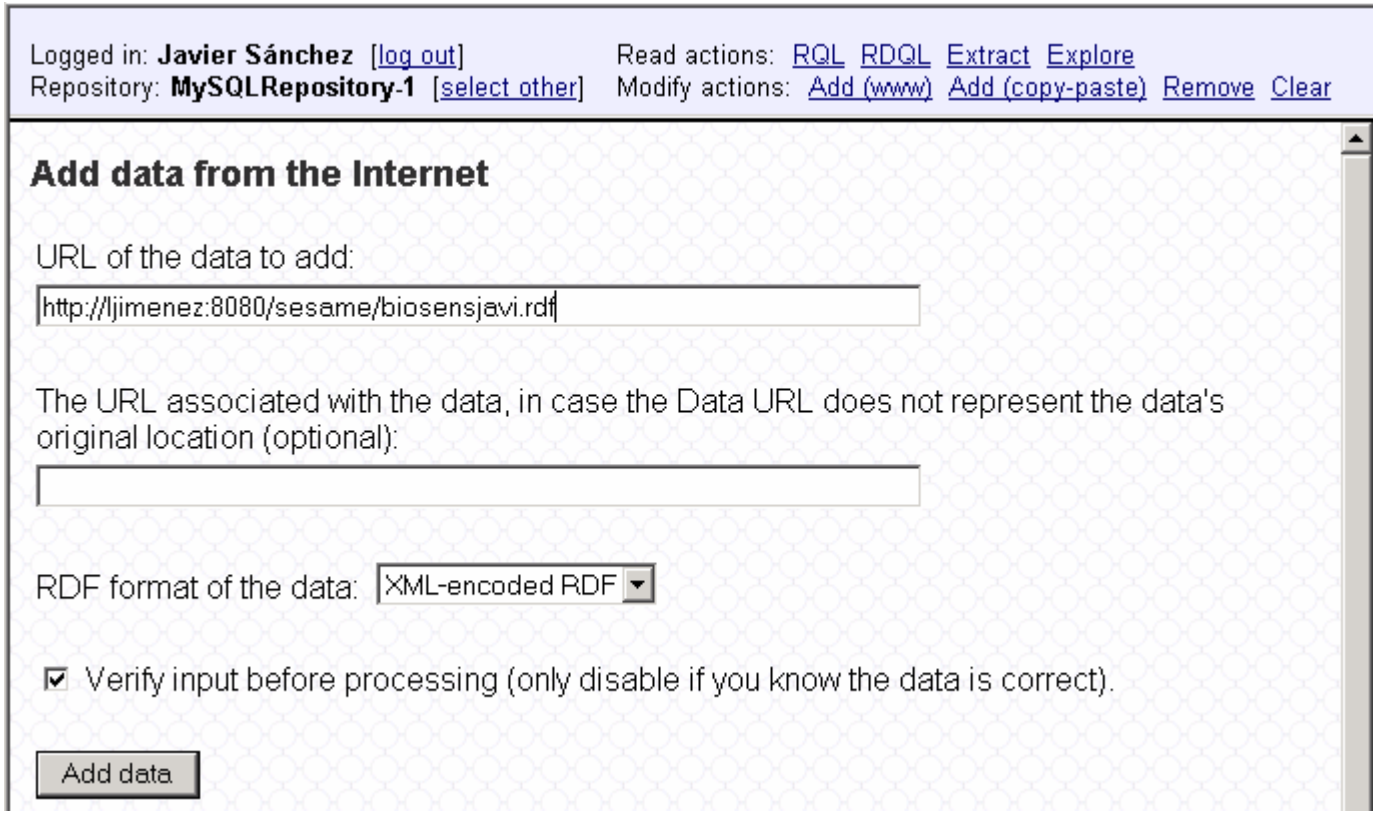
RDQL: Con esta opción puedes hacer consultas sobre tu RDF con el lenguaje RDQL, y de esta forma explorar manualmente la información que contiene tu Ontología.

RQL: Con esta opción puedes hacer lo mismo que con la opción de RDQL, haces consultas ahora bien con el lenguaje RQL.

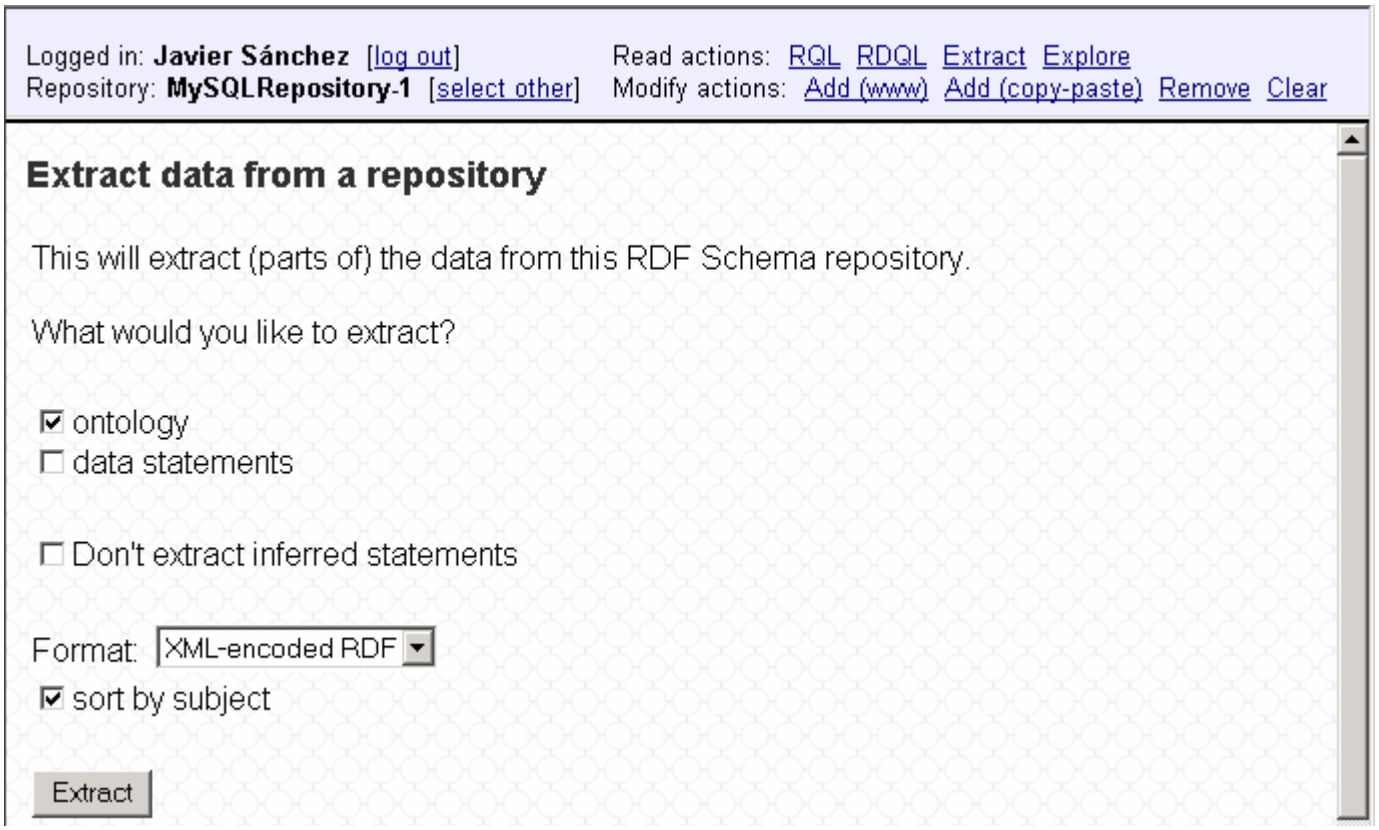
Extract: Esta opción te permite extraer el RDF del repositorio para poder utilizarlo en otras aplicaciones, hay tres tipos de formato en el que se puede extraer un RDF: **XML-Encoded RDF**, N-Triples, **Notion 3/N 3**.

Explore: Con esta opción puede explorar el contenido que se haya en el repositorio, donde puedes indicarle que te clasifique la información en clases, propiedades etc.

Las opciones que más he tenido que utilizar es la add(www), para añadir la **Information Ontology** en el repositorio, y la de extract para una vez metido el RDF extraerlo y probar si realmente no se ha perdido información en el paso de **añadir-extraer.** Refiriéndome a esto último he podido comprobar que en este paso no hay pérdida de información alguna, ya que para comprobarlo vuelvo a cargar la Information Ontology en la interfaz de usuario del sistema multiagentes, y ésta sigue conteniendo el mismo número de links que antes.



En la figura anterior se muestra la ventana para añadir RDF's, en este caso se esta queriendo añadir **biosensjavi.rdf**, y para eso se ha de especificar la ruta completa.
En esta misma ventana también puede elegir 2 tipos de formatos para que Sésame guarde tu RDF en la base de datos, el primero (y el que yo siempre utilizo) es **XML-Encoded RDF,** y el segundo **N-Triples**.



Esta otra ventana representa la opción de extraer un RDF del repositorio. Tienes la opción de extraer la Ontología "a secas", o la Ontología con todos los datos añadidos en cada una de las propiedades que ésta contiene. En mi caso como lo que necesito es extraer la Ontología con las páginas webs escogeré las casillas :

- ☐ Ontology
- ☐ Data statements

También puede elegir que el RDF contenga para cada clase los atributos que esta heredando de clases anteriores, con la casilla "Don't extract inferred statements", en mi caso siempre activo esta casilla porque si no la activase el sistema multiagentes no me reconocería este RDF como formato de OntoEdit, y no b cargaría en la interfaz de usuario. Por último por lo que hace referencia a casillas tenemos la opción "Sort by subject", con la cual podemos clasificar nuestro RDF por tipo de recurso, por clase por propiedades y si hubiese por instancias.

Ya por último como hemos comentado anteriormente, la extracción de un RDF del repositorio se puede hacer en tres tipos distintos de formatos, yo, como os podéis imaginar, utilizo XML-encoded RDF ya que la Ontología en RDF la añado con el mismo formato, y éste es el único que reconoce el sistema multiagentes.

5. CONCLUSIONES

Este proyecto, me ha ayudado a pasar de tener una visión muy general de la estructura de la información en Internet a tener una visión concreta, de cómo está expuesta la información en este medio y sobre todo como puede estarlo para aprovechar el conocimiento que en él reside.
También he aprendido, un concepto totalmente desconocido para mi antes, como son las Ontologías, y como éstas juegan una papel fundamental en la representación del conocimiento, y no solo eso si no que sirven de soporte imprescindible a sistemas multi-agente y programas automatizados que han de navegar por Internet, facilitando la interoperabilidad de éstos al partir de una misma fuente de información.

El proyecto en sí me ha gustado bastante, porque estaba enmarcado dentro del campo de la Inteligencia Artificial, y el hecho de haber realizado Inteligencia Artificial I como asignatura optativa de la carrera, me ha facilitado las cosas a la hora de entender conceptos como, que es un sistema multi-agentes, como se comunican los agentes entre sí., o a la hora de diseñar algoritmos de recorrido efectivos sobre la Ontología.

Como ya habréis podido ver, el proyecto ha tenido una gran parte teórica, donde no había mucha dificultad, pero si bastante trabajo. En cambio la parte práctica es la que me ha llevado mas tiempo, ya que todo el código que he tenido que diseñar, lo debía insertar en un programa que ya estaba hecho, y por tanto antes de todo tenía que entender muy bien como estaba diseñado ese programa. Además herramientas como Sésame, o la interoperabilidad de documentos RDF entre programas, no estaban muy estudiados, por lo que he tenido que averiguar en cierta medida como podía conectar varios de esos programas de forma eficiente.

En resumen, este proyecto me ha proporcionado una visión más amplia de lo que es hoy Internet y de lo que puede llegar a ser con la utilización de herramientas que organicen de una manera inteligente toda la información que aún está por domar.

Por último me gustaría dar las gracias al grupo **Grusma** en general, por admitirme en su grupo de trabajo, y por darme todas las facilidades que he tenido para desarrollar este proyecto, así como a David Isern y Jaime Bocio por la gran ayuda que me han ofrecido.

6. REFERENCIAS

[1] R. Neches, R.E. Fikes, T. Finin, T.R. Gruber, T. Senator, W.R. Swartout, Enabling technology for knowledge sharing, AI Magazine 12 (3) (1991).

[2] T.R. Gruber, Atranslation approach to portable ontology speci.cation, Knowledge Acquis ition 5 (1993) .

[3] W.N. Borst, Construction of Engineering Ontologies, PhD Thesis, University of Tweenty, Enschede, NL Centre for Telematica and Information Technology, 1997.

[4] R. Studer, V.R. Benjamins, D. Fensel, Knowledge engineering: principles and methods, Data and Knowledge Engineering 25 (1998).

[5] N. Guarino, M. Carrara, P. Giaretta, Ontologies and knowledge bases: towards a terminological clarification, in:
N. Mars (Ed.), Towards Very Large Knowledge Bases, Knowledge Building and Knowledge Sharing, IOS Press, Amsterdam, 1995.

[6] A. Bernaras, I. Laresgoiti, J. Corera, Building and reusing ontologies for electrical network applications, in: Proc.
European Conference on Arti.cial Intelligence (ECAI_96), Budapest, Hungary, 1996,

[7] D.B. Lenat, R.V. Guha, Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project, Addison-Wesley, Boston, 1990.

[8] M. Uschold, M. King, Towards a Methodology for Building Ontologies, in: IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.
O. Corcho et al. / Data & Knowledge Engineering xxx (2002) xxx–xxx 23

[9] M. Gruninger, M.S. Fox, Methodology for the design and evaluation of ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.

[10] A. Gomez-Perez, M. Fernandez-Lopez, A. de Vicente, Towards a Method to Conceptualize Domain Ontologies, in: ECAI96 Workshop on Ontological Engineering, Budapest, 1996.

[11] B. Swartout, P. Ramesh, K. Knight, T. Russ, Toward Distributed Use of Large-Scale Ontologies, AAAI Symposium on Ontological Engineering, Stanford, 1997.

[12] S. Staab, H.P. Schnurr, R. Studer, Y. Sure, Knowledge processes and ontologies, IEEE Intelligent Systems 16 (1) (2001).

[13] J. Euzenat, Corporative memory through cooperative creation of knowledge bases and hyper-documents, in: Proc. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW96), Ban., 1996.

[14] M. Fernandez-Lopez, A. Gomez-Perez, A. Pazos-Sierra, J. Pazos-Sierra, Building a chemical ontology using METHONTOLOGY and the ontology design environment, IEEE Intelligent Systems & their applications (1999).

[15] I. Horrocks, F. van Harmelen, Reference Description of the DAML+OIL (March 2001) Ontology Markup Language,Technical report, 2001. Available from <http://www.daml.org/2001/03/reference.html.>

[16] R. MacGregor, Inside the LOOM clasifier, SIGART bulletin 2 (3) (1991) .

[17] E. Motta, Reusable Components for Knowledge Modelling, IOS Press, Amsterdam, 1999.

[18] M. Kifer, G. Lausen, J. Wu, Logical foundations of object-oriented and frame-based languages, Journal of the ACM 42 (4) (1995).

[19] S. Luke, J. He.in, SHOE 1.01. Proposed Speci.cation, SHOE Project technical report, University of Maryland,2000.
Available from <http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm>.

[20] T. Bray, J. Paoli, CM. Sperberg-McQueen, E. Maler, Extensible Markup Language (XML) 1.0, second ed., W3C
Recommendation, 2000. Available from <http://www.w3.org/TR/REC-xml>.

[21] R. Karp, V. Chaudhri, J. Thomere, XOL: An XML-Based Ontology Exchange Language, technical report, 1999.
Available from <http://www.ai.sri.com/_pkarp/xol/xol.html>.

[22] O. Lassila, R. Swick, Resource description framework (RDF) model and syntax specification, W3C Recommendation (1999), http://www.w3.org/TR/REC-RDF-syntax/.

[23] D. Brickley, R.V. Guha, RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft, 2002. Available from <http://www.w3.org/TR/PR-RDF-schema>.

[24] I. Horrocks, D. Fensel, F. Harmelen, S. Decker, M. Erdmann, M. Klein, OIL in a Nutshell, in: ECAI_00 Workshop on Application of Ontologies and PSMs, Berlin, 2000.

[25] A. Farquhar, R. Fikes, J. Rice, The Ontolingua Server: A Tool for Collaborative Ontology Construction, in:
Proc. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW96) Ban., 1996.

[26] J. Domingue, Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web, in: Proc. 11th Knowledge Acquisition Workshop (KAW98), Ban., 1998.

[27] N.F. Noy, R.W. Fergerson, M.A. Musen, The knowledge model of protege-2000: combining interoperability and flexibility, in: 2th International Conference in Knowledge Engineering and Knowledge Management (EKAW_00), Lecture Notes in Artificial Intelligence, vol. 1937, Springer, Berlin, 2000.

[28] N.F. Noy, M.A. Musen, PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, in: 17th National Conference on Artificial Intelligence (AAAI_00), Austin, 2000.

[29] J.C. Arpírez, O. Corcho, M. Fern andez-L opez, A. G omez-P erez, WebODE: a scalable ontological engineering
workbench, in: First International Conference on Knowledge Capture (KCAP_01), ACM Press, Victoria, 2001.

[30] M. Blazquez, M. Fernandez-Lopez, J.M. García-Pinar, A. Gomez-Perez, Building ontologies at the knowledge level using the ontology design environment, in: B.R. Gaines, M.A. Musen (Eds.), 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW_98) Ban., 1998.

[31] O. Corcho, M. Fernandez-Lopez, A. Gomez-Perez, O. Vicente, WebODE: an integrated workbench for ontology representation, reasoning and exchange, in: A. Gomez-Perez, V.R. Benjamins (Eds.), 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW_02), Lecture Notes in Artificial Intelligence, vol. 2473, Springer, Berlin, 2002.

[32] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke, OntoEdit: collaborative ontology engineering for the semantic web, in: First International Semantic Web Conference (ISWC_02), Lecture Notes in Computer Science, vol. 2342, Springer, Berlin, 2002.

7. APÉNDICE: CÓDIGO FUENTE

Algoritmos de razonamiento sobre la Ontología

Funciones Auxiliares

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Comparamos si dos nodos contienen alguna página web común
public List comparar_webs (List lista,int child) {

    // variables locales
  InformationURL pag;
  RDFClass nodo=new RDFClass();
  String info;
  int trobats;
  List url=new ArrayList();
  Iterator it=lista.iterator();


if (it.hasNext()) nodo=(RDFClass)it.next();

   Iterator itweb=nodo.getAllWeb_pages();

 while (itweb.hasNext()){

        Iterator it2=lista.iterator();
        trobats=0;
        pag=(InformationURL)itweb.next();

      while(it2.hasNext()){

         Iterator it2web=((RDFClass)it2.next()).getAllWeb_pages();

             while (it2web.hasNext()) {

          info=((InformationURL) pag).getUrl();
          if(info.equals((((InformationURL)it2web.next()).getUrl())))
          {
            trobats=trobats+1;
          }
```

```
                }
            }
            if(trobats==child) url.add(pag);
        }

        return (url);
    }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//  comparamos si dos listas son iguales
private boolean comparar_taules(List t1,RDFClass n){

// variables locales
  boolean cont=false;
  String nom;
  Iterator it=t1.iterator();

  while(it.hasNext()) {
     nom=((String) it.next());
    if(((n.getName()).equals(nom))){
      cont=true;
    }
  }
  return(cont);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//  comparamos si dos listas son iguales como antes pero con diferentes parámetros de entrada.
private boolean comparar_taules2(List nodos,String name){
// variables locales
   Iterator it=nodos.iterator();
   String nod;
   boolean trobat=false;

   while (it.hasNext()){
     nod=(String)it.next();
     if ((nod).equals(name)) trobat=true;

  }
     return(trobat);
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Rellenamos la lista "nodos" donde estará todos los conceptos representados de la Ontología
  private void omplir_nodes(RDFClass node) {

    // Variables locales
    RDFClass nodo;
    eliminar_duplicats(node);
    nodos.add(node);

    if (node.getIs_instance()){
          return;
    }

    else {
      Iterator it=node.getAllChildren();
         while (it.hasNext()) {
           nodo=(RDFClass)it.next();
           omplir_nodes(nodo);
           }


       }
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Algoritmos

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

 // Aplicamos el algoritmo de recorrido en post-orden para recorrer el árbol, y razonamos con él subiendo a un nivel superior las páginas que se repiten en los hijos.

  private void recorregut (RDFClass node) {

 // Variables locales
    RDFClass nodo=new RDFClass();
    RDFClass nod=new RDFClass();
    List url=new ArrayList();
```

```
  if (node.getIs_instance()){
        return;
  }

  else {
    Iterator it=node.getAllChildren();
       while (it.hasNext()) {
         nodo=(RDFClass)it.next();
         recorregut(nodo);
         }
       url=(List)comparar_webs(node.getChildren(),node.get_number_of_children());
       Iterator it2 =url.iterator();
       while(it2.hasNext()){
       node.addWeb_pages((InformationURL) it2.next());
       }
       Iterator it3=node.getAllChildren();

         while(it3.hasNext()){
           Iterator it4 =url.iterator();
           nod=(RDFClass)it3.next();
             while (it4.hasNext()){
               nod.removeWeb_pages((InformationURL) it4.next());

             }
         }

      }

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Buscamos si nodos que no tienen ningún tipo de parentesco contienen webs comunes
  private void relations(){

  // Variables locales
   RDFClass nodo;
   RDFClass nodo2;
   Iterator it =nodos.iterator();
   List webs=new ArrayList();
   List url=new ArrayList();
   InformationURL temp;
   String info;
   String varA=" ",varB=" ";
   boolean cont1=false;
   boolean cont2=false;
```

```
while (it.hasNext()){
  Iterator it2=nodos.iterator();
  nodo=(RDFClass)it.next();
  nodo.clearAllRelations();
    while (it2.hasNext()){
      nodo2=(RDFClass)it2.next();
      webs.clear();
      webs.add(nodo);
      webs.add(nodo2);

      url.clear();
      url=comparar_webs(webs,2);
      Iterator it3=url.iterator();
      Nodo nod=new Nodo();
        while (it3.hasNext()) {
          temp=(InformationURL)it3.next();
          nod.name=nodo2.getName();
          nod.url=((InformationURL)temp).getUrl();
          info=nodo.getName();
          List fills1=nodo.getAncestors();
          List fills2=nodo2.getAncestors();

          if (!fills1.isEmpty()) varA=(String) fills1.get(fills1.size()-1);
          if (!fills2.isEmpty()) varB=(String) fills2.get(fills2.size()-1);

            cont1=comparar_taules(fills1,nodo2);
            cont2=comparar_taules(fills2,nodo);



          if (varA == varB) {
           nod.setTipParent("Hermanos");
           nod.setParent(true);
          }
         if(cont1 || cont2) {
           nod.setTipParent("Herencia de atributos");
           nod.setParent(true);
         }
          if (!info.equals(nodo2.getName())){
          nodo.addNodes(nod);
         }

         }
        }
    }

}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Mostramos las relaciones entre nodos mediante una interficie gráfica.
  private void mostrarRelations() {

 // variables locales
  RDFClass nodo;
  Nodo nod=new Nodo();
  boolean trobat=false;
  Iterator it =nodos.iterator();


    while (it.hasNext()) {

     nodo=(RDFClass)it.next();
      Iterator it2=nodo.getAllRelation();

       while (it2.hasNext()) {
        nod=(Nodo)it2.next();
        trobat=comparar_taules2(nodostrob,nod.getName());
       if(!trobat) {
        if     (nod.getParent()){     gui.jTextArea1.append("     "+nodo.getName()+"     /
"+nod.getName()+"\n");
                        gui.jTextArea3.append(" "+nod.getURL()+"\n");
                        gui.jText Area4.append(" "+nod.getTipParent()+"\n");
         }

         else { gui.jTextArea2.append(" "+nodo.getName()+" / "+nod.getName()+"\n");
              gui.jTextArea5.append(" "+nod.getURL()+"\n");

            }
        }

    }
    nodostrob.add(nodo.getName());
  }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Eliminamos duplicaciones de links en una misma clase
  public void eliminar_duplicats(RDFClass node){
 // Variables locales
   Iterator it =node.getAllWeb_pages();
   InformationURL info,info2;
   List elim=new ArrayList();
   int n=0;

   while(it.hasNext()) {
```

```
        info=(InformationURL) it.next();
        Iterator it2 =node.getAllWeb_pages();
        elim=new ArrayList();
        n=0;

          while (it2.hasNext()){
            info2=(InformationURL) it2.next();
            if ((info.getUrl()).equals(info2.getUrl())) {
                  n=n+1;
                  if (n>=2) elim.add(info2);
              }
           }
       }

       Iterator it3=elim.iterator();
          while(it3.hasNext()){
           node.removeWeb_pages((InformationURL)it3.next());
           }
          while(it3.hasNext()){
           node.addWeb_pages((InformationURL)it3.next());
           }
     }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    // Aplicamos razonamiento sobre el árbol ontológico
    private void razonamiento_onto(Ontology_TreeModel ont){

      // variables locales
      RDFClass nodo;

    recorregut((RDFClass)ont.getRoot());
    omplir_nodes((RDFClass)ont.getRoot());
    relations();
    mostrarRelations();
    gui.panel_tabulado.setEnabledAt(4,true);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```