UNIVERSIDAD DE JAÉN

Ontologías, Servicios Web y terminología basada en corpus aplicados a la descripción de la calidad del aceite de oliva.

Trabajo Tutelado de Iniciación a la Investigación

Alumno: Isabel Vico Peinado

Tutor: Dr. Pedro J. Sánchez Sánchez

Cotutor: Dra. Mercedes Roldán Vendrell

Curso 2012/2013

Jaén, Diciembre 2012

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dr. D. Pedro J. Sánchez Sánchez, director del Trabajo Tutelado de Iniciación a la Investigación titulado: Ontologías, Servicios Web y terminología basada en corpus aplicados a la descripción de la calidad del aceite de oliva, que presenta Dña. Isabel Vico Peinado, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Diciembre de 2012

El alumno: El director:

Dña. Isabel Vico Peinado Dr. D. Pedro J. Sánchez Sánchez

Índice general

Índice de figuras **1**

1. Introducción **3**

1.1. Motivación 3

1.2. Contexto 5

1.2.1. Web semántica 5

1.2.2. Ontologías 8

1.2.3. Servicios web 9

1.2.4. Terminología 10

1.3. Propósito y objetivos 12

1.4. Estructura de la memoria 12

2. Ontologías, servicios web y terminología basada en corpus **15**

2.1. Ontologías 16

2.1.1. Definición 16

2.1.2. Historia 23

2.1.2.1. Metodologías 25

2.1.2.2. Lenguajes 39

2.1.3. Clasificación 49

2.1.4. Aplicaciones 50

2.2. Servicios web 51

2.2.1. Definición 52

2.2.2. Arquitectura 54

2.2.3. Retos y aplicaciones 61

2.2.3.1. Retos 61

2.2.3.2. Aplicaciones 61

2.3. Terminología basada en corpus 62

2.3.1. Definición 63

2.3.2. Metodología 64

2.3.3. Olivaterm 69

3. Integración de ontologías y servicios web **73**

3.1. Trabajos relacionados 73

3.2. Análisis y diseño 77

3.3. Desarrollo e implementación 85

3.3.1. Desarrollando la ontología de dominio 85

3.3.1.1. Preguntas de competencia 85

3.3.1.2. Consideración de reutilización de ontologías 86

3.3.1.3. Enumeración de términos importantes 87

3.3.1.4. Definición de clases y jerarquía de clases y definición de las propiedades de las clases 88

3.3.1.5. Definición de las restricciones de las propiedades 93

3.3.1.6. Creación de instancias 96

3.3.2. Desarrollando e integrando el servicio web 98

4. Conclusiones y trabajo futuro **107**

A. Glosario de términos de las ontologías **111**

Bibliografía **127**

Índice de figuras

2.1. Fases para la metodología Cyc 25

2.2. Fases para la metodología Gruninger y Fox 27

2.3. Fases para la metodología Unschold y King 29

2.4. Fases para la metodología Kactus 30

2.5. Fases para la metodología Methontology 31

2.6. Fases para la metodología Sensus 34

2.7. Fases para la metodología On-to-knowledge 35

2.8. Fases para la metodología Ontology 101 development process 37

2.9. Arquitectura de un servicio web 56

2.10. Esquema de capas de los servicios web 60

2.11. Sistema conceptual de Olivaterm 71

3.1. Jerarquía de subclases de la clase Grasa 89

3.2. Jerarquía de clases de la clase Componente 90

3.3. Jerarquía de clases de la clase Propiedad Organoléptica 90

3.4. Relaciones entre las clases 91

3.5. Tabla de clasificación 91

3.6. Categorías comerciales 92

3.7. Atributos del aceite de oliva 94

3.8. Relación “Contiene” 94

3.9. Relación “Evaluado” 95

3.10. Relación “Hecho” 95

3.11. Relación “Definido” 97

3.12. Estructura de un documento SOAP 98

3.13. Estructura de un documento WSDL 102

Capítulo 1

Introducción

Para la realización del *Trabajo Tutelado de Iniciación a la Investigación* en la etapa de formación del *Programa de Doctorado en Ingeniería y Arquitectura* impartido en la *Universidad de Jaén* se redacta la presente memoria, resultado del trabajo de investigación titulado: *"Ontologías, servicios web y terminología basada en corpus aplicados a la descripción de la calidad del aceite de oliva"*. Se comienza exponiendo al lector la motivación para el desarrollo del proyecto y el contexto en el que se sitúa, para continuar con el propósito y los objetivos perseguidos y finalmente definir la estructura que seguirá la memoria.

1.1. Motivación

Según Gruber, [41], la WWW (World Wide Web) se puede definir como un conjunto de documentos alojados en diferentes servidores y distribuidos por todo el mundo. En 1983 ARPANet hizo público su protocolo TCP/IP, se trataba de protocolos de comunicación que, actualmente, aún son los principales estándares utilizados para la transmisión de cualquier página en Internet [29]. A finales de los 80 y principios de los 90, Tim Berners-Lee y Robert Caillau (ambos del CERN), pretendían que los investigadores pudiesen consultar cualquier artículo que estuviese almacenado en un servidor. Comenzaban la implementación de lo

que sería considerado el primer navegador web, denominado WorldWideWeb, que fue una herramienta para poder visitar las páginas web. Por el momento sólo consistía en un servidor y una página web, limitándose ésta a texto, más tarde fue renombrado como Nexus para diferenciar la herramienta de la información abstracta que suponía Internet. Tanto esta herramienta como otras desarrolladas en ese tiempo permitían al usuario no sólo visitar una página, sino navegar de servidor en servidor. Poco a poco fueron desarrollándose permitiendo buscar información en páginas más lejanas. Otro navegador, Gopher [91, 51] incluso guiaba a través de un menú para poder visitar páginas alojadas en el mismo servidor. Pero no fue hasta la creación de Mosaic, desarrollado por Marc Andreesen y Eric Bina del NCSA (National Center or Supercomputing Application) en 1993, cuando Internet alcanzó su fama.

Una gran contribución a la divulgación de la WWW se debe al W3C (World Wide Web Consortium) [5], esta comunidad pretende asegurar el crecimiento y desarrollo de la web, proporcionando al usuario estándares con el fin de llevar esta tecnología a una "web única".

El motivo de que la WWW sea tan universal es debido a sus estándares de identificación, interacción y formato. En la WWW se han de identificar los recursos que se tienen, siendo esto muy importante, ya que permitirá identificar cada vínculo sin tener en cuenta dónde está situado en la web. Una vez que los recursos se hayan identificado, será necesario establecer un protocolo que haga la interacción más cómoda y eficiente. Esto es posible gracias a la arquitectura cliente-servidor [48] en la que se sustenta la web. Una vez validado el protocolo, el servidor enviará al cliente la información que éste presentará al usuario, dicha información son las conocidas páginas web. Estas páginas se desarrollan en un lenguaje de programación etiquetado, como HTML (HyperText Markup Language)[3]; y son únicas: para que cada recurso pueda identificarse de manera inequívoca, será necesario que tenga un código único, esto es lo que se denomina URI (Uniform Resource Identifiers) [29].

Hasta ahora la web es una herramienta que cumple su finalidad: dos usuarios pueden perfectamente comunicarse a través de páginas web. El problema surgirá cuando esos usuarios

no sean personas, sino ordenadores. Es entonces cuando la comprensión de lo escrito hasta el momento en la WWW adquiere una dificultad añadida, debido a que las personas son capaces de diferenciar según la semántica, a qué se refiere cada palabra u oración en una página web. Sin embargo, un ordenador, será incapaz de distinguir si una palabra se refiere a un contexto u otro, ya que no están previstos para tener en cuenta la semántica de cada palabra y oración. Esto ocurre porque en cualquier idioma, una palabra u oración toma un significado u otro según la semántica de las palabras utilizadas. Para solucionar esta cuestión surge el concepto de web semántica.

La web semántica pretende facilitar la comprensión de cualquier página web, siempre que los documentos contengan información que podrá ser utilizada por un ordenador para comprender a lo que se refiere [107].

1.2. Contexto

1.2.1. Web semántica

Uno de los mayores usos que se le da a la web actualmente es el de la búsqueda de información. Los motores de búsqueda son los encargados de realizar esa acción encontrando diferentes recursos en toda la web [107]. El principal problema encontrado es la sobrecarga de información y la heterogeneidad de la misma, por ello, los motores de búsqueda no son todo lo exactos que deberían ser a la hora de ofrecer los resultados encontrados en las búsquedas.

Cualquier motor de búsqueda al realizar su trabajo, lo hace basándose en las palabras literales de la consulta sin tener en cuenta el significado de la palabra, por lo que devuelve numerosos resultados. Esos resultados son documentos encontrados en la web que han sido hallados por el motor de búsqueda, el cuál no tiene forma de comprenderlos y no puede

tomar ninguna decisión sobre qué hacer con ellos, por tanto, lo presenta al usuario de forma legible, y de alguna manera lo obliga a tener que realizar un filtrado manual para encontrar lo que necesita, lo que en ocasiones resulta costoso y molesto para el mismo.

La solución al problema descrito anteriormente es modificar los documentos que hay en la web, añadiéndoles información adicional que permitirá a los ordenadores entender el significado de los mismos. Suponiendo que estas modificaciones sean posibles se podrían construir herramientas, como los agentes [103], que fuesen capaces de procesar esos documentos a escala local. Nace así el concepto de web semántica.

La web semántica surge a principios de siglo, definida principalmente por Berners-Lee [14] que, pretendía enfocar la web hacia la idea que tenía cuando surgió la web tradicional: una web llena de significado donde los ordenadores fuesen capaces de utilizar la información facilitando al usuario las búsquedas de documentos.

Existen varias definiciones de lo que es la web semántica, el autor de la web semántica lo describe como: *"una extensión de la web actual en la cual la información es dada con un significado bien definido, estableciendo una mejor relación entre usuario y ordenador [...] Una web cuyos datos podrán procesar los ordenadores de forma directa e indirecta"* [15]. Por parte del grupo del W3C dedicado a estandarizar este sistema, la web semántica se define como: *"la idea de tener datos definidos y vinculados de manera que puedan ser utilizados por ordenadores no sólo de manera "visual" sino para integración, automatización y reutilización de datos a través de varias aplicaciones"*. El autor de la WWW espera que los ordenadores sean capaces no sólo de presentar información sino también de utilizarla. Esta es la idea en la que se fundamenta la web semántica, este es el siguiente paso de la web tradicional.

La web tradicional es capaz de ofrecer información solamente en dirección computador-hombre, el reto que supone la web semántica es que sea capaz de ofrecerla también en la dirección computador-computador. Esto daría lugar a un paradigma donde *"la fantasía del*

hombre y la lógica del ordenador coexistan en una combinación ideal y poderosa" [14]. La clave principal para la comunicación computador-computador es la semántica. Pero en la web semántica el concepto de semántica no está limitado por la definición lingüística ni por su conocimiento en informática, sino que es una extensión de ambos.

Según la W3C en la web semántica la información está bien definida, basándose en el significado y resolviendo así el problema de la web tradicional.

Para poder llevar a cabo lo idea de la estructuración de la web semántica es necesario añadir información que de significado en cualquier página web. Esa información es necesario añadirla con un lenguaje determinado para que el motor de búsqueda pueda interpretarlo correctamente. En la actualidad existen varios lenguajes para hacer esto, entre los más populares están: XML, OWL y RDF.

- XML (eXtensible Markup Language). Se trata de un lenguaje etiquetado destinado a definir la estructura de la información en una página web. El formato de etiquetado de XML permite definir elementos que podrán contener otros elementos, además también podrán contener propiedades. Un documento escrito con este lenguaje es válido siempre que cumpla una serie de reglas específicas que establecen las propiedades del documento. Estas reglas se definen en otro documento, en el caso de la web semántica ese documento suele ser en un formato denominado XML Schema, en este documento se establecen las entidades, elementos y atributos y la forma de combinación permitida para los mismos.

- OWL (Web Ontology Language). Es un lenguaje para programación de ontologías diseñado para utilizarse en la programación web, de manera que ofrezca la información que pueda ser procesada por otros sistemas. Este lenguaje ofrece más posibilidades para describir clases, axiomas de clases y propiedades.
  Contiene tres niveles de lenguaje [107]: OWL Full, la gran ventaja es que se dispone

de todo con total libertad, pero la gran desventaja es que el resultado puede ser una ontología muy potente de manera que sea demasiado costosa; OWL DL (Description Logic), se trata de un sublenguaje de OWL Full, la diferencia con éste, y quizás la desventaja, son las restricciones que le restan fluidez al modelo. Por otro lado, ofrece una gran expresividad, y el motor de razonamiento es fácil de construir por sí mismo; OWL Lite, que permite establecer relaciones jerárquicas pero tiene restricciones de cardinalidad (0 ó 1).

- RDF (Resource Description Framework). Se trata de un lenguaje creado por W3C para programar aplicaciones web [47]. Estas aplicaciones tienen una sintaxis común. Debido a que ha sido un lenguaje escrito pensando en su utilidad para la web, tiene en cuenta la descripción de sus recursos. Entendemos como recurso cualquier unidad de la que se pueda dar información. Este lenguaje está descrito por tripletas compuestas por: sujeto, predicado y objeto, lo que hace que la información pueda ser mejor comprendida por un ordenador, ya que cada recurso tendrá una descripción con una tripleta dando información adicional sobre él mismo.

1.2.2. Ontologías

Para llevar a cabo la idea de web semántica es necesario, como ya se ha dicho antes, añadir cierta información a la web, esa información se añade a través de ciertos lenguajes, algunos de ellos ya han sido descritos anteriormente y entre estos hay alguno específico para crear ontologías.

Según Borst [19], una ontología es una especificación formal de una conceptualización compartida, esto significa que define un área de conocimiento específico. Para ello se define un grupo de términos y las relaciones que existen entre ellos de manera que el dominio de conocimiento se hace comprensible para un ordenador.

Según Geneserth and Nilsson [33], una conceptualización es una vista simplificada y abstracta del mundo que se desea representar para algún propósito. En nuestro caso, este propósito es dotar de información adicional a la web para que pueda ser procesada por ordenadores.

Una ontología se define principalmente gracias a clases y relaciones, pero hay más partes que se verán de manera detallada en capítulos posteriores.

1.2.3. Servicios web

En su origen la web suponía un protocolo de acceso a objetos que facilitaba el acceso a los mismos ya fuese de manera local o remota. En la actualidad, los servicios web, proporcionan la idea de interfaz facilitando el acceso a los recursos que existen en la web [29].

Se puede decir que un servicio web es un conjunto de protocolos que simplifican el intercambio de datos entre aplicaciones. Para la ejecución de este recurso (o conjunto de recursos) es necesario disponer de una serie de protocolos sobre los que se sustenta un servicio web, éstos son utilizados para intercambiar la información entre aplicaciones independientemente del lenguaje de programación que utilicen éstas.

Los estándares más utilizados en los servicios web son SOAP y WSDL:

- SOAP (Simple Object Access Protocol). Se trata de un estándar que permite la comunicación entre objetos de diferentes procesos que además están en distintos ordenadores [79]. Este protocolo facilita la interoperabilidad. Cuando se recibe una petición y un servicio web ha de dar respuesta, ésta habitualmente está escrita en lenguaje XML y SOAP, este último es el que contiene la información necesaria para que el cliente interprete correctamente la respuesta ofrecida.
- WSDL (Web Services Description Language) es un lenguaje utilizado para describir

servicios web. Es independiente del lenguaje utilizado para la programación del servicio y del sistema operativo en el que se esté ejecutando [79]. Está basado en XML y contiene toda la información necesaria para contactar con un servicio web.

Básicamente, habrá que tener en cuenta que hay que conectar dos ordenadores y eso se hace mediante URL. Un servicio web funciona de manera muy parecida a una página web: es independiente del ordenador en el que se ejecute, el sistema operativo del cliente, etc. Pero también es cierto que hay que tener en cuenta que no todos los clientes que solicitan la ejecución del servicio web tendrán los mismos requerimientos [79].

¿Cuál es, entonces, la diferencia entre servicio web y sitio web? La respuesta proporcionada. Efectivamente, los servicios web no solo proporcionan a los usuarios una respuesta legible, también facilitan la comunicación entre aplicaciones permitiendo que otro servicio utilice las funcionalidades de éste y dar una respuesta más concreta al usuario. Es decir, la ventaja de los servicios web frente a los sitios web tradicionales es la interoperabilidad entre aplicaciones para ofrecer respuesta a la solicitud del usuario, en un capítulo posterior veremos la arquitectura de los servicios web de manera más detallada.

1.2.4. Terminología

La Terminología es una materia de carácter interdisciplinar basada en la lingüística aplicada, la lógica, la teoría de la información, la comunicación y las disciplinas objeto de estudio que se encarga del establecimiento, análisis, ordenación y descripción de los términos de una especialidad [20]. Cuando hablamos de terminología se puede estar hablando de la disciplina que trata las expresiones de especialidad, de la metodología que se encarga de extaer esas expresiones o del conjunto de términos que forman el léxico de una materia específica. Gracias a la terminología se pueden normalizar las denominaciones.

A principios de los 80, la informática era utilizada en la terminología para almacenar y recuperar información especializada [105], pero gracias a la evolución de las tecnologías, en la actualidad, el papel de los ordenadores en la terminología se ha convertido en fundamental, ya que las herramientas informáticas se utilizan también para extraer y clasificar el conocimiento. Esta clasificación es posible gracias a la unión de las herramientas informáticas, el proceso de elaboración y representación de la terminología.

Este proyecto surge ante la necesidad de vincular: la web semántica, las ontologías, los servicios web y la Terminología.

Para poder dotar de significado semántico a la web semántica es necesario dotar a los documentos de la web de significado; una ontología es la encargada de realizar esa dotación. Cuando una página web está compuesta por una serie de metadatos o incluso de una ontología, cobra un significado determinado; es así como se está dotando a la página de sentido semántico convirtiéndola en una página web ideal para la web semántica.

Por otro lado, para poder realizar esa ontología es necesario conocer cada uno de los términos que la van a componer, definiendo de manera clara y precisa el significado de cada uno de ellos. Este proceso no sería posible si no se conoce la terminología del campo que se va a tratar, y por ello es necesario tener muy claros los conceptos que se van a definir en esa página web, para no dar cabida a la definición de la ontología.

Para finalizar los servicios web facilitan la comunicación entre el cliente y la página web, obteniendo la información solicitada por el cliente, con independencia de la plataforma utilizada para buscarla.

1.3. Propósito y objetivos

En esta investigación se pretende realizar un estado del arte de ontologías y, en menor medida, de servicios web explicando a su vez la relación entre estos y la terminología. Proponiendo después una posible ontología aplicada a la definición de la calidad del aceite de oliva, como ejemplo ante la emergente necesidad de definir claramente en qué contexto se habla en la WWW.

Por lo antes comentado, los objetivos definidos para este proyecto son los siguientes:

- Elaboración del estado del arte sobre las ontologías. Definición, historia y aplicaciones de las ontologías.
- Revisión de la temática de servicios web y aplicaciones en el ámbito del proyecto.
- Exposición de terminología basada en corpus aplicada al propósito del proyecto.
- Integración de ontologías y servicios web, comentando los trabajos relacionados.
- Propuesta de la ontología para la descripción de la calidad del aceite de oliva, explicando de manera detallada su análisis, diseño, desarrollo e implementación.

1.4. Estructura de la memoria

Una vez que el lector está en situación, se presenta la estructura que sigue la memoria para la correcta realización de la investigación:

- Capítulo 2: en este capítulo se describen las tecnologías utilizadas. Para la mejor comprensión se describen detenidamente las ontologías, haciendo un repaso sobre su definición, historia, clasificación y posibles aplicaciones; los servicios web, en los que se

define, se explica la arquitectura que siguen y se comentan los retos y posibles aplicaciones; y terminología basada en corpus comentando le que es, la metodología que se sigue al utilizarla y su aplicación para este proyecto.

- Capítulo 3: En este capítulo se explica detalladamente cómo se relacionan dos de los conceptos explicados en el capítulo anterior las ontologías y servicios web, exponiendo algunos trabajos relacionados, así como la propuesta de la ontología y el análisis, diseño e implementación de la misma.

- Capítulo 4: Se exponen las conclusiones extraídas del proceso de investigación así como algunas propuestas para un trabajo futuro.

Capítulo 2

Ontologías, servicios web y terminología basada en corpus

Este capítulo, dividido claramente en tres partes, trata de dar al lector una visión más precisa de las disciplinas que se han investigado en el proyecto.

La primera parte del capítulo trata de definir el estado del arte para las ontologías. Por ello se define primero qué es una ontología. Una vez definido se realiza un breve repaso a la historia de las ontologías, describiendo las metodologías y lenguajes utilizados a la hora de implementarla. Posteriormente se explica la clasificación que se le puede dar a una ontología. Esta parte del capítulo finaliza con las aplicaciones que se le da a esta tecnología.

La segunda parte del capítulo trata de servicios web. En ella se pretende acercar al lector a un mejor entendimiento de esta tecnología, por ello, se describe en primer lugar lo que es, en segundo lugar la arquitectura utilizada para su desarrollo y en tercer lugar los retos que supone y qué aplicaciones tiene.

Por último, en la parte final del capítulo, se define qué es la terminología basada en corpus y se hace una descripción de la metodología que utiliza esta disciplina. Para finalizar se describe de manera más detallada como se ha procedido a la realización de la terminología

utilizada para este proyecto.

2.1. Ontologías

En este trabajo la ontología toma un papel fundamental, es el tema principal que ha ocupado la investigación, pero para poder definir y hacer uso de esta ontología es necesario tener claros algunos conceptos detallados a continuación.

Para diseñar y desarrollar una ontología correctamente será necesario saber exactamente qué es, para qué se define y, por supuesto, cómo se hace.

2.1.1. Definición

Pero, ¿qué es exactamente una ontología? Depende de la disciplina en la que estemos hablando una ontología puede tener diferentes significados. El concepto ontología nace de la filosofía y significa descripción del ente. Según Gruber [41] y Guarino [44] se define ontología como una conceptualización de un mundo concreto. Para el ámbito computacional dicha definición ha de ser legible para un ordenador, es en ese momento cuando la ontología entra en relación con los conceptos de "representación del conocimiento", "razonamiento inductivo" y "lingüística computacional".

Entendemos como representación del conocimiento aquella que realiza inferencia a partir del razonamiento. Según Sowa [93], se trata de una combinación de ontologías, lógica, bases de datos y sistemas orientados a objetos, en los que a través de reglas y lenguajes determinados se puede llevar a cabo un razonamiento inductivo.

El razonamiento inductivo es posible gracias a la definición previa de premisas y reglas. Merced a éstas y a un proceso de inferencia se obtiene el resultado deseado.

En cuanto a la lingüística computacional, es la disciplina encargada de estudiar la lingüística en el ámbito de la informática, en otras palabras, es una combinación de ambas. En este proyecto toma una especial relevancia a la hora de realizar la ontología debido al ámbito que se propone. La lingüística computacional nos ayudará a describir los conceptos que tendrán cabida en nuestra ontología.

Si es cierto que debido al concepto de ontología pueden existir varias formas de ver un mismo dominio, aunque no es lo más habitual, de ahí la importancia de la lingüística en el trabajo. Es gracias al conocimiento que almacena la ontología que es posible extraer datos y, por tanto, conocimiento de este tipo de herramientas. Hay muchas ventajas en la construcción de ontologías, y entre ellas están: la reutilización de la información, la representación explícita o la clarificación de la estructura de conocimiento.

Las ontologías son herramientas utilizadas para la representación del conocimiento. Dicho conocimiento es normalmente recopilado a través de la web. Cuando se unen los conceptos de web y conocimiento surge el concepto de web semántica. Cuando un usuario visita la web en búsqueda de información no siempre encuentra lo que busca, y la mayoría de las veces ha de hacer un filtrado manual a través de los resultados devueltos por el navegador. El navegador realiza la búsqueda basándose en las palabras que el usuario ha introducido, no en la semántica de las mismas y en muchas ocasiones los resultados pueden abarcar varios ámbitos, obligando al usuario a realizar ese filtrado manual del que hablábamos. Esto se debe a la cantidad de información que existe en la red hoy día. Por ello surge el concepto de ontología. Una ontología es definida para facilitar la búsqueda de información y lo que es más importante aún, para saber que los resultados obtenidos tienen exactamente el significado que el usuario estaba buscando.

Una vez que sabemos qué es una ontología y por qué surge, vamos a detallar cómo ha de definirse.

Cuando se diseña una ontología es necesario tener unos buenos principios que serán unos u otros según el resultado final deseado. Por lo general, una ontología debe seguir los siguientes principios:

- Toda ontología ha de ser **clara** [42]. Los términos definidos en la ontología han de estar documentados en lenguaje natural. Aunque una ontología define una conceptualización de un mundo, debe ser independiente de requerimientos sociales y computacionales, es decir, debe ser objetiva.

- Debe ser **coherente** [42]. Para que la ontología no sea incoherente la inferencia extraída y la definición de los términos han de ser consistentes, esto será posible si tanto términos como reglas están bien definidos. En el momento en que una sentencia es extraída de axiomas contradictorios la ontología perderá su coherencia.

- **Extensible** [42]. El principal motivo por el que se diseña una ontología es para organizar y compartir el conocimiento. Por ello, cualquier ontología diseñada deberá ser extensible por otros usuarios de manera sencilla y sin que tenga la necesidad de estar en constante revisión.

- El **mínimo compromiso ontológico** [42] se basa en diseñar la menor cantidad posible de suposiciones del mundo a modelar. La ontología ha de dar la mínima información siempre que sea suficiente para modelar el mundo, permitiendo a las partes comprometidas la libertad de instanciarlas y especializarlas tanto como sea necesario.

- Todas las clases correspondientes a diferentes criterios de identidad han de ser disjuntas, esto es lo que defiende el **principio de distinción ontológica** [18]. El criterio de identidad ha de ser fijado previamente, para así poder determinar distinciones ontológicas.

- Si se utilizan criterios de clasificación muy diferentes, será más sencillo especificar

nuevos conceptos a partir de lo ya definido y se potenciará la herencia múltiple, esto es lo que Arpírez [10] denomina **diversificación de las jerarquías**.

- La **especialización** de conceptos, dentro de los límites definidos previamente, facilita que se clasifiquen aquellos con características similares y se garantice de esa manera su herencia [10].

- Los conceptos similares se agrupan presentándose en subclases y agrupados en clases, los menos parecidos se sitúan más lejos de la jerarquía. Esto es lo que Arpírez [10] denomina **mínima distancia semántica**. Además, los conceptos emparentados (más similares) contaran con las mismas primitivas (gracias a la herencia).

- **Estandarización de nombres**. Las relaciones, siempre que sea posible, se denominarán con el nombre de la ontología (o el del primer concepto) concatenado con el de la relación y el del concepto de destino [10].

- **Modularidad**. Se debe minimizar el acoplamiento entre módulos para así permitir una mayor flexibilidad entre los nuevos y los ya diseñados [10, 12].

Estos son los principios de una ontología, pero para diseñarla también es necesario tener en cuenta que cualquier ontología debe contar con unos componentes mínimos. Hay varias visiones de los componentes necesarios para una ontología. Vamos a definir cuáles son los componentes de los que debe constar y quién señala que es apto dicho componente.

Según Gruber [42], ha de formalizarse el conocimiento en cinco tipos diferentes de componentes que son: clases, relaciones, funciones, axiomas e instancias.

Gómez-Pérez asegura [35] que una ontología consta de conceptos, relaciones, funciones, instancias y axiomas, y además añade que los conceptos son organizados en taxonomías, algo que también comentaba Lord [94].

Fensel et al. apuntan que los componentes de una ontología son: conceptos, relaciones y funciones, instancias de conceptos y axiomas [29].

Como se h visto, la mayoría de los autores de la bibliografía especializada coinciden en determinados componentes, que son detallados a continuación:

- **Clases**, también son denominadas conceptos. Son las ideas básicas de una ontología. Se trata de un conjunto de individuos con similares características que se agrupan en una misma clase. Algunos autores definen también subclase, pero esta no deja de ser una clase dentro de otra, es en realidad el mismo concepto pero agrupado un nivel más de especialización. Genesereth y Nilsson [33] identifican a las clases como universo de discurso. El universo de discurso no es más que un conjunto de objetos representados de los que se puede hablar y razonar.

- Las **relaciones** son la manera que tienen los individuos de las clases de relacionarse unos con otros. Hay varias formas de expresar una relación: entre individuos concretos o entre clases. Normalmente las relaciones son las que facilitan la formación de la taxonomía del dominio. Toda relación depende de 0 o más relaciones, en caso de ser una subrelación de otra, hereda las características de las superrelaciones a las que pertenece. Algunos ejemplos de relaciones son: parte-de, tipo-de, subclase-de, conectado-a, conectado-con, etc. Cabe comentar que algunas de las posibles relaciones dependen, en parte, del lenguaje en el que se programe la ontología [63].

- **Funciones**. Las funciones son un tipo especial de relación. Se utilizan para identificar un elemento único que es calculado a partir de varios elementos de la ontología [10].

- Una **instancia** es un objeto determinado de una clase que está descrito por características particulares.

- Los **axiomas** son sentencias que siempre son verdaderas y se utilizan para mejorar los conceptos, relaciones y funciones. En una ontología se utilizan los axiomas cuando el

conocimiento no puede ser expresado por ninguno de los componentes anteriores. Las instancias y clases han de cumplir el axioma que se refiera a ellas. Los axiomas facilitan la inferencia en una ontología.

De manera aclaratoria vamos a ver un ejemplo de cada uno de los componentes anteriores:

Supongamos una ontología que pretenda definir la cata de los aceites de oliva. Tendremos todos los componentes que se han mencionado antes.

Clases: cata, sabor, flavor, textura, atributos positivos, atributos negativos.

Instancias: untuoso, amargo, picante, frutado, almendrado, gusano, metálico, henomanera.

Relaciones: la relación establecida entre "cata" y "amargo" es que *amargo* es un *atributo positivo* que define un *sabor* en la *cata* de un aceite.

Funciones: para poder determinar cómo es un aceite es necesario hacer una cata del mismo, y para ello se tienen que categorizar los atributos que se distingan en las instancias definidas. (Un aceite tiene sabor amargo, textura untuosa, etc., es decir, viene definido por las categorías de nuestra ontología).

Axiomas: "Si es *atributo positivo* no puede ser *atributo negativo*". El *atributo positivo* denominado *almendrado* no podrá ser *atributo negativo*.

Sin embargo, según otros especialistas [77, 93], el alcance o dominio de la ontología puede motivar la definición de más componentes, comentados a continuación:

- **Subclase**: como se ha dicho anteriormente una clase puede ser hija de otra clase, convirtiéndose así en subclase de la anterior.

- **Clase jerárquica**: según la relación que haya entre la clase que se está tratando y la clase padre, la relación entre ellas puede ser jerárquica, esto ocurre si se enlazan por relaciones "tipo-de".

- **Roles**, también denominados propiedades, atributos o slots. Se encargan de describir un concepto, gracias a los roles se precisan las clases.

- **Facetas**, también denominadas restricciones de los roles, utilizadas para encuadrar qué valores se permite dar a esos roles.

- **Valores**, definen las propiedades que tiene una instancia o una clase.

- **Tipo**: determina el tipo de valor, es decir, booleano, cadena, entero...

- Las relaciones unen conceptos. Estas relaciones han de especificar su **cardinalidad**, de manera que se diga cuántos elementos de cada clase es posible instanciar.

- Gracias a la **herencia** las subclases e instancias toman las propiedades, facetas y valores de su clase padre.

- Las **variables** son espacios que pueden rellenarse preguntando a clases e instancias. Todas las variables comienzan por "?".

Cuando se define una ontología es muy importante determinar el alcance de la misma. Para ello, lo mejor es redactar una serie de preguntas previamente para adaptarse a la definición de la misma, así como definir un flujo de trabajo que facilita su diseño, es decir, es recomendable establecer cuál es exactamente el dominio, para qué se utilizará, quién la mantendrá y quién la usará. Estas son las preguntas, a grandes rasgos que se tienen que tener claras antes de comenzar a diseñar una ontología. Sin embargo, de manera más concreta se puede realizar un cuestionario para determinar cuál será el dominio que alcanzará. En la parte de diseño se definirá este cuestionario para más adelante darle respuesta.

Es necesario, antes de diseñar nada, comprobar si es posible reutilizar una ontología ya definida, en caso afirmativo, ya se tendrá un largo camino recorrido y, además, si la ontología a reutilizar está bien diseñada no será difícil ampliarla al contexto que se desea. En secciones posteriores se dan algunos diseños públicos en los que se podrán encontrar bancos de ontologías y comprobar en ellas si es posible reutilizar alguna ontología ya diseñada.

Es necesario, una vez diseñada e implementada la ontología, evaluarla. La forma de evaluar la calidad de una ontología puede medirse a partir de una serie de preguntas que obtendrán respuesta en función de la diseñada, esas preguntas son, entre otras:

- ¿Contiene información suficiente para responder a las preguntas de competencia?
- ¿El nivel de detalle de las respuestas dadas es correcto?
- ¿Las respuestas requieren un nivel de representación especial?

Estas preguntas son orientativas, es decir, no es necesario que la ontología responda a ellas de manera particular y exhaustiva.

Es útil tener en cuenta que en el caso de que la ontología diseñada vaya a ser utilizada para procesamiento del lenguaje natural (PLN) se deberá definir sinónimos y demás información para un mejor aprovechamiento de la misma, ya que facilitará su lectura y comprensión, no obstante estas y otras cuestiones de diseño se verán en el capítulo destinado para ello en la presente memoria.

2.1.2. Historia

El término ontología es utilizado fundamentalmente en el campo de la filosofía. A lo largo del tiempo y a consecuencia del camino que ha llevado la representación de información, el

concepto de ontología cambió, siendo cada vez más común su uso en informática, concretamente en el área de la Inteligencia Artificial. Concretando un poco más podemos atribuirle un emergente significado en los campos de la representación del conocimiento, razonamiento inductivo y lingüística computacional.

Dentro del ámbito de la representación del conocimiento no sólo están las ontologías destinadas a este fin, existen otras metodologías como son los tesauros, que no entramos a comentar en profundidad en esta memoria.

Los tesauros son listas de términos que se utilizan para describir temas en documentos [32]. Tienen como fin indexar documentos, además de la recuperación de información.

Un fin distinto es el de la ontología que, como se ha dicho antes, pretende modelar un mundo concreto y para ello utiliza la definición de conceptos y clases. El ámbito que abarca la ontología es más amplio que el de los tesauros, pretendiendo dar un paso más en la representación del conocimiento realizando inferencia.

Para desarrollar una ontología cada grupo tiene sus propios fundamentos, normalmente se siguen distintas fases e incluso se implementan con diferente lenguaje. Al no existir un consenso para esta acción es difícil desarrollar ontologías que puedan compartirse de manera sencilla y por consiguiente ser integradas en las que se están desarrollando.

Si algo tienen en común los desarrolladores de ontologías es la carencia de fases intermedias a la hora del diseño, es habitual que se pase de la adquisición del conocimiento a la implementación de manera directa, sin haber estructurado el conocimiento adquirido. Esto favorece que existan problemas, por ejemplo, de esa forma el modelo conceptual solamente está definido dentro del código elegido para la implementación; para un usuario de la ontología no es posible determinar el dominio y el alcance; para dos desarrolladores de ontologías, si no conocen el lenguaje en el que está implementado es posible que no puedan comprender del todo la ontología y es por ello por lo que en ocasiones hay conceptos repetidos en

ontologías diferentes.

2.1.2.1. Metodologías

A continuación se exponen en orden cronológico las metodologías más interesantes utilizadas en el diseño de ontologías:

Cyc (1990)

Esta metodología es desarrollada gracias al Cyc Knowledge Base (KB). Cyc KB está construido sobre une base de conocimiento extraída a partir de afirmaciones diseñadas específicamente para ese propósito. Cyc KB se expresa en CycL y sigue el esquema representado en la figura 2.1 [60].

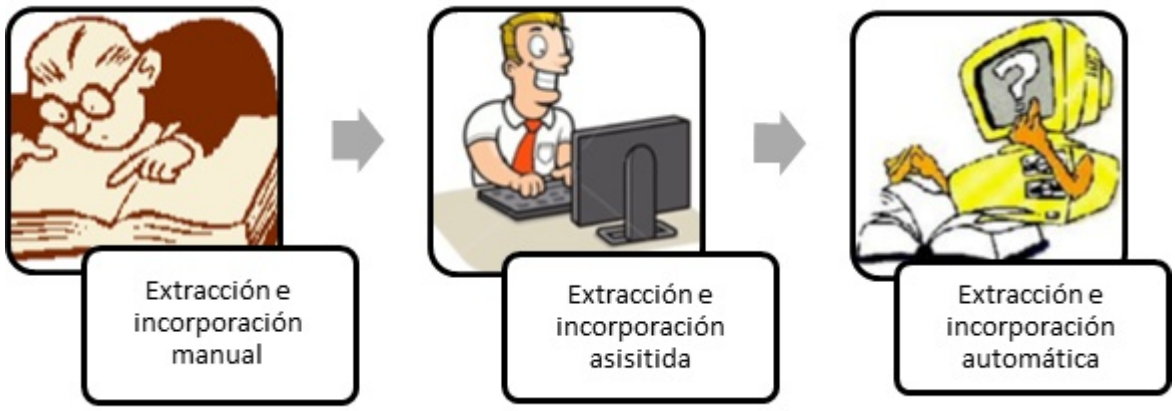

Figura 2.1: Fases para la metodología Cyc

Extracción e incorporación de información de manera manual: en esta fase se codifica el conocimiento implícito y explícito manualmente. No son utilizados sistemas de aprendizaje ni procesamiento del lenguaje natural (PLN).

Extracción e incorporación de información de manera asistida: codificación del conocimiento de manera automática, ayudado por el que se ha almacenado en Cyc KB.

Esta fase es una mezcla de la primera y tercera fase de la metodología, ya que combina el trabajo manual con el trabajo computacional.

Extracción e incorporación de información de manera automática: por lo general, en los documentos que se recopilan, se suele recomendar la lectura de las fuentes y se explica la parte más compleja, es ese el motivo de que en esta fase se delegue el trabajo en las herramientas computacionales.

Si es cierto que cada una de las fases comentadas cuenta con dos tareas esenciales:

Desarrollar una representación de conocimiento y una ontología de alto nivel que serán los que contengan los conceptos abstractos.

Representar el conocimiento. Para esta tarea se utiliza HPKB (High Performance Knowledge Bases). Se trata de un programa de investigación para la adquisición, manipulación y representación de conocimiento.

Cabe destacar el hecho de que hasta principios de siglo solo había sido utilizado de manera completa en el seno del proyecto Cyc, aunque si contiene microteorías que han sido utilizadas para representar el conocimiento de un mismo dominio desde diferentes puntos de vista.

Gruninger y Fox (1995)

Gruninger define ontología como una descripción formal de entidades con sus propiedades, relaciones, restricciones y comportamientos [40]. Esta metodología está basada en el lenguaje PROLOG y en lógica de primer orden.

Basándose en el proyecto TOVE se toma el enfoque inicial de la ontología en el panorama industrial, siendo la mayoría del esfuerzo utilizado para la creación de representaciones.

Haciendo un esquema de ella se pueden observar las fases descritas en el esquema de la figura 2.2. Vamos a describir brevemente cada una de las fases:

Figura 2.2: Fases para la metodología Gruninger y Fox

Escenario motivador: normalmente el escenario viene presentado por problemas que los propios compañeros se encargan de presentar. Tienen forma de historias, cosas que han ocurrido y que no están adecuadas a la situación actual. Cada escenario motivador tiene su propio conjunto de soluciones. Éstas dan una primera idea de la semántica que será incluida en la ontología. Cualquier proposición de ontología debe describir los escenarios motivadores y el conjunto de soluciones.

Preguntas informales de competencia: una vez que se ha descrito el contexto en el que se desea diseñar la ontología, surgen un conjunto de preguntas que se responderán con la ontología subyacente y que son los requerimientos. Estas preguntas son denominadas de esta manera hasta que se expresen en un lenguaje ontológico formal. Por lo general han de ser solucionadas de manera jerárquica, así, una pregunta de alto nivel podrá ser respondida, además de por su propia respuesta, por una respuesta a una pregunta de nivel bajo.

Especificación de la lógica de primer orden: Terminología. Una vez definidas las preguntas informales de competencia, éstas han de expresarse en lógica de primer orden. Para ello se debe proporcionar la termología necesaria para contestarlas. Para una ontología nueva, cada pregunta será respondida por un nuevo elemento. Lo principal para definir la terminología es identificar los objetos en el dominio de discurso.

Preguntas formales de competencia, ahora es el momento de definir las preguntas

de competencia de manera formal. Estas preguntas restringen los axiomas que serán definidos posteriormente. Todos los términos que se utilicen en esas preguntas han de estar definidos en la ontología. Cualquier ontología, ya sea nueva o una extensión, ha de tener un conjunto de preguntas formales de competencia.

Especificación en lógica de primer orden: Axiomas. Los axiomas definen los términos de manera más específica. Esta es la parte más tediosa, y resulta más sencilla gracias a las preguntas formales de competencia. Los axiomas son los encargados de contestar las preguntas de competencia (formales e informales) y sin ellos no podrá ser representada la solución a cualquiera de ellas. De manera iterativa se irá comprobando si los axiomas contestan a las preguntas, en caso de que no sea así será necesario incluir más objetos o axiomas a la ontología. Cabe destacar que los axiomas no son generados gracias a las preguntas, pero si son capaces de evaluar la expresividad de nuestra ontología gracias a ellas.

Completitud de teoremas. Una vez que las preguntas se han declarado de manera formal se definen las condiciones bajo las que las soluciones están completas.

Unschold y King (1995)

Está basada en la construcción de Enterprise Ontology [99]. Y propone lo expresado en la imagen 2.3. Se explican a continuación cada uno de los pasos:

Propósito: lo primero que se debe hacer es determinar por qué se construye la ontología y a quién va dirigida. En caso de realizar preguntas de competencia, pueden ayudar a definir el propósito de manera muy específica.

Construcción:

Captura: En esta fase se intenta definir qué es la ontología identificando los conceptos y relaciones más importantes en el dominio. Se desarrollan las definiciones de manera no ambigua y se identifica qué términos se refieren a cada relación y concepto.

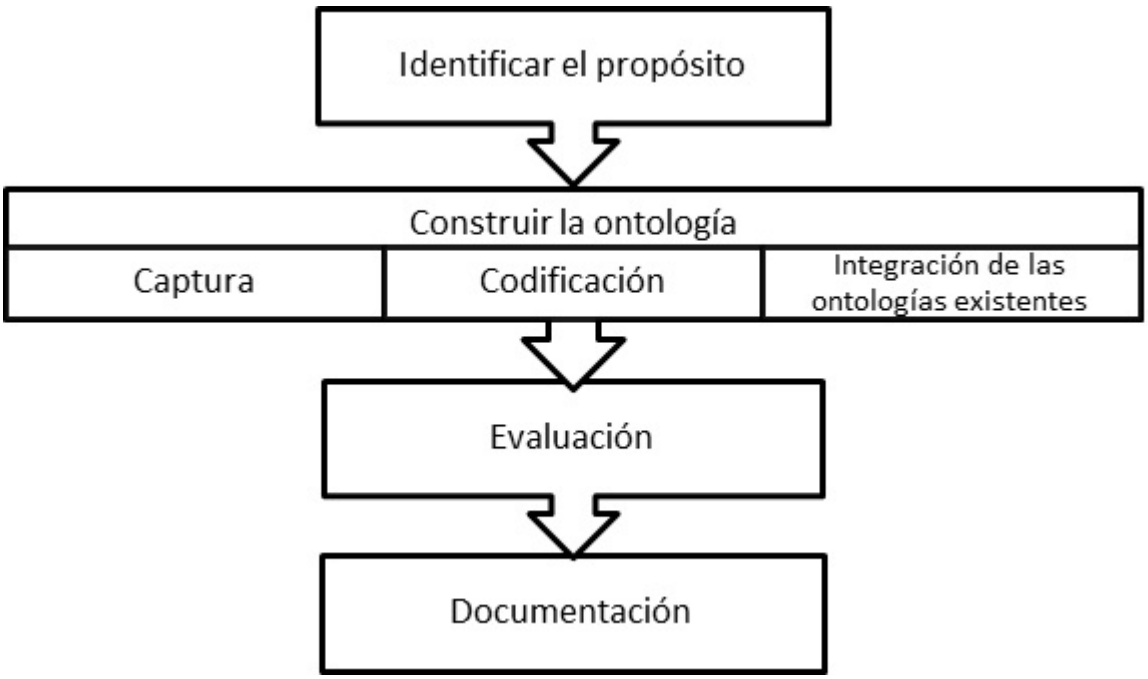

Figura 2.3: Fases para la metodología Unschold y King

Codificación de la ontología: una vez terminado el paso previo, es necesario traducir todo lo definido a un lenguaje formal. En ocasiones estas dos fases se llevan a cabo en un solo paso, aunque la experiencia resuelve que es mejor hacerlo por separado. Para la codificación los autores señalan que los criterios de Gruber [42] son fundamentales y deberían ser utilizados en cualquier metodología.

Integración de las ontologías existentes: este es un paso muy complicado. La orientación y herramientas en este sector no están muy desarrolladas aún.

Evaluación: este trabajo apuesta por echar un vistazo a la propuesta dada por la KBS y adaptarlo a la ontología. Para este juicio tiene en cuenta la visión de evaluación de una ontología dada por Gómez-Pérez [36].

Documentación: se destaca necesario establecer ciertas guías a la hora de elaborar la documentación de una ontología. Actualmente la mayor barrera para el intercambio de información es la documentación [92].

Kactus (1996)

El proyecto Kactus surge para observar la posibilidad de reutilización de un sistema de conocimiento y el apoyo a este de una ontología [13]. Está basado en aplicaciones. Cuando una aplicación se construye será necesario revisar la ontología, que puede haber sido construida reutilizando otras y podrá integrarse en las desarrolladas posteriormente.

Figura 2.4: Fases para la metodología Kactus

Se exponen a continuación las fases de las que consta esta metodología (figura 2.4):

Especificación de la aplicación: es conveniente dar un contexto de aplicación y de los componentes que van a utilizarse. Suele estar formada por una lista de términos y tareas.

Diseño preliminar basado en las categorías de alto nivel ontológico: se realizará una búsqueda de ontologías previamente diseñadas y reutilizadas en aplicaciones desarrolladas posteriormente. Gracias a ella se obtendrá una mejor visión de las categorías ontológicas de nivel superior.

Estructuración y refinamiento de la ontología: es recomendable que los módulos que se diseñen cuenten con un bajo acoplamiento y sean coherentes para permitir la máxima homogeneidad.

El proyecto Kactus está ilustrado por tres ontologías resultado del desarrollo del mismo número de aplicaciones.

Methontology (1996)

Desarrollada para la construcción de ontologías de conocimiento [34]. Está soportado por la herramienta ODE [16, 30] monousuario y WEB-ODE [11] en la que pueden trabajar varios usuarios.

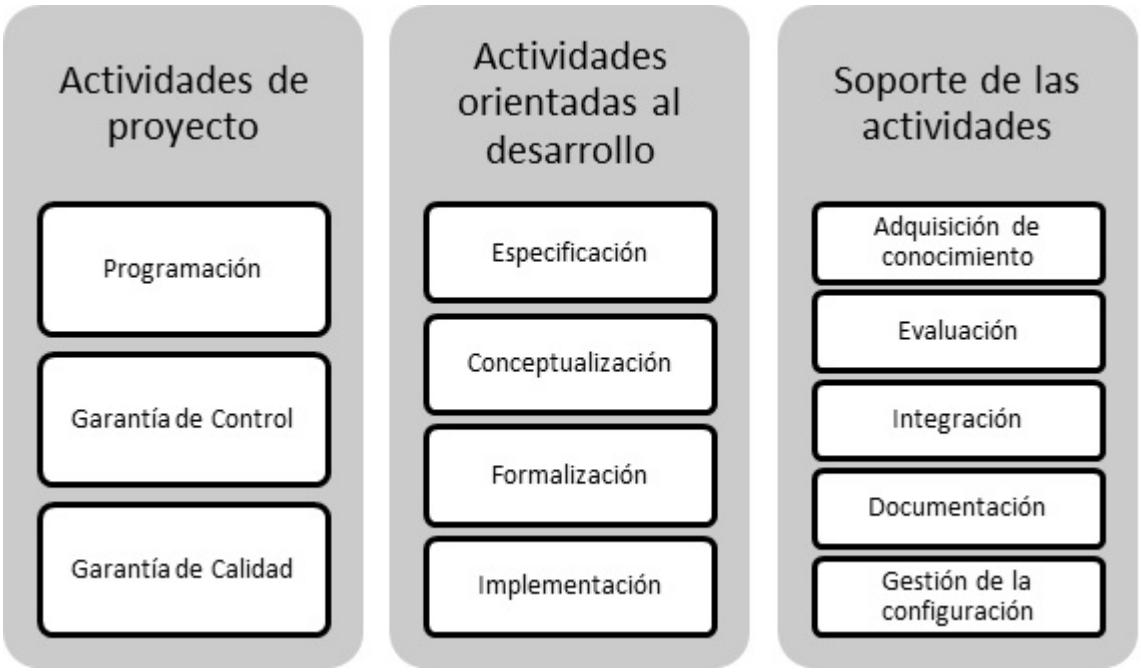

Figura 2.5: Fases para la metodología Methontology

Es importante determinar si la ontología a desarrollar se hará por equipos que estén cerca geográficamente, ya que en el caso de que no sea así es importante dejar claro cuáles son las actividades que han de realizarse y seguir las pautas marcadas a continuación (figura 2.5):

Actividades de proyecto. Dentro de esta fase podemos distinguir tres apartados:

Programación: es importante determinar qué tareas se van a llevar a cabo, cómo se harán, en cuánto tiempo y qué recursos serán necesarios. Esta fase cobra un especial interés en los casos en los que se ha de reutilizar otra ontología.

Garantizar el control: es necesario saber que cada tarea ha sido llevada a cabo y que

se ha hecho de la manera que estaba prevista.

Garantía de calidad: se asegurará de que los productos desarrollados tienen una buena calidad.

Actividades orientadas al desarrollo. En esta fase se incluyen:

Especificación: en este paso se establece un contexto de la ontología que se va a desarrollar especificando por qué se construye, para quién y qué uso se le va a dar.

Conceptualización: se encarga de estructurar el dominio de conocimiento de la ontología. Da una perspectiva del área que se pretende modelar.

Formalización: se encarga de transformar el modelo conceptualizado a un modelo formal, transcribiéndolo a un formato semicomputacional.

Implementación: en este último paso se construye el modelo computacional, traduciéndolo al lenguaje establecido previamente.

Asimismo es necesaria una etapa de mantenimiento en la que se corregirá y actualizará la ontología.

Soporte de las actividades. Esta fase es posible ejecutarla a la par que la anterior, es decir, incluye actividades que pueden realizarse al tiempo que otras de la fase de desarrollo. Incluye:

Adquisición de conocimiento: es necesario tener cierto control sobre el dominio que se está conceptualizando.

Evaluación: será necesario realizar una valoración, respecto a un marco de referencia, sobre las ontologías, software y documentación en cada fase de su ciclo de vida.

Integración: cabe la posibilidad de que sea necesaria una reutilización de ontologías,

por lo que sería inevitable una unificación entre la ontología desarrollada y otras disponibles.

Documentación: es impensable que no exista una fase que no esté detallada de manera concisa.

Gestión de la configuración: es altamente recomendable almacenar cada una de las versiones de software, documentación y ontologías para el control de cambios.

Cuando en la etapa de evaluación se habla del ciclo de vida, se refiere al conjunto de fases por el que pasa una ontología, describiendo cómo se relacionan las etapas y qué actividades se desarrollan en cada una.

Cabe la posibilidad, y está reflejada en esta metodología, de que las actividades desarrolladas a lo largo de la creación de la ontología reflejen cambios en la que se basa, si se da el caso. Si así fuese y se produjeran dichos cambios será necesario señalarlo y llevarlos a cabo. Este es el motivo por el que METHONTOLOGY está dentro del ciclo de vida de varias ontologías y posibilita la unificación de las políticas de desarrollo de las mismas.

Por último, señalar que METHONTOLOGY ha sido propuesta por la FIPA [8] (Foundation for Intelligent Physical Agents) para la construcción de ontologías.

Sensus (1997)

Este proyecto, pensado de manera más específica para el uso de los resultados en procesamiento del lenguaje natural (PLN), se sustenta en el uso de una ontología de grandes dimensiones para la construcción de otras más específicas y de bases de conocimiento. Su interés está apoyado en la facilidad para el intercambio de conocimiento, debido a que la misma base ontológica es utilizada para desarrollar otras más específicas. En la actualidad [96], cuenta con más de 50000 conceptos organizados categóricamente, extraídos de fuentes electrónicas y que no son de dominio específico sino de nivel medio-alto. Como veremos a continuación, cuando se construye una ontología de nivel específico con esta metodología,

los términos interesantes se agregan y los de nivel específico no son incorporados a la base de conocimiento. En la imagen 2.6 se observa el proceso a seguir [31, 96].

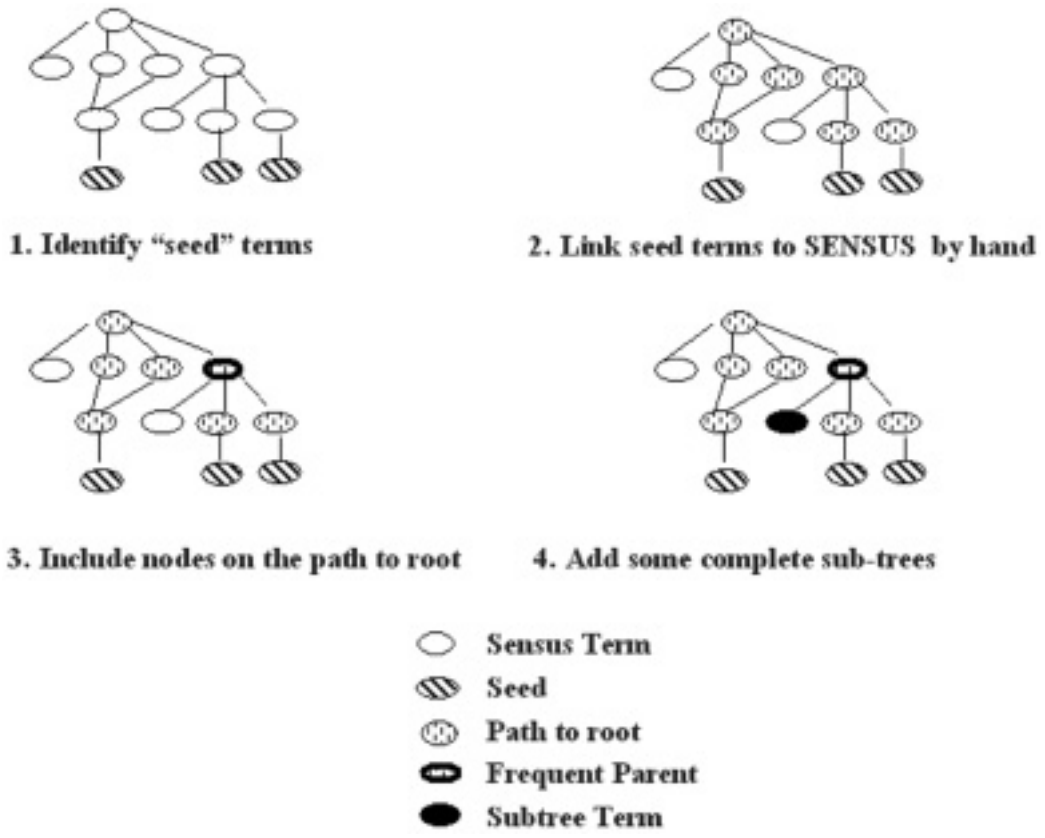

Figura 2.6: Fases para la metodología Sensus

Una vez descrito el dominio de aplicación de la ontología a crear, se señala un conjunto de **términos** que serán **tomados como semillas**.

Cada uno de los términos semilla se **vincula** de manera manual **a la base de SENSUS**.

Se analizan los resultados anteriores y se **añaden los términos** que pueden ser **interesantes para el dominio** y que aún no están en él.

Una vez añadidos esos términos se estudian los caminos de cada nodo a través de ellos. En los que se tenga un gran número de caminos el **subárbol es añadido completo**. Se añade

completo basándose en la idea de que si muchos nodos de un subárbol se han encontrado relevantes, es posible que los demás también lo sean. Para llevar a cabo este proceso, es necesario tener un alto conocimiento del dominio que se está tratando, por ello se requiere hacerlo de manera manual asegurando las decisiones tomadas.

Por último se **añaden los términos** definidos en este proceso a la **base completa**.

On-to-knowledge(2000)

Este proyecto diseñado para la mejora de la gestión del conocimiento en grandes empresas, promueve las tareas de soporte de adquisición, acceso a la información y mantenimiento. Desarrolla tanto la metodología como las herramientas necesarias para al acceso inteligente a grandes volúmenes de información. Consta de las fases reflejadas en la figura 2.7 [89].

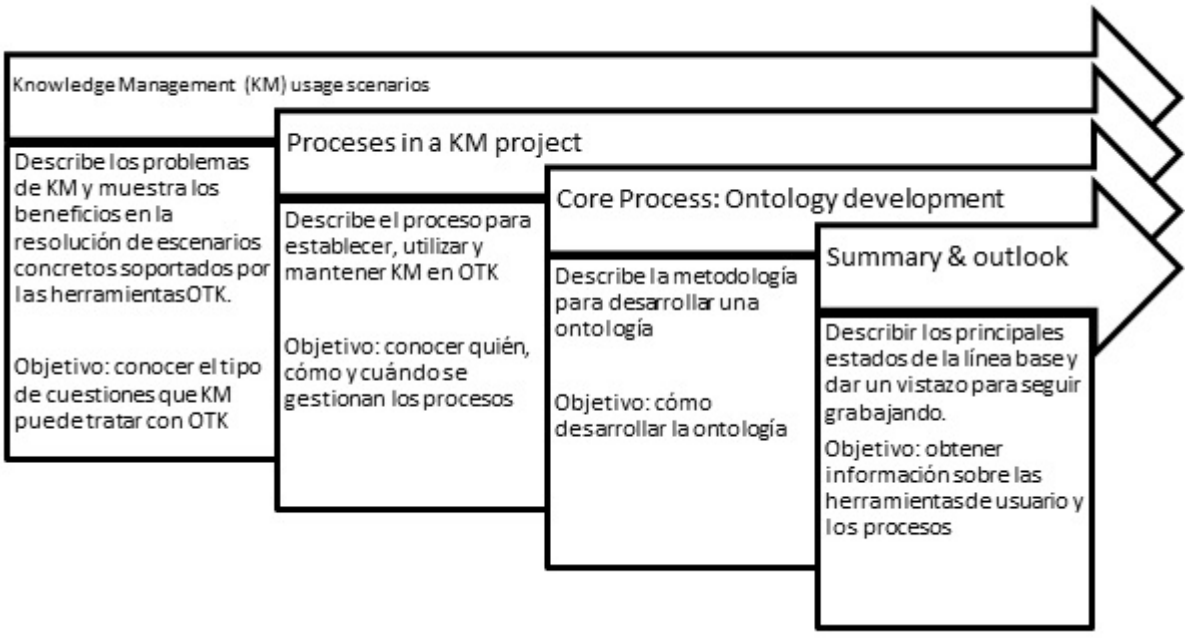

Figura 2.7: Fases para la metodología On-to-knowledge

Para llevar a cabo un proyecto OTK es recomendable diferenciar los casos de uso obtenidos a partir de un estudio de viabilidad que ayudará a:

Determinar los **usuarios del sistema** y los **mantenedores del sistema**.

Describir los escenarios denominados **casos de uso controlados por el usuario**.

Determinar los casos de **uso de apoyo** que serán aquellos que mantienen a los anteriores.

De manera más precisa, se puede definir la metodología siguiendo los siguiente pasos:

- *Identificación del propósito.* Determina qué es la ontología, para qué se utilizará, quiénes serán los usuarios, quién la mantendrá, etc.
- *Construcción de la ontología.* Se divide en tres pasos: 1) Capturar la ontología. Trata de definir la disciplina en la que se enmarca, los conceptos clave y las relaciones entre ellos. Es necesario que las definiciones que se desarrollen no sean ambiguas e identifiquen claramente a qué concepto se refieren. 2) Una vez definidos los conceptos será imprescindible codificar la ontología a un lenguaje computacionable formal. 3) En caso de que fuera necesario se tendrá que integrar las ontologías existentes.
- *Evaluación.* A partir de un conjunto de preguntas de competencia que se encargarán de definir los requerimientos de la ontología, será necesario llevar a cabo una evaluación de la misma.
- *Documentación.* Esta fase es imprescindible. Es necesario que toda ontología cuente con una buena documentación que deje claro su propósito y su tipo.

Cabe destacar que en esta metodología no se pretende especificar las técnicas necesarias para definir los conceptos y relaciones.

Ontology 101 development process (2005)

Parte de la idea de que no existe una metodología correcta para el desarrollo de ontologías priorizando la simplicidad. Se basa en la experiencia de los desarrolladores de la misma en el

uso de Protégé, Ontolingua y Chimaera, en el diseño orientado a objetos. Trata de describir una metodología que facilite el proceso a los nuevos desarrolladores de ontologías. En la descripción de la metodología se comenta que cualquier ontología es desarrollada de manera iterativa [76].

En el presente procedimiento se parte de un vistazo frontal de la ontología de manera iterativa se establece un proceso evolutivo que termina detallando la ontología.

Consta de las etapas descritas en la figura 2.8.

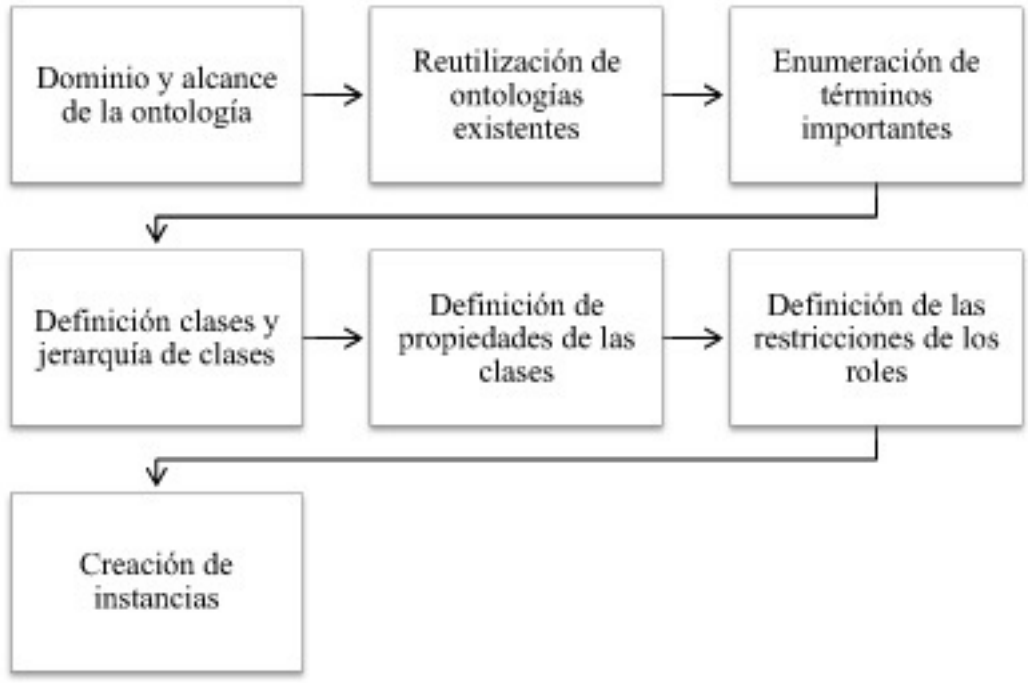

Figura 2.8: Fases para la metodología Ontology 101 development process

1. **Determinación del dominio y alcance de la ontología.** Esta etapa abarca las preguntas de competencia que ayudarán a acotar el alcance del domino.

2. **Consideración de la reutilización de ontologías existentes.** La naturaleza de la ontología es el conocimiento compartido. Lo ideal cuando se desarrolla una es poder

reutilizarla en algún momento.

3. **Enumeración de términos importantes para la ontología.** Expresar los términos que son interesantes para explicar.

4. **Definición de las clases y la jerarquía de clases.** Existen varios enfoques a tener en cuenta cuando se desarrolla una ontología: *top-down:* desde los conceptos más genéricos a los más especializados; *bottom-up:* desde conceptos específicos hasta los más genéricos; *combinado:* coordina los dos enfoques anteriores.

5. **Definición de las propiedades de las clases: slots.** En esta etapa se describen la estructura de los conceptos. De la lista descrita en el paso 3 se debe determinar qué clase está descrita por la propiedad.

6. **Definición de las facetas de los slots.** Describen los valores de los slots (las restricciones de los roles): cardinalidad, tipo de valor, dominio y rango entre otras.

7. **Creación de instancias.** En esta etapa se definen las instancias individuales.

Cabe destacar que los pasos 4 y 5 están estrechamente relacionados y son difíciles de separar.

Es posible deducir de lo explicado anteriormente todas las metodologías tienen en común la descripción del propósito y alcance de la ontología, así como la necesidad del desarrollador de tener un conocimiento amplio de la materia que está tratando. Otro aspecto común de las metodologías descritas es la fase de evaluación de la ontología, aunque sobre esto no hay ningún acuerdo.

2.1.2.2. Lenguajes

Como se puede deducir del estudio de las metodologías, es posible utilizar diferentes lenguajes a la hora de representar una ontología, incluso algunas metodologías proponen su propio lenguaje. Según el experto que la diseñe, la metodología seleccionada y por tanto el enfoque que se desee seguir (basado en lógica, en marcos o ambos) se escribirá en uno u otro lenguaje. Entre los más utilizados se encuentran:

XML (eXtensible Markup Language, 1996)[2]

Creado por XML Core Working[1] y XML Activity[2] dentro del W3C. Es un perfil de aplicación del lenguaje SGML (Standard Generalized Markup Language), caracterizándose entre otras cosas por ser menos restrictivo, que ofrece la comunicación entre SGML y HTML. Se compone de un conjunto de objetos y el comportamiento de los mismos cuando son procesados computacionalmente, dichos objetos son denominados documentos XML. Construidos por unidades completas denominadas entidades, los documentos XML contienen datos compuestos por caracteres y marcadores que se encargan de codificar el documento para que pueda ser analizado. Estos datos son analizados por un procesador de XML que, junto con una aplicación, permite acceder a la estructura del documento y a sus datos. Este lenguaje pretende ser idóneo para su uso en internet, soportado en la mayoría de las aplicaciones, compatible con SGML, de diseño rápido, formal y conciso, además, es fácil desarrollar programas que procesen documentos escritos en XML y es legible por un humano.

La sintaxis del lenguaje puede ser consultada en la página del W3C[3].

SHOE (Simple HTML Ontology Extensions, 1996)[49]

Este lenguaje nace como una extensión del lenguaje HTML Simple, se trata del primer

[1]http://www.w3.org/XML/Core/
[2]http://www.w3.org/XML/Activity
[3]http://www.w3c.org/TR/xml11

lenguaje creado para el diseño de ontologías y pretende facilitar las anotaciones que consten de semántica en las webs de los autores y, por consiguiente, una mejora en la representación del conocimiento. Dichas anotaciones son expresadas y pueden ser generadas gracias a estándares ontológicos.

Se desarrolla un superconjunto de HTML que ofrece mecanismos para dotar de semántica a la web. SHOE facilita: la creación de ontologías extendiendo las existentes y su definición utilizando HTML; la declaración de entidades, declaración de atributos en ellas y relaciones entre las mismas; y como la clasificación bajo relaciones "is a". La intención de SHOE es un punto medio entre la consulta de grandes cuerpos de información distribuida y la dotación de semántica a la web. Su especificación añade cláusulas de Horn exentas de negaciones y añade semántica. Su uso se centra en la clasificación en relaciones "is a". Este lenguaje es abandonado a medida que surgen OIL y DAML.

Su sintaxis puede verse en la página del proyecto[4].

KIF (Knowledge Interchange Format) [4]

Es un lenguaje para el intercambio de información entre sistemas diferentes (distintos programadores, tiempo, lenguaje, etc.). KIF, basado en lógica de primer orden, no está pensado para ser un lenguaje de representación del conocimiento en los sistemas informáticos, ni un lenguaje que pueda ser entendido por humanos, aunque esto último es posible. Es destacable su expresividad y la posibilidad de traducción desde y hacia otros lenguajes, permitiendo la declaración de términos, funciones, relaciones, conjuntos, representación de conocimiento y razonamiento no monótono. Los datos que son procesados en KIF por un ordenador, son pasados a un formato propio del sistema, cuando es necesario comunicarse con otro ordenador, son de nuevo pasados a KIF. El diseño de KIF enumera tres características principales:

Cuenta con *semántica declarativa*. Es posible comprender el significado de las expre-

[4]http://www.cs.umd.edu/projects/plus/SHOE/spec.html

siones sin que sea interpretado, esta es una diferencia significativa que tiene respecto a otros lenguajes.

Es *completo*, en lo referente a la lógica proporciona expresión para sentencias lógicas, en contraposición a otros lenguajes como SQL.

Ofrece *representación para el conocimiento*, esto permite al usuario tomar las decisiones de representación e introducir construcciones sin necesidad de traducirse a otro lenguaje.

KIF pretende maximizar la implementación, aunque no es este su objetivo, si otros programas desean utilizarlo debe ser posible; legibilidad, no es un lenguaje de interacción con humanos, pero es una buena característica el que sea posible la representación semántica.

Es en KIF, con algunas extensiones, en el lenguaje en el que se basa el mecanismo propuesto por Gruber [41]: Ontolingua.

Es posible consultar la semántica de este lenguaje en la página web del proyecto[5].

LOOM [64]

Se trata de un lenguaje de alto nivel basado en lógica de primer orden desarrollado al mismo tiempo que Ontolingua [21]. LOOM, que apoya su razonamiento en clasificadores, proporciona un lenguaje declarativo con gran capacidad de inferencia consistente en reglas, definiciones, hechos, etc. Su función principal es la construcción de taxonomías. Según los desarrolladores de LOOM, los paradigmas de orientación a objetos y lenguajes basados en reglas, que no son muy compatibles, se combinan generalmente favoreciendo la creación de sistemas de representación del conocimiento. Esta combinación no siempre resulta tan buena como se desea, aunque es remediable con el paradigma de la lógica descriptiva ya que evade las deficiencias de dichos paradigmas mientras se beneficia de sendas cualidades. Por ejemplo: los sistemas orientados a objetos son más fácilmente comprensibles para desarrolladores y

[5]http://logic.stanford.edu/kif/kif.html

facilitan el diseño de aplicaciones.

Como se ha comentado antes, está basado en clasificadores y en su lenguaje declarativo destacan entre otros recursos: definiciones, representadas por conceptos; reglas, representadas como vínculos entre descripciones; hechos, representados por cada instancia de relaciones entre descripciones.

Se puede consultar la sintaxis se LOOM en la página web del proyecto[6].

CycL (Cyc Language) [59]

A raíz del proyecto Cyc KB nace un lenguaje de representación que está originariamente basado en marcos y va evolucionando hasta uno de lógica de primer orden caracterizándose por tener una semántica clara, capacidad de inferencia rápida y toda la capacidad del cálculo de predicados en lógica de primer orden. Comienza a desarrollarse apoyándose en la necesidad de un lenguaje de representación que no tenga las mismas debilidades que los desarrollados hasta entonces.

Consta de un programa externo manejado por un usuario humano que es el encargado de interactuar con los dos niveles que lo conforman:

Nivel epistemológico gracias a un lenguaje de cálculo de predicados de primer orden se crea la base de conocimiento de forma que tenga semántica simple y sea fácil de utilizar.

Nivel heurístico utiliza representaciones de propósito especial que activan la inferencia.

Un lenguaje no ha de ser independiente de la base de conocimiento [61], esto puede hacer que sea menos útil e incluso más débil. Cabe destacar que es un lenguaje compuesto por microteorías, conjuntos de conceptos y hechos de una misma disciplina, teniendo en cuenta que cada microteoría no puede tener contradicciones.

[6]http://www.isi.edu/isd/LOOM/documentation/manual/quickguide.html

Está formado por un vocabulario de objetos (constantes, términos no atómicos, variables) combinables para formar predicados, conectivas, etc. Su sintaxis se encuentra en la página web del proyecto[7].

RDF (Resource Description Framework) [29, 101, 107]

Se trata de un lenguaje estructurado para el modelado de metadatos y para facilitar a las aplicaciones de la web el intercambio de conocimiento. Los metadatos son datos que describen los recursos en la web, se construyen en forma de tripleta: sujeto-predicado-objeto. RDF es a la web semántica lo que HTML es a la Web.

Tiene como objetivo favorecer la especificación semántica en la web a partir de XML y RDF. Así, define un mecanismo para la descripción de datos sin aceptar ningún dominio particular. En este lenguaje las clases se organizan de manera categórica, de forma que pueden ser extendidas convirtiéndose en subclases. RDF se compone de:

Recursos: cualquier cosa que sea descrita en RDF es un recurso, cada recurso es llamado por una URI y a su vez está fijado por un ID.

Propiedades: una propiedad es una cualidad característica, un atributo, una relación o cualquier entidad que describa un recurso, y cada una tiene un significado particular.

Declaraciones: un recurso junto con su propiedad característica y el valor de dicha propiedad forma una declaración. Cuando se lleva a cabo una declaración puede ser para definir otro recurso.

La sintaxis de RDF puede consultarse en la página del W3C[8].

RDF Schema(Resource Description Framework Schema) [6, 29, 107]

[7]http://www.cyc.com/cycdoc/ref/cycl-syntax.html
[8]http://www.w3.org/TR/REC-rdf-syntax/

Diseñado como una extensión de RDF. Define clases, relaciones entre ellas, propiedades y asociaciones con clases, el resultado de utilizar estas entidades es un vocabulario que describe el conocimiento sin tener en cuenta la validación. Se denomina esquema (schema) a una colección de clases. La semántica de los términos dados es definida por sus propiedades y el tipo de objetos valor de dichas propiedades, esta semántica será comprensible computacionalmente siguiendo en este caso una estructura: recurso-propiedad-valor-propiedad.

A diferencia de RDF, RDF Schema limita la expresividad utilizando las propiedades y valores y requiriendo una especificación en la definición del vocabulario. Se diferencia fundamentalmente con RDF en los mecanismos de descripción para los grupos de recursos y las relaciones entre ellos.

OIL (Ontology Inference Layer) [49]

Desarrollado en parte a partir de SHOE y en el seno del proyecto On-to-knowledge. OIL está diseñado para agrupar los estándares XML y RDF y diseñado para ser extensible en un futuro con clases, primitivas, e incluso definiciones probabilísticas. OIL pretende dar apoyo al razonamiento automático, ofrece las primitivas para modelar ontologías en paradigmas basados en marcos, siempre con una semántica simple, bien definida y clara. Los desarrolladores del lenguaje estiman [49] que las propuestas de lenguajes para intercambio de conocimiento basados en ontologías no son adecuadas a los estándares web.

Se puede resumir la intención de este lenguaje en la unificación de:

La *lógica descriptiva*. La idea fundamental en la representación del conocimiento es proporcionar teorías que expresen dicho conocimiento de manera estructurada, esto se consigue gracias a conceptos y restricciones que dan lugar a taxonomías.

Sistemas basados en marcos. Un marco ofrece un enfoque para modelar un aspecto de un dominio dado, para poder realizar ese modelado son necesarias primitivas que se

componen de clases, propiedades y atributos. El lenguaje propuesto incorpora este enfoque, pudiendo definir las relaciones como entidades independientes siendo posible su clasificación de manera jerárquica.

Estándares web: XML y RDF. Si por algo destaca la web semántica es por un lenguaje de intercambio de ontologías, por ello, cualquier lenguaje definido para este propósito debería incorporar esos estándares [52, 68]. XOL, es un lenguaje basado en XML desarrollado por BioOntology Core Group íntimamente relacionado con OIL llegando a ser éste último una extensión del primero. El papel de RDF en la web es añadir semántica al documento [54, 70], además estandariza la sintaxis de escritura y aporta un conjunto de primitivas que facilitan el modelado subclases, relaciones e instancias.

Se puede dividir el lenguaje en tres capas:

Nivel de objeto: en este nivel se describen las instancias de una ontología.

Primer meta-nivel: el nivel intermedio se encarga de facilitar las definiciones de la terminología que será instanciada en el nivel de objeto.

Segundo meta-nivel. Por último se describen las características de cada ontología, autor, nombre, clase... Originariamente diseñado para la descripción de los datos del autor de cualquier recurso web, es ahora utilizado comúnmente por grandes organizaciones.

Es posible consultar la sintaxis de este lenguaje en la página del proyecto[9].

DAML (DARPA Agent Markup Language) [1]

Creado por DARPA y la W3C como una extensión de XML y RDF (al igual que OIL) que facilite la comunicación y comprensión.

Actualmente la web, escrita en HTML, puede visualizarse gracias a que los navegadores

[9]http://www.ontoknowledge.org/oil

interpretan el lenguaje mostrándolo a los usuarios de manera comprensible para ellos, pero este lenguaje no facilita el uso de la información para que cualquier software la interprete. Aunque XML permite la descripción de la información a partir de etiquetas, no está diseñado para establecer las relaciones entre objetos. Es a partir del uso de ontologías cuando surge la posibilidad de relacionar objetos.

DAML+OIL [28, 50, 69]

Aprovechando los estándares XML y RDF surge un lenguaje más concreto para la representación y clasificación de conceptos añadiendo primitivas de sistemas basados en marcos. Se concibe para la descripción de la estructura de un ámbito orientado a objetos. De manera formal se puede decir que equivale a lógica descriptiva con una ontología que corresponde a la terminología de dicha lógica.

Hereda de OIL la potencia de la lógica descriptiva, dejando atrás el enfoque basado en marcos; y de DAML las clases y tipos de datos que facilitan la definición de la semántica del lenguaje y los problemas de inferencia. Tiene como gran inconveniente su complicado uso, que se intenta solventar con la creación de OWL. Es posible consultar la sintaxis del lenguaje en la página web del W3C[10].

OWL (Web Ontology Language) [29, 102, 107]

Desarrollado por la W3C especialmente para el diseño de ontologías, se trata de los más actuales y utilizados con este propósito gracias a su facilidad de procesado software. Se puede decir que está desarrollado incorporando ciertos aspectos aprendidos en la aplicación del lenguaje DAML+OIL. Apoyado en XML, RDF y RDF Schema con estructuras que facilitan la expresividad; contiene, igual que RDF: clases, relaciones y propiedades, diferenciándose de este en las limitaciones y restricciones, añadiendo cardinalidad en las relaciones, igualdad y clases enumeradas entre otras características. Se disgrega en tres sublenguajes clasificados

[10]http://www.w3.org/TR/daml+oil-reference

según su nivel de expresividad:

OWL Lite: es el sublenguaje que menos expresividad ofrece de los tres, el más restrictivo, si se compara con RDF añade restricciones de rango local, de cardinalidad (0 ó 1), igualdad, en ciertas propiedades, etc. Diseñado para usuarios que desean relaciones jerárquicas. Destaca la rápida migración que puede darse de tesauro a taxonomía. Es el que menor complejidad tiene.

OWL DL: añade, comparativamente al anterior, soporte para negaciones, disyunciones, restricciones de cardinalidad, enumeraciones y valores restrictivos. Diseñado para usuarios que desean la máxima expresividad manteniendo la completitud computacional. Contiene todas las construcciones de OIL.

OWL Full: carece de restricciones. Permite especificar clases como instancias. Diseñado para usuarios que desean la máxima expresividad y libertad sintáctica sacrificando, posiblemente, la completitud computacional. Cabe destacar que es muy poco probable que un software pueda soportar la capacidad de razonamiento de las características de OWL Full.

Se comprueba que cada sublenguaje es una extensión del anterior. Siendo *DL* y *Lite* una versión restrictiva de *Full* y éste una extensión de RDF. Aunque *Lite* es muy restrictivo su expresividad está muy próxima al siguiente, *DL,* sin embargo *Full* aunque ofrece la máxima expresividad, puede acarrear problemas de inferencia causados por la libertad sintáctica.

La sintaxis es consultable en la página de la W3C[11].

FOAF (Friend of a friend) [7]

Se trata de un proyecto dedicado a vincular gente e información de esa gente en la web. Apoyado en los lenguajes RDFS/OWL, no es propiamente un lenguaje en sí mismo, sino un

[11]http://www.w3.org/TR/owl2-overview/#Syntaxes

vocabulario con expresiones de los mencionados previamente. Está basado en tres tipos de redes:

Redes sociales, entre ellas asociaciones, reacciones y colaboración entre personas.

Redes de representación describe una vista de universos en términos actuales.

Redes de información este tipo de redes utilizan la web y los vínculos para compartir publicaciones entre diferentes dominios.

FOAF no intenta competir con sitios web sociales, sino dar un enfoque diferente con el que se upede recopilar información de estos sitios.

Describe un mundo utilizando ideas de la web dando como resultado una red de documentos capaces de describir a la gente, cada documento recoge términos, descripción de gente, grupos, etc. Cualquier aplicación que utilice un documento FOAF puede utilizar las partes que desee e ignorar las que no le resulten interesantes, por lo general, suelen ser ignoradas las partes antiguas y obsoletas y utilizadas las demás, realizando vínculos entre la gente.

Se puede agrupar FOAF en:

Base: es el centro de FOAF. Sin tener en cuenta la tecnología ni el tiempo, existen ciertos términos capaces de describir a gente y grupos sociales.

Social web: además de los términos detallados en la base, existen otros que describen Internet tales como cuentas, libretas de direcciones, etc.

Linked Data Utilities: FOAF comienza como proyecto de RDF Web y pretende especificar un modelo para la publicación de datos.

Su idea básica es sencilla publicar información en documentos FOAF de manera que sea computacional, estando dicha información formada por aquella que los usuarios mantienen en la cabeza junto con la que está en los documentos. Para ello crea una base de datos

estandarizando categorías, relaciones, tipos de relaciones, etc.

Es posible ver la sintaxis de FOAF en la página web del proyecto[12].

2.1.3. Clasificación

Cuando se diseña y desarrolla una ontología es necesario saber para qué se está diseñando, porque en función de la finalidad de la ontología podrá clasificarse en un tipo u otro, así, una ontología se puede clasificar según el problema que pretenda resolver [71]:

- Ontología de contenido. [100] Este tipo de ontologías pretende poder reutilizarse en otros sistemas.
- Ontologías de indexación. Permiten recuperar información de otros sistemas.
- Ontologías de comunicación. Estas ontologías contestan a preguntas específicas, se utilizan fundamentalmente por agentes.
- Meta-ontologías. Pretenden representar otras ontologías de un mismo dominio. Pueden ser clasificadas según el área de conocimiento que abarquen. Existen cuatro categorías: de aplicación, de dominio, genéricas y de representación [100].
    - Las *ontologías de aplicación* [43] son las que definen todos los conceptos necesarios para modelar el conocimiento con una tarea específica. Normalmente este tipo de ontologías se combinan con conceptos de ontologías de dominio y de ontologías genéricas. Además, contienen métodos específicos. Tienen como inconveniente no ser reutilizables por sí mismas.
    - Las *ontologías de dominio* son aquellas que describen una disciplina especializada.

[12] http://xmlns.com/foaf/spec/

- *Ontologías genéricas* o *de nivel superior* son parecidas a las de dominio, en este caso los conceptos que se definen son de un ámbito más general es por ello que su distinción con las de dominio es muy leve. Este tipo de ontologías no se centran en ninguna disciplina.
- *Ontologías de representación*. Ofrecen el vocabulario necesario para el diseño de otras ontologías.

Es fácil apreciar que aquellas que más utilidad ofrecen son las llamadas *de dominio* ya que favorecen el conocimiento compartido de materias específicas según un conjunto de expertos. Las ontologías de aplicación, por otro lado son muy útiles, ya que facilitan la gestión de conocimiento en empresas.

Así, se puede decir que las ontologías de aplicación son las de nivel más bajo, mientras que las de dominio y representación establecen un nivel intermedio siendo superadas por las genéricas.

2.1.4. Aplicaciones

La principal aplicación de una ontología tal y como explica su propia definición es la organización y el conocimiento compartido.

Es una herramienta muy demandada para la adquisición de información, además dicha información se ofrece de manera organizada en la mayoría de las ocasiones.

Una ontología se fundamenta en los metadatos, por tanto, otra de las aplicaciones que se le da es el uso y organización de metadatos en las páginas web.

En este caso, vamos a enfocar esta sección en las aplicaciones de las ontologías dentro del ámbito de la ingeniería software. Basándose en el proceso del ciclo de vida de esta ingeniería,

podemos diferenciar distintas aplicaciones:

Análisis y diseño: cuando se comienza un proceso de ingeniería software es fundamental determinar los requerimientos. Es necesario que los ingenieros software que los describen se documenten sobre el dominio en el que se encuentra, así una ontología permitirá comprender de manera más sencilla los requerimientos [62, 104], siempre que se hayan descrito de la manera adecuada [25, 67] y la ontología esté correctamente implementada.

Reutilización de componentes: gracias al uso de ontologías será más sencillo para el ingeniero software comprender cual es el funcionamiento del módulo que se desea reutilizar.

Implementación: se puede pensar que una ontología no es útil cuando se va a realizar la implementación, no obstante, el hecho de que el desarrollo de ontologías se base en un enfoque orientado a objetos favorece la implementación de aplicaciones con este mismo paradigma.

servicios web Semánticos: el objetivo de esta tecnología es dotar a la web de semántica propia, gracias a las ontologías y a los servicios web esto es posible. Una combinación de ambos será la base de la web semántica ideal.

2.2. Servicios web

Un servicio es la forma en la que se comunica cualquier software con otro: script, aplicaciones entre distintos servidores u otro sistema en cualquier ordenador. Los servicios ofrecen al usuario una interfaz con la que comunicarse con otros servicios. Se puede decir que son servicios siempre que consten de las siguientes características:

Se *describen* a sí mismos haciendo que otro componente comprenda su funcionalidad y cómo acceder a ellos.

Permiten ser *localizados* por otros componentes pudiendo ser utilizados cuando se

requieran.

Podrán *invocarse* en cualquier momento por cualquier componente que lo solicite.

Cuando este tipo de componentes son accesibles a través de la red se denominan servicios web.

Mientras que la web tradicional se visualiza como poco más que un protocolo de acceso a objetos que permite la conexión entre usuarios y aplicaciones, los servicios web proporcionan funcionalidad totalmente independiente del lenguaje de representación y de su ejecución (remota o local), permitiendo una conexión no sólo usuario-aplicación sino también aplicación-aplicación; esto hace, entre otras cosas, que sea un instrumento ideal para los modelos de negocio B2B y B2C [97].

Con el desarrollo de los servicios web, surge un reto entre los proveedores de servicios de comunicación: sacar provecho a la confluencia entre software y redes a partir de los servicios web.

Los servicios web se comunican con otros gracias a la estandarización de ciertas tecnologías: su infraestructura ha de ser común, siendo tolerada por cualquier proveedor. Además cualquier servicio expresado en un lenguaje ha de implementar las especificaciones estándar para servicios web de la misma forma que si estuviera en otro lenguaje.

Fue gracias a grandes compañías como IBM, Microsoft y Sun Microsystem [73], que se hizo posible la creación de estándares para el desarrollo de los servicios web.

2.2.1. Definición

Un servicio web es un tipo de aplicación web. Podrá ser desde una solicitud sencilla hasta un complejo modelo de negocio. Basado en estándares XML e identificado por una URL se

diferencia con los sitios web tradicionales en su forma de responder a las solicitudes que se les realizan. Cuando accedemos a un sitio web la respuesta ofrecida es un documento de texto con instrucciones sencillas que el navegador se encarga de procesar y visualizar al usuario. Cuando se realiza una solicitud a un servicio web no se envía el nombre de un documento o una llamada con ciertos parámetros, sino un documento con una determinada especificación (SOAP) [79].

El principal inconveniente con el que se encuentra un servicio web a la hora de ejecutarse es que necesita responder a las solicitudes realizadas en tiempo de ejecución, sin conocer a la perfección con qué aplicación se está encontrando, esto es fácilmente solucionable gracias a los documentos tipo "MIME", con ellos cualquier aplicación podrá saber cómo reaccionar ante cualquier solicitud. El problema que existe en los servicios web es que se encuentran más tipos de servicios que tipos de documentos, convirtiendo su interacción en un proceso más complicado, ya que no es habitual que un servicio sepa cómo actuar en cuanto se encuentra con una aplicación totalmente nueva para él [38].

Como todo, el uso de servicios web tiene ciertas **ventajas** sobre el uso de otras tecnologías: al estar estandarizados los servicios web facilitan la programación y mantenimiento de las aplicaciones que los utilizan. Un servicio web no pretende sustituir otras aplicaciones, de hecho, no puede hacerlo ya que dependerá de ellas para ejecutarse, en ocasiones dependerá sólo de una y en otras de varias. Ofrecen interoperabilidad, cualquier servicio web puede interactuar con otro, además, gracias a la forma de comunicación podrá jugar el papel de cliente o servidor. Cabe destacar que los servicios web hacen a los sistemas más rápidos, debido a su integración con otros servicios y sobre todo su interoperabilidad, están diseñados para que sean soportados sobre cualquier sistema operativo u ordenador. Es útil conocer ciertas herramientas que facilitan a los desarrolladores la implementación de servicios web con SOAP.

También como en todo, el uso de servicios web tiene ciertos **inconvenientes** y el más

fácil de ver es que necesita conexión a Internet para poder ejecutarse, lo que suma también el inconveniente de la disponibilidad, pues, como todos los componentes web es posible que en algún momento no estén disponibles, por otro lado, es posible que el usuario necesite agregar métodos que no estén disponibles o modificar alguno de los existentes, comprometiendo así la integridad del servicio.

Existen ciertos riesgos de seguridad. Los servicios web utilizan tecnologías de seguridad vulnerable y la carencia de sistemas de seguridad estandarizados supone un riesgo en esta tecnología, aunque en la mayoría de los casos los servicios web se utilizan de manera interna en las empresas, por lo que la seguridad no se ve comprometida al quedar dicho servicio dentro de los sistemas de seguridad de la compañía.

Debido a la arquitectura interna de los servicios web, más concretamente el estándar SOAP que debe realizar varios saltos y es enrutado en diferentes puntos antes de alcanzar su destino final, la seguridad en los servicios web es un hecho importante. Sin embargo, hay ciertas técnicas que pueden utilizarse para garantizar la seguridad, como con el cifrado XML, la firma digital XML, SAML (Security Assertion Markup Language), o la validación de datos son algunas de las más utilizadas. No obstante cabe destacar que la seguridad en UDDI garantiza en cierto modo que ningún servicio no confiable pueda ser registrado.

2.2.2. Arquitectura

Desde el inicio de la web tradicional [14] se había estado trabajando para desarrollar estándares básicos. Una vez establecidos, el siguiente paso era conseguir una especificación que ayudara a la transferencia de mensajes y asegurara la interoperabilidad estando, finalmente, dicha especificación basada en XML, lo que facilitó el camino a los servicios web. Fue posible gracias al esfuerzo de grandes compañías: Microsoft trabajó en el diseño de SOAP, un estándar flexible y de propósito general; por otro lado era necesario conectar los servicios

web, esto se hizo posible gracias a la unión de los lenguajes desarrollados a partir de IBM (Network Accessible Service Specification Language [86]), y Microsoft (Service Description Language y SOAP Contract Language), unión que dio lugar al conocido WSDL.

Es gracias a SOAP y WSDL la viabilidad que existe a la hora de desarrollarlos. Como ya se había comentado cuando se realiza una solicitud a un servicio web se envía un documento con una especificación SOAP. Básicamente la arquitectura de un servicio web, que está descrito gracias a WSDL, es el intercambio de mensajes XML en formato SOAP. Por tanto, y resumiendo de manera muy general, podemos decir que gracias a WSDL y SOAP se realizan las solicitudes y respuestas entre servicios web.

Una vez que se tiene implementado la siguiente etapa es poder descubrir si existe alguno que tenga la misma funcionalidad, y lo que es más importante, encontrar cualquier servicio que pueda ser interesante para interaccionar con otro, esto se consigue gracias a UDDI.

Cualquier servicio web consta de una arquitectura básica que puede visualizarse en la figura 2.9.

- *Service Requester*: recibe un servicio de otra aplicación. En caso de saber cuál es su localización envía a *Service Provider* una función *Bind()* que le permitirá establecer un servicio, en caso de no conocer su localización lanza la función *Find()* a *Service Registry.*
- *Service Registry*: devuelve información sobre el servicio web por el que ha preguntado *Service Requester* gracias a la función *Find(),* este servicio contiene detalles de conexión y clasificación.
- *Service Provider*: contiene los elementos de conexión y clasificación que envía a *Service Registry* a través de la función *Publish().*

Los servicios web cuentan con una estructura dinámica que facilita su localización e

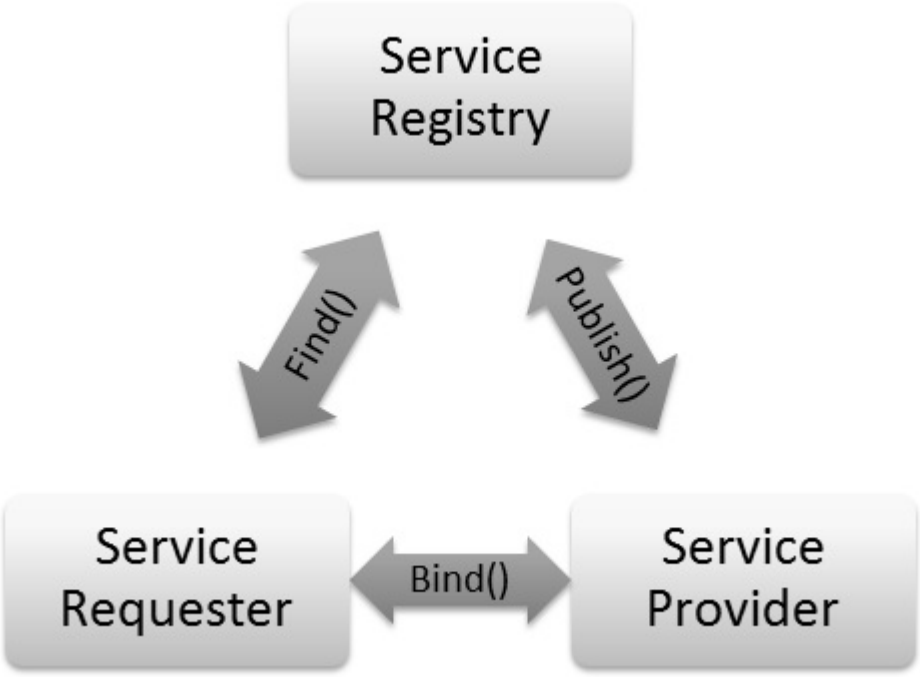

Figura 2.9: Arquitectura de un servicio web

invocación. Resumiendo la cuestión, es descrito gracias a un fichero WSDL, implementado gracias a SOAP y localizado gracias a UDDI. A continuación se describen los componentes de su arquitectura de manera más específica:

XML (eXtensible Markup Language)[5, 37, 73, 79, 85]

Como se ha comentado anteriormente, XML es un lenguaje estándar basado en etiquetas. Gracias a él es posible que una aplicación interprete de manera correcta la salida de otra reaccionando ante ella de manera correcta. Con los lenguajes informáticos ocurre lo mismo que cuando dos personas desean comunicarse, será posible si hablan el mismo idioma. En caso contrario se necesitará un intermediario que interprete qué es lo que se quiere decir. En este caso XML cumple el papel de intermediario. Se pueden distinguir los siguientes elementos:

Documento XML: no es más que un documento escrito en XML estándar.

Cuando se utiliza un *parser XML* se da como entrada un documento XML y éste devuelve como salida un fichero comprensible con el contenido de la entrada.

Document Type Definition (DTD): se trata de la descripción de las etiquetas de un documento y la relación entre ambos. Su versión mejorada es *XML Schema*

Namespace o espacio de nombres: define los nombres de las etiquetes que serán únicos para evitar conflictos.

SOAP(Simple Object Access Protocol) [22, 38, 73]

Basado en XML y soportado en los protocolos básicos de Internet (HTTP, SMTP, TCP/IP, MQSeries...), ofrece una forma normalizada de intercambio de mensajes y llamadas a procedimientos remotos (RCP), en nuestro caso servicios web. SOAP se soporta en un mecanismo de pregunta-respuesta utilizando mensajes que invocan a funciones. Los mensajes pueden actuar como un simple protocolo o ser de paso y contiene ciertas decisiones: cómo se debe procesar el mensaje e incluso qué servicio web debe recibirlo cuando termina de ejecutarse el actual. SOAP incluye un marco de descripción de lo que hay en el mensaje y cómo ha de procesarse; un reglamento sobre codificación para las instancias del mensaje y un convenio que define cómo se han de llamar y responder las solicitudes. Su estructura es bastante sencilla, consta de un elemento XML con dos "hijos", uno contiene la cabecera y el otro el cuerpo del mensaje. Cuando en el mensaje enviado se eliminan la URL, el espacio de nombres y el encabezado SOAP, pasa a ser una llamada a un método incrustada en un envío utilizando HTTP. Son muchos los lenguajes que generan y procesan mensajes SOAP de manera automática (Java, C, Perl...), el hecho de que éstos los acepten facilita el intercambio entre aplicaciones escritas en distintos lenguajes.

WSDL (Web Services Description Language) [22, 38, 73]

Es el lenguaje específico para descripción de servicios web que ofrece a SOAP la informa-

ción sobre la interacción de servicios. Detalla los servicios web como conjuntos de documentos que agrupan las aplicaciones y pueden interaccionar entre ellas con el intercambio de mensajes. Los servicios web interactúan entre ellos gracias a funciones con mensajes de entrada y posibles mensajes de salida, sus funciones pueden ser respuestas a solicitudes realizadas al servicio o ser lanzadas por este para una consulta a otro. A la hora de programar un servicio web, los desarrolladores necesitan conocer detalles de la implementación y de la interfaz del servicio con el que van a interaccionar. Además, cuando la integración de componentes es necesaria, si el proceso se realiza manualmente es más propenso a errores. Los documentos WSDL resultan ser de mucha utilidad tanto para conocer dichos detalles como para el procesado de información cuando se integran componentes, llegando, incluso, a poder automatizar el proceso. HTTP, XML y SOAP aportan significado para la llamada de los servicios web; WSDL se encarga de facilitar la información para integrar los detalles de funciones, parámetros de llamada, tipos de datos para los mensajes XML, información sobre los protocolos de transporte y dirección para la localización de los servicios.

Se pueden localizar dos aclaraciones:

Descripción abstracta: es el intercambio de mensajes en la interacción de servicios, se diferencian tres componentes en la interfaz abstracta: el vocabulario, el mensaje y la interacción.

Información de enlace concreta: hasta ahora todo lo que se ha descrito define la funcionalidad a nivel de aplicación, pero también es necesario conocer: qué protocolo de comunicación ha de utilizarse, cómo lograr la interacción individual con ese protocolo, y dónde terminar la comunicación.

UDDI (Universal Description, Discovery and Integration)[22, 38, 73]

Es en sí mismo un servicio web accedido vía SOAP y descrito en WSDL que ofrece una interfaz para registro y publicación de servicios web disponibles. Cuando una compañía no

conoce la localización de un fichero WSDL que desea utilizar ha de recurrir a un registro. UDDI ofrece a las compañías de manera rápida, fácil, sistemática y unificada encontrar los servicios que desean utilizar a través de un registro centralizado de servicios: es un directorio de servicios web online (UBR: UDDI Bussines Registry). UDDI está pensado para que cualquier empresa que ofrezca servicios web pueda describir tanto su negocio como sus servicios, pueda conocer otros servicios interesantes para ellos así como las compañías que ofertan estos servicios y además permite integrar sus propios servicios con los encontrados anteriormente. UBR organiza la información en tres tipos:

Páginas blancas: contiene la información de nombres y detalles de contacto.

Páginas amarillas: contiene información según la taxonomía de servicio y negocio.

Páginas verdes: contiene la información técnica del servicio.

La información en UDDI se organiza de la siguiente forma:

Entidad de negocio: da información sobre la empresa y contiene identificador (único), nombre de la empresa, descripción, información de contacto, lista de categorías e identificadores que describen a la empresa y una URL para más información.

Servicio de negocio: es una lista de servicios ofertados por la entidad, en esta lista cada servicio contendrá su propia descripción, lista de categorías que lo describen y una lista de punteros de referencia relacionada con el servicio.

Punteros de especificación: es una lista de plantillas de unión con los puntos técnicos del servicio.

Tipos de servicios: cada tipo de servicio viene definido por lo que se denomina un *tModel.* El concepto *tModel* es sencillo: se trata de una plantilla de unión resultado de la implementación de uno o más *tModel,* dentro de la plantilla está la URL de la especificación del *tModel.* Para la identificación de sistemas según su taxonomía se registra como *tModel,*

dicha información se codifica en un par nombre-valor clasificado por la clave de *tModel* que será capaz de identificar a qué taxonomía pertenece cada par.

UDDI no se utiliza solamente para ser localizado en registros comunes, muchas empresas comenzaron utilizándolo de manera privada ofreciendo la misma funcionalidad a un conjunto restringido de usuarios. Sí es cierto que en el directorio UDDI se pueden localizar servicios muy específicos.

Así, un usuario acudiría a UDDI para localizar un servicio que estará descrito en WSDL. A dicho servicio se podrá acceder gracias a un estándar de acceso denominado SOAP. Todos estos protocolos están basados en XML y soportados sobre los estándares de Internet (HTTP, TCP/IP...). El esquema de la figura 2.10 se resume la arquitectura de un servicio web.

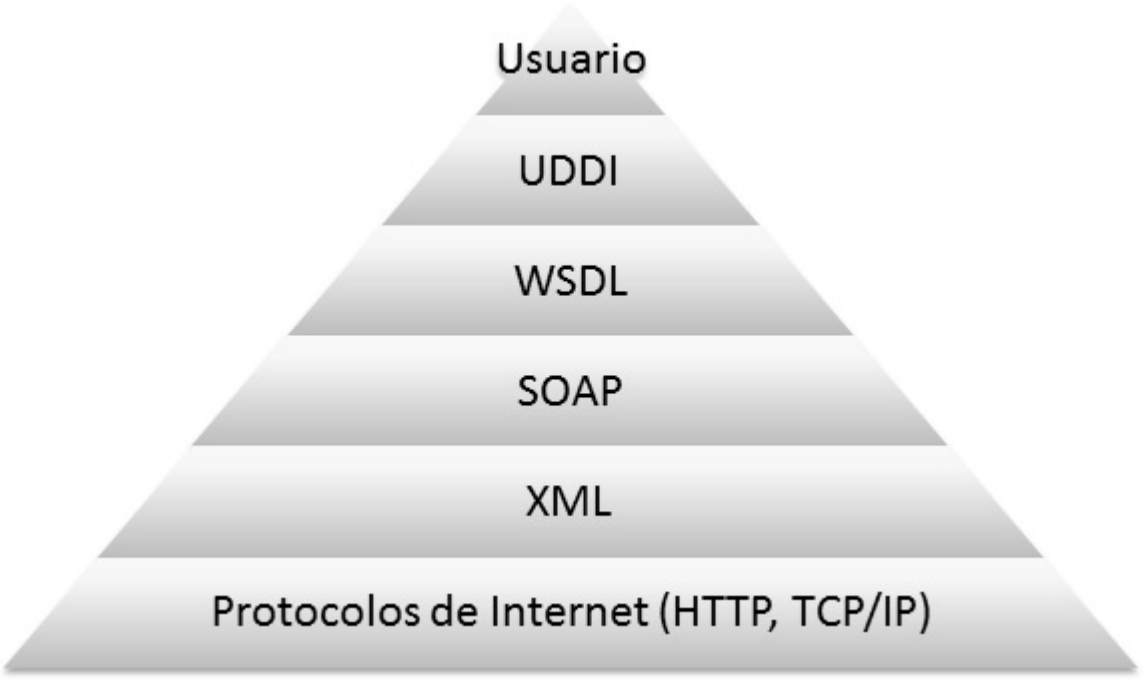

Figura 2.10: Esquema de capas de los servicios web

2.2.3. Retos y aplicaciones

2.2.3.1. Retos

Los servicios web son una tecnología en constante evolución lo que los hace crecer de manera rápida gracias a los retos que se le plantean [29]:

Modelo de complejidad: antes era normal utilizar los servicios web para operaciones de negocio seguras, algo que todavía está poco maduro y requiere una total comprensión de todos los aspectos que se manejan, debido al modelo de comunicación entre cliente y servidor.

Tecnología: uno de los aspectos más complicados a los que se enfrentan los servicios web es la comunicación. La comunicación requiere entre otras áreas persistencia, intercambio, recuperación sólida de los errores..., áreas que están en proceso de estandarización.

Semántica: sin duda el reto más grande de los servicios web, y el que ocupa este proyecto es la semántica. Los servicios web carecen de mecanismos para dotar de semántica a las aplicaciones con las que se comunican. Cabe destacar que el proyecto OASIS Semantic Execution Environment[13] junto con W3C están en proceso de estandarizar esta cuestión.

2.2.3.2. Aplicaciones

Gracias a la claridad de las rutinas los servicios web es posible implementar un programa que para el desarrollador sea muy difícil de realizar a través de librerías.

Por lo general los servicios web prestan utilidades a los clientes que solicitan su uso. Pero no siempre es así, existen servicios web que se integran con otras aplicaciones para unir

[13]https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=semantic-ex

conocimientos ofreciéndolos al cliente en la medida que necesite. Ejemplos de estos usos son, entre otros:

- La integración de un buscador de páginas web a cualquier otro sitio, realizando una programación distribuida o sin carga en el ordenador del cliente. Este servicio lo presta Google con una de sus APIS.
- Hay implementados ciertos servicios webs como el de Barnes and Noble[14] que facilita el precio de un libro dado su ISBN.
- Otro servicio interesante es conocer a quién pertenecen los dominios registrados en Internet, como ofrecen NetworkSolutions[15]
- Otra de las aplicaciones de los servicios web y una de las más utilizados es la traducción automática, es decir, traducir un sitio web sin necesidad de hacerlo paso a paso.

2.3. Terminología basada en corpus

En la actualidad existen múltiples aplicaciones elaboradas en el marco de la Lingüística Computacional, es decir, combinando la Lingüística y la Informática. Aplicaciones que utilizan el lenguaje sin necesidad de analizarlo (procesadores de textos, sistemas de comunicación ...); aquellos que automatizan algún proceso lingüístico como bases de datos, sistemas de traducción, sistemas de corrección, ...; sistemas que analizan los elementos lingüísticos, como analizadores, clasificadores, etc.; sistemas que se valen de las unidades lingüísticas para la representación y adquisición del conocimiento a partir de procesos de inferencia, como son los sistemas de vaciado automático, traducción y corrección automática ... [20].

[14] http://www.barnesandnoble.com/
[15] http://www.networksolutions.com/

En este proyecto, el área de la ingeniería del conocimiento tiene especial interés, debido a los aspectos comentados anteriormente. Como ya se ha dicho, la ingeniería del conocimiento pretende codificar la información de manera que sea analizable por parte de un ordenador. Para ello la información es codificada a partir de reglas lógicas que serán utilizadas en un proceso de inferencia. Dicha codificación solamente será posible si las reglas y axiomas se han definido correctamente y su diseño es el adecuado.

Es cierto que la Informática es muy útil para la Terminología, pero también es cierto que la Informática necesita de la Terminología en algunos aspectos. Cualquier programa que utilice procesamiento del lenguaje natural necesitará de un diccionario como punto de referencia. De manera más concreta, para poder procesar automáticamente la información es necesario que su terminología esté correctamente definida. Cada término denominará un concepto que representará una parte del conocimiento buscado.

Existen varias formas de representación del conocimiento: tesauros, ontologías y taxonomías. Según la utilidad que vaya a dársele a la representación, será mejor un tipo de representación u otro.

2.3.1. Definición

Como se ha comentado en el capítulo 1, cuando hablamos de Terminología nos referimos a una materia interdisciplinar cuyo objetivo principal es el estudio y descripción de unidades de conocimiento especializado en lenguas de especialidad. Gracias a la Terminología es posible la elaboración de léxicos de especialidad, como diccionarios y la normalización de nuevos términos. Pero no es posible la elaboración de terminologías si no se poseen datos de las materias específicas que se sirven de ellas. Por tanto, esta especialidad se nutre de datos de todas las disciplinas a las que dedica su trabajo [20].

En las primeras etapas de la colaboración entre Informática y Terminología se contaba

con grandes bancos de información. Con la evolución de la Informática se ha ido dando paso a una información más especializada, distribuida y clasificada. En la actualidad existen muchos bancos de información de tamaño mucho más pequeño que los de las primeras etapas. La evolución de la Informática orientada al usuario ha refinado las herramientas con las que los terminólogos trabajan facilitando al usuario la localización de información hasta el punto de que, aunque en un inicio era necesaria la intervención de un terminólogo para poder realizar una consulta, en la actualidad el propio usuario interesado puede ejecutarla y verse gratamente sorprendido con la información encontrada.

En las primeras etapas, la información estaba situada en los sistemas de manera local. Esto suponía un inconveniente a la hora de realizar cualquier consulta. La solución vino dada por la evolución de las redes, que facilitó el acceso a datos distribuidos de manera remota. Esto permitió la unificación de conocimiento, así como la consulta y modificación por parte de expertos y usuarios.

En la actualidad, la terminología se beneficia de los sistemas expertos y la inteligencia artificial. Estas tecnologías permiten al usuario descargar trabajo realizando de manera automatizada algunas tareas que el terminólogo tenía que llevar a cabo manualmente. Los ordenadores cobran de esta manera un papel importante en el proceso de elaboración de la terminología permitiendo entre otras técnicas la selección de documentos, la detección de neologismos y la elaboración de definiciones.

2.3.2. Metodología

La aportación de la Informática a la Terminología ha supuesto la alteración de la metodología en especial en lo relacionado con la compilación de terminologías y la organización del trabajo a seguir. Por parte del área de la inteligencia artificial, esta unión de disciplinas supone la posibilidad de crear sistemas expertos que realicen las funciones de los terminólo-

gos.

La terminología, para organizar un área de conocimiento, utiliza textos que serán manejados posteriormente para detectar nuevos conceptos. Sinclair, define un corpus como *una colección de documentos en lenguaje natural, seleccionados para caracterizar el estudio de una lengua o variedad de la misma* [90]. Consta de textos que hablan de una misma materia y servirán para la documentación y extracción de terminología. Los textos en formato electrónico serán idóneos para ese proceso de manera que será más sencillo seguir la metodología explicada a continuación [24]. Gracias a la organización de los corpus es posible trabajar sobre ellos de manera automatizada. Se analizará el corpus correspondiente incluyendo los textos que facilitarán la detección de unidades representativas. Además, el acceso a bases de datos permitirá la comprobación de términos detectados y las referencias a los mismos.

En el proceso terminológico en el que interviene la Informática es posible distinguir las siguientes etapas [20]:

- Documentación previa. En esta etapa el experto se forma y recopila documentación que será la que componga el corpus. Gracias a Internet esta etapa se ha hecho más llevadera.
- Constitución del corpus y extracción de datos. Existe cierto software que agiliza la transcripción e incorporación de textos al corpus. Un ejemplo de este software son los denominados OCR (Optical Character Recognition). Una vez incroporados podrán analizarse y realizar el proceso de extracción.
- Confección de ficheros. En esta fase, el terminólogo confecciona las fichas, en ocasiones, de manera automática, ya que el trabajo anterior ha permitido indexar los documentos, esta tarea permitirá combinar fichas para completarlas con otras e incluso redactar las definiciones.
- Verificación y compleción de la información. Llegados a este punto del proceso, se

realiza una nueva búsqueda de información que pueda resultar relevante. Este trabajo es más específico.

- Edición de la terminología. Por último, el soporte elegido para manetener la terminología será elegido convenientemente en función de la edición y función del tipo de usuarios, objetivos, etc., que tenga la terminología extraída en el proceso anterior.

Cabe esperar que existan inconvenientes y es que el proceso terminológico no puede llevarse a cabo solamente de manera automática, necesita la intervención humana para realizarse de manera correcta. Un ejemplo claro de esta intervención es el procesado de textos a partir de OCR. En ocasiones el software utilizado no es lo suficiente fino como para detectar todo lo que se desea, en ese caso, el terminólogo ha de completar o complementar la acción. Otro de los problemas a los que se hace frente es el hecho de que no existan corpus textuales especializados computacionales; es un proceso que hay que realizar paso por paso. Aunque estos problemas van viendo poco a poco su solución gracias a los avances de la Lingüística Computacional, la inteligencia artificial y la ingeniería del conocimiento.

Una de las formas de gestión del conocimiento en la Terminología son los bancos de datos. Los bancos de datos son herramientas de referencia cada vez más común y accesible al usuario que almacenan información especializada. La necesidad y oportunidad de intercambio de información a nivel internacional ha favorecido la aparición y desarrollo de bancos de datos y recursos que facilitan su almacenamiento y recuperación.

En los bancos de datos la información se almacena en registros que están organizados en campos. Para llevar a cabo la elaboración de un banco de datos se realiza un proceso metódico [20], que no será explicado en este trabajo ya que no se considera relevante para su comprensión. Existen tres tipos básicos de bancos de datos: documentales y bibliográficos, textuales especializados, sobre conocimientos. Para el presente proyecto cobran especial interés los bancos sobre conocimientos: basados en inteligencia artificial aplicada a corpus

textuales. Los sistemas expertos tratan la información que encuentran en la base de datos a través de módulos específicos para ello.

A partir de los bancos de referencia podemos llegar al concepto de banco terminológico. Los bancos terminológicos son bancos de datos que almacenan información multilingüe de especialidad, facilitando a terminólogos el almacenamiento, mantenimiento y actualización de la misma.

Los bancos terminológicos cuentan con diferentes bases de datos que se complementan entre sí. Suelen crearse para tareas específicas relacionadas en su mayoría con la traducción. Simplifican la tarea a los traductores, devolviendo información de diferentes fuentes y con propuestas fiables dentro de un ámbito de especialidad cuando éstos realizan búsquedas de datos.

En la actualidad, los bancos de datos son accesibles no solamente por usuarios expertos en la materia sino por casi cualquier persona, convirtiéndose en herramientas de gestión del conocimiento conforme a las necesidades de los usuarios.

Según Rondeau existen diferentes tipos de bancos terminológicos [83]:

(a) Objetivos.

(b) Naturaleza de los datos.

(c) Destinatarios.

(d) Organización de los datos.

(e) Actitud lingüística.

(f) Modos de difusión.

Aunque según otros autores especializados los tipos *c)* y *f)* no pueden ser clasificados

en sí mismos como tipos de bancos terminológicos. Y además incluyen otros que consideran más importantes, así a la clasificación anterior se añade según:

- La orientación de base [27].
- Sistema (primera o segunda generación) [87].
- El hardware [20].
- El número de lenguas [20].
- Interés de los datos (enciclopédicos, visuales, fraseológicos, ...) [20].
- Dimensión del banco [20].
- Temática [20].

Cabe destacar que los criterios para definir un banco de terminología no son exclusivos sino que suelen combinarse.

Para la creación de un banco terminológico es habitual seguir el mismo procedimiento que para la creación de un banco de datos, diferenciándose en tres etapas diferentes: recopilación, almacenamiento y recuperación [20].

En la etapa de recopilación se decide el material, el lugar en el que se buscará dicho material, el método de extracción de información, el proceso a seguir, la información que se va a describir de cada término, fuentes, contextos y distribución de la ficha terminológica. Al final de esta fase se tendrá el fichero principal que compondrá la base de términos.

En la etapa de almacenamiento la información queda registrada en ficheros con la estructura adecuada para la ficha terminológica, en estos ficheros se encuentra el término, la fuente de referencia, la definición de conceptos relacionados [72], categorías gramaticales, abreviaturas, equivalencias a otras lenguas así como los datos del autor de la ficha. El fichero

principal se relaciona con ficheros complementarios que contienen información aclaratoria, también se relacionará con otras bases de datos que tendrán información relacionada como la bibliografía u otras áreas de conocimiento interesantes relacionadas.

En la última fase, la de recuperación de información, es importante saber que hay que tener presente en todo momento que la finalidad de un banco de términos es ofrecer información. El hecho de que un usuario consulte la base de términos de manera habitual vendrá determinado por la estructura que tenga. Dicha estructura será la base para el proceso de recuperación de información realizado de manera automática a través del ordenador. La recuperación tendrá en cuenta la forma en la que se pregunta, el lenguaje, el modo (online, local, por teléfono, por correo, ...). Además ha de ofrecer al usuario total libertad en las consultas, pudiendo éste realizar la búsqueda según criterios, datos concretos o aproximaciones a datos, uno o varios, etc.

2.3.3. Olivaterm

Olivaterm[16] es el resultado del proyecto de investigación multidisciplinar *Terminología del aceite de oliva y comercio: China y otros mercados internacionales*[17] cuyo fin es facilitar la comunicación especializada en materia de aceite de oliva y transferir conocimiento sobre esta área temática. Surge para paliar las dificultades de comunicación del sector en las transacciones comerciales internacionales y de cara a su promoción dentro y fuera de nuestras fronteras [81].

Se pueden distinguir las siguientes etapas en el desarrollo del trabajo:

- Definición y delimitación del trabajo. En esta estapa el trabajo es presentado definiendo su alcance y objetivos.

[16] http://www.olivaterm.com

[17] P07-HUM-03041, financiado por la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía

- Preparación y documentación. Se lleva a cabo el proceso de documentación del experto para el mejor conocimiento de la materia a tratar.
- Estructuración del sistema conceptual. Se elabora el sistema conceptual que facilitará el desarrollo del trabajo.
- Compilación del corpus. En este caso la lengua base es la española por ser la prioritaria en el área. Al final del trabajo se tendrán tres corpus uno en cada lengua: español, inglés y chino.
- Elaboración de la terminología. Se desarrollan las definiciones y fichas terminológicas.
- Revisión. Se revisa el trabajo que se ha llevado a cabo hasta el momento antes de que sea traducido a los otros idiomas.
- Establecimiento de equivalencias. Una vez comprobadas las labores hasta ahora, los expertos en traducción se encargan de establecer las equivalencias a otras lenguas.
- Traducción y resolución de los casos problemáticos. Se resuelven los casos que pueden suponer algún problema a la hora de realizar las traducciones.
- Edición. Por último se realiza el proceso de edición del trabajo.

Es posible consultar el proceso que se ha llevado a cabo de manera detallada en otros trabajos de los autores del proyecto [82].

Para poder realizar el trabajo terminológico se ha compilado un corpus textual especializado que sirve de base al desarrollo de la propuesta de categorización de manera que se puedan vincular los términos a los conceptos del sistema conceptual elaborado para el proyecto.

El objetivo de Olivaterm es proporcionar un vocabulario multilingüe de especialidad capaz de responder a cualquier consulta de un usuario. Cualquier terminología tiene diferentes tipos

de usuarios, desde especialistas en la disciplina descrita, hasta usuarios interesados en conocer un poco más la materia que se trata.

Al tratar la terminología del aceite de oliva ha sido necesario recopilar términos de todos los ámbitos relacionados con los procesos de elaboración y comercialización de aceites de oliva. Se presenta en la figura 2.11 un esquema de los ámbitos que se abarcan en el proyecto.

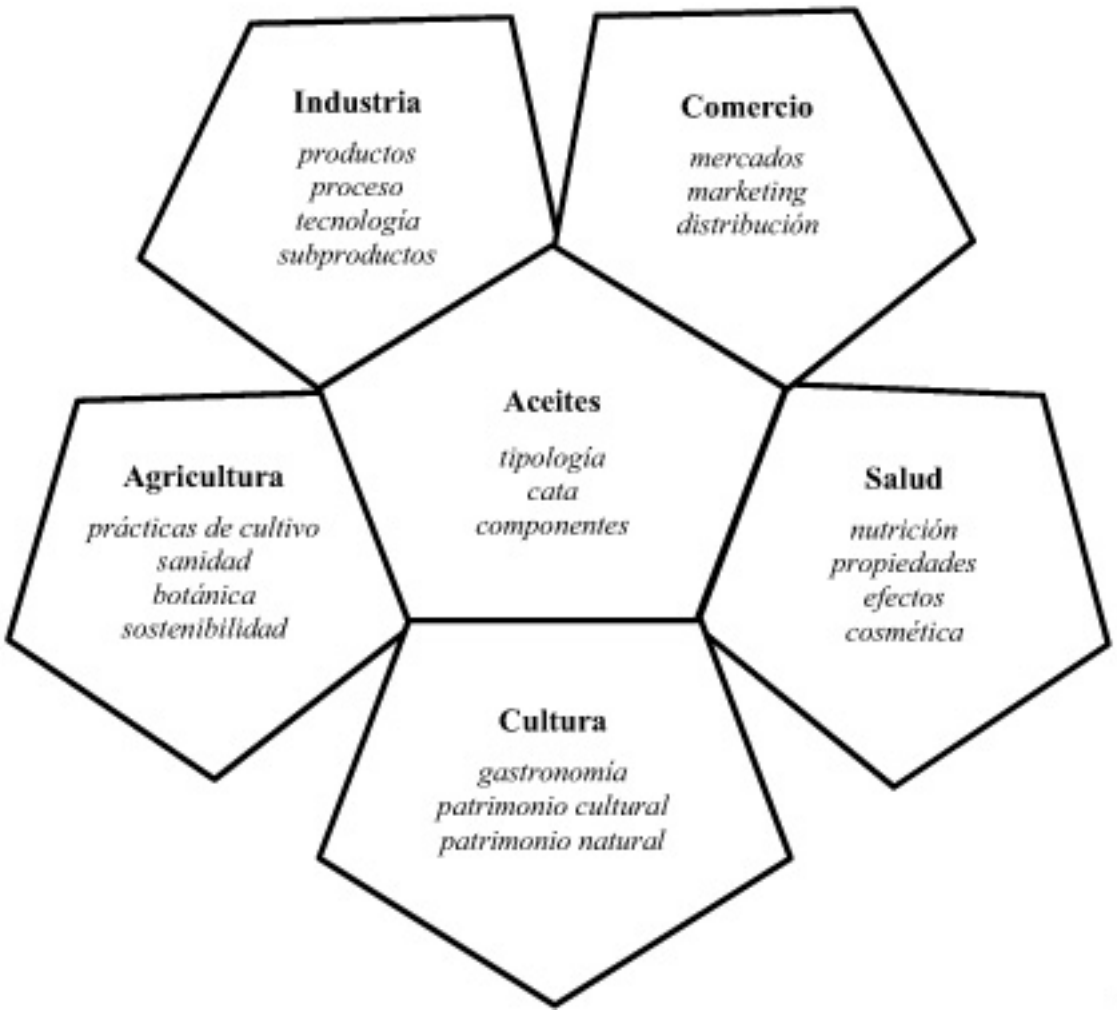

Figura 2.11: Sistema conceptual de Olivaterm

El diagrama presentado en la figura 2.11 pretende visualizar el alcance del proyecto en

cuanto a las disciplinas que se han tenido en cuenta a la hora de desarrollar la terminología en el proyecto. Se ha determinado considerar la terminología vinculada a seis ámbitos: *aceites, agricultura, industria, comercio, salud* y *cultura*. Según nos muestra el sistema conceptual (2.11) todos los ámbitos tienen relación directa con el de *aceites*. Además, se relacionan de manera directa con su adyacente en el sistema conceptual. El ámbito de *industria* se relaciona también de manera directa con *cultura* y *salud*, aunque esto no está explícito en el diagrama. Según el esquema todos los términos quedan relacionados entre ellos por el ámbito nuclear denominado *aceites*.

Olivaterm es un banco de términos que, como ya se ha comentado, busca estandarizar las unidades de conocimiento especializado sobre el aceite de oliva en español, inglés y chino. Para poder hacerlo se parte de un corpus textual en tres idiomas.

Gracias al trabajo terminológico llevado a cabo en el proyecto Olivaterm, se dispone de un diccionario en forma de banco de términos soportado sobre un servicio web. La ontología propuesta en el presente trabajo se basa en parte de la terminología compilada y descrita en este proyecto.

Capítulo 3

Integración de ontologías y servicios web

En el presente capítulo se dará una perspectiva de los trabajos relacionados en el ámbito de las ontologías y la semántica web. Consta de tres partes: en la primera se dan una serie de trabajos relacionados con el tema que se está tratando, en la segunda parte se ofrece el análisis y diseño de la ontología: tipología necesaria, metodología a seguir, herramientas utilizadas, ... Además se describe paso a paso la metodología utilizada. En la tercera parte se desarrollan las ontologías según la metodología que se explica en la sección previa y el servicio web que se va a utilizar.

3.1. Trabajos relacionados

Las circunstancias que rodean la web tradicional hacen que carezca de significado semántico. Las ontologías facilitan la representación del conocimiento gracias a clases y relaciones entre ellas. Si combinamos los conceptos definidos obtenemos la solución a la carencia de la web tradicional constituyendo los inicios de la web semántica.

Cuando se discute la cuestión de la web semántica las ontologías juegan un papel muy importante, facilitan el vocabulario necesario para realizar el intercambio e interpretación de los datos e información en la web, además de ofrecer el significado que tanto se desea en la web. A continuación se ofrecen una serie de usos de las ontologías en los sistemas de información de la web semántica.

Búsqueda semántica

La búsqueda semántica es una de las aplicaciones más codiciadas de la web semántica y una de las más extendidas para las ontologías. Existen diversos enfoques, que podrán ser combinados si es necesario:

1. Considerar una fuente. Lo habitual es que existan varias fuentes disponibles, pero serán los metadatos definidos por la ontología los que seleccionen cuál es la fuente que mejor se adapta a la coyuntura. Es posible que el sistema que valora la petición estime necesario dividir la cuestión en varias partes y enviar cada parte a la fuente que considera más adecuada, existiendo la posibilidad de que tengan que evaluarse en diferentes lenguajes. Cuando le sea devuelta la información deberá volver a unir las partes separadas y reintegrar las respuestas que ofrecerá al cliente.

2. Basándose en consultas de acceso a la información estudiada. Este enfoque se utiliza fundamentalmente si existen agentes humanos en el estudio y cuando el dominio de información relativamente desconocido es explorado.

En la actualidad las búsquedas se realizan sobre textos indexados con sistemas de recuperación de información. Las ontologías ofrecen un beneficio extra aprovechando su capacidad de procesamiento y evaluación de la consulta [39].

Portales semánticos

Un portal web es un lugar único en Internet que reúne información y la presenta obte-

niéndola de diferentes fuentes y exponiéndola de una sola manera, habitualmente acumula y representa información sobre un dominio facilitando el trabajo de una comunidad orientada a un asunto común; puede representar la información necesaria para un enfoque colaborativo de un equipo dedicado a esa tarea [39]. Se puede decir que tiene las siguientes funcionalidades:

- Proporciona información: los usuarios mantienen un flujo de comunicación activa.
- Gestión de información: los administradores pueden integrar y mantener las fuentes de información e incluso establecer los enlaces entre documentos del portal de manera automática.
- Busca información de manera unificada sobre el contenido proporcionado.
- Personaliza configuraciones individuales para el diseño, selección de contenidos, modo de navegación, ...

Si se comparan los portales web tradicionales y los considerados semánticos, los segundos utilizan un dominio ontológico central con una estructura adecuada para la organización, navegación, gestión, etc.; contienen mecanismos de búsqueda y lenguajes orientados a web semántica para la gestión interna de datos. Algunos ejemplos de marcos para la creación de portales semánticos son OntoWeaver [58], Ontoviews [66], SEAL [65] o HealthFinland [95].

Integración de información semántica

La integración de datos e información como base para la reutilización, respuesta de consultas de diversas fuentes y sistemas interoperables es uno de los objetivos fundamentales para la web semántica [39]. Existen dos enfoques:

1. La forma tradicional. En este caso, los datos o sistemas no están diseñados para interoperar, deben ser solicitados en una interfaz e integrar las piezas de información entre

distintos orígenes.

2. Enfoque novedoso. Vinculado a datos donde se utiliza la web semántica para fomentar la reutilización y vinculación de datos en contextos de aplicación.

Asesoramiento de sistemas inteligentes

Un sistema basado en conocimiento o sistema experto pretende simular el comportamiento que humanos expertos tienen en un área concreta. Por lo general, cada sistema contiene una base de conocimiento con experiencia declarativa acumulada y conjuntos de reglas aplicadas junto con la base de conocimiento a cualquier área [39].

Este tipo de sistemas ayudan en la toma de decisiones gracias a la recuperación inteligente de información, ayuda a analizar los datos complejos, comprueba restricciones complejas y sugieren soluciones alternativas al usuario para continuar con el análisis.

Tanto la recuperación inteligente de información y el análisis de datos son tareas pertenecientes al área de la búsqueda e integración de la información semántica. La comprobación de restricciones y la sugerencia de soluciones suelen ser implementadas de manera tradicional, aunque incrementan su funcionalidad gracias a la tecnología de la web semántica.

Un ejemplo de este tipo de aplicaciones son Hospital Care Watch [17] o IASO system [78].

Middleware semántico

En los últimos años la informática ha evolucionado del software compacto al dividido, desarrollando distribuciones concretas que van desde la virtualización del almacenamiento y procesamiento de subsistemas, el conocido Cloud Computing, a la composición de flujos de trabajo complejos a partir de servicios web accesibles; en estos ejemplos las aplicaciones e interfaces de usuario se desacoplan de los sistemas operativos. Todas las implementaciones

técnicas de los paradigmas anteriores son conocidos como componentes middleware y deben hacer frente a ciertos retos en el momento de ejecución [39]:

- Localizar los subsistemas disponibles más adecuados. Este aspecto puede ser mejorado gracias a la búsqueda semántica.
- Intercambio seguro de mensajes. El establecimiento de una comunicación entre sistemas conectados puede ser enriquecido por integración semántica.
- Gracias a la programación declarativa e incluso al razonamiento podrá mejorarse la ejecución de un control del sistema sin tener un adecuado paradigma de control.

Cuando se lleva a cabo middleware se combinan diversas tecnologías semánticas como servicios web semánticos [46, 53, 55, 56, 57, 80, 108], redes semánticas, peer-to-peer semántico [9, 45] o cloud semántico [106].

3.2. Análisis y diseño

A la hora de comenzar el proceso de análisis y diseño de la ontología lo primero que se plantea es definir el tipo de ontología que se necesita. Según la clasificación hecha en la sección 2.1.3 se determina que para modelar el dominio será requerida una ontología de dominio. Las ontologías de dominio son aquellas que describen de manera precisa una determinada área de conocimiento especializado. En el caso que nos ocupa se trata del dominio de la calidad del aceite de oliva.

Una vez establecido el tipo de ontología, el siguiente paso es definir la metodología que se va a utilizar para su diseño, así como el lenguaje en el que se desarrollará y las herramientas utilizadas para ello. Partiendo de las metodologías descritas brevemente en la sección 2.1.2.1

se ha optado por utilizar la metodología desarrollada por Noy & McGuinness [76] denominada "Ontology 101 development process". Se ha seleccionado esta metodología por ser la más actual (2005), además, establece en sus principios que se trata de una metodología especialmente apta para los desarrolladores principiantes en el área de las ontologías, es sencilla de comprender y contiene la mayor parte de las etapas que también están descritas por otras metodologías, pero de manera más comprensible para los diseñadores noveles. Asimismo se trata de un desarrollo del equipo de la Universidad de Standford al igual que la herramienta Protégé.

La herramienta escogida para el desarrollo de la ontología es Protégé[1]. Se ha elegido este software por ser de licencia libre, por su facilidad de uso y por ser conocido por los desarrolladores.

La metodología "Ontology 101 development process" cuenta con las etapas registradas en la figura 2.8. Los autores definen antes de comenzar con la metodología tres reglas a tener en cuenta cuando se diseña:

No existe una forma correcta de modelar un dominio. Cuando se diseña una ontología su desarrollo depende del uso que vaya a dársele y del mantenimiento de la misma.

El desarrollo de ontologías es un proceso *iterativo.*

Los conceptos en la ontología deben ser cercanos a los objetos y relaciones en el dominio de interés. Los *sustantivos* suelen ser los *objetos* y las *relaciones* suelen ser los *verbos* de las frases que describen el dominio.

A continuación se describen con más detalle dichas etapas.

- Determinar el dominio y alcance de la ontología. Para llevar a cabo el trabajo correspondiente a esta primera etapa se definirán las preguntas de competencia que serán

[1]http://protege.stanford.edu/

las encargadas de ayudar a concretar el dominio y alcance mencionados anteriormente; durante el desarrollo de la ontología es posible que las respuestas a estas preguntas cambien, ya que conforme se vaya diseñando es posible que el alcance previsto sea mayor o menor. Las preguntas de competencia son aquellas a las que la ontología debe dar respuesta y servirán al final del proceso para comprobar si es de la calidad esperada.

- Considerar la reutilización de ontologías existentes. Normalmente merece la pena comprobar lo que se ha hecho sobre el dominio que se está tratando para así intentar reutilizar el trabajo de otras personas. Las ontologías se fundamentan en el conocimiento compartido y la reutilización de las mismas es su significado puro. Existen diversos bancos donde se puede revisar si la ontología sobre el dominio que se va a describir ha sido modelada previamente: Ontolingua [2], la biblioteca de DAML [3], UNSPC [4], DMOZ [5], RosettaNet [6], Swoogle[7].

- Enumerar los términos importantes para la ontología. Es muy recomendable especificar una lista con los términos que el experto estima fundamentales para la comprensión del dominio por parte del usuario.

- Definir las clases y la jerarquía entre ellas. La jerarquía va a depender del enfoque dado a la ontología, de cuál vaya a ser su uso y a quién vaya dirigido, entre otras cosas. Uschold y Gruninger defienden que existen varios enfoques a tener en cuenta cuando se desarrolla una ontología[98]: *top-down:* en este enfoque se comienza definiendo los conceptos más generales del dominio y se va especializándolos poco a poco; *bottom-up:* en este enfoque se comienza con los conceptos más especializados del dominio y se van definiendo "hacia arriba" de manera que se termina con los conceptos más generales;

[2]http://ksl.standofrd.edu/software/ontolingua/
[3]http://www.daml.org/ontologies
[4]www.unspsc.org
[5]www.dmoz.org
[6]www.rosettanet.org
[7]http://swoogle.umbc.edu/

combinado: en este caso se combinan los enfoques anteriores.

Ninguno de los enfoques anteriores suele utilizarse más que otro, sin embargo el último, combinado, suele ser más común, pues los conceptos intermedios son los más utilizados por los usuarios [84]. La forma más habitual de proceder es definir las clases a partir de la lista creada en el paso anterior, para más tarde seleccionar los términos que describen objetos cuya existencia es independiente de otros. Después se organizan las clases de manera jerárquica así, *dadas dos clases A y B, B es un concepto "tipo-de" A, siempre que A sea superclase de B → cada instancia de B será también instacia de A.*

A la hora de definir la jerarquía de clase es recomendable ir comprobando su consistencia, se ofrecen una serie de recomendaciones:

- Asegurar la jerarquía de clases. Cuando se establece una jerarquía de clases es habitual que las relaciones vengan determinadas por relaciones "is-a". *Cuando todas las instancias de una clase B son también instancias de una clase A → B es subclase de A.* Las clases pueden cumplir la propiedad transitiva al ser una jerarquía, así, *siendo B una subclase de A y C una subclase de B → C será subclase de A.* Otro aspecto a tener en cuenta cuando se nombran clases es que estos nombres representan conceptos del dominio y no las palabras que aluden a esos conceptos, de manera habitual los términos definidos en el dominio tendrán sinónimos, es importante recordar que éstos no significan la creación de nuevas clases. Es notable que se eviten los ciclos entre clases, ya que estas serían equivalentes: *dada una clase A que tiene como subclase a B, y B que tiene como subclase a A → A y B son equivalentes.* Es recomendable no utilizar singulares y plurales del mismo concepto en la jerarquía y menos aún si se utiliza la clase singular como subclase de la plural, para evitar esto se dan más adelante una serie de convenciones a la hora de definir nombres.

- Considerar las clases hermanas. Se denominan clases hermanas a aquellas que de-

penden de la misma superclase de manera directa, este tipo de clase ha de tener el mismo nivel de generalidad. Es bueno tomar en consideración casos extremos en el modelado. Por ejemplo, si una clase solamente tiene una subclase es lógico que exista algún problema de modelado; también está el caso contrario que existan demasiadas subclases (doce o más), en ese caso se debe plantear alguna clase intermedia; claro que estos comentarios son recomendaciones, es posible que no sean necesarias subclases intermedias o deba existir solamente una clase dependiente de otra. Todo depende de la realidad del dominio que se esté modelando.

- Estudiar la herencia múltiple. Es normal que en una jerarquía que pretende modelar el mundo real existan casos de clases que dependen de varias superclases. Esto no es un problema al diseñar una ontología, de hecho, hay que tener en cuenta dichas relaciones y representarlas.

- Comprobar la inserción de clases nuevas. Cuando se está desarrollando una ontología se puede tender a realizarla demasiado anidada o demasiado plana, complicando así la navegación a través de ella. Este es uno de los aspectos que más se deben tener en cuenta, y existen una serie de pautas a tener en cuenta para saber si establecer una nueva clase o no, las subclases por lo general:

  tienen propiedades que la superclase no tiene,

  tienen restricciones diferentes a las que tiene la superclase,

  se relacionan con diferentes clases que la superclase,

  Lo mejor será la armonía entre las clases necesarias para la comprensión y la creación de demasiadas clases.

- Verificar la adecuación de nuevas clases o valores de propiedades. En este caso se debe tener en cuenta si será necesario definir otra clase o solamente permitir varios valores en la propiedad correspondiente. Existen ciertas directrices que facilitan esta cuestión:

Si los conceptos que tienen diferentes valores en una propiedad son restricciones para diferentes propiedades en otras clases, se debe crear una clase. En otro caso lo mejor es modelarlo como un valor de propiedad.

Si en el mundo a diseñar se piensa en los objetos con diferentes valores y la distinción es importante para diferenciar los conceptos y objetos en el mundo real, se debe crear una nueva clase.

Por lo general los colores, números, localizaciones no suelen ser nuevas clases sino valores de propiedad

- Constatar la adaptación de instancias o clases. Diferenciar entre instancia y clase depende del nivel de detalle que se le quiera dar a la ontología y de la utilidad que vaya a tener. Habiendo llevado a cabo el proceso de manera correcta, es fácil determinar hasta que punto llegará la ontología gracias a las preguntas de competencia establecidas previamente, las respuestas a esas preguntas ofrecerán los conceptos que serán candidatos a instancias. Las instancias son los conceptos más específicos que se van a representar en la ontología. Los conceptos anteriores que puedan formar una jerarquía podrán ser clases en lugar de instancias, es decir, solamente si existe una jerarquía natural entre los términos que se están definiendo será recomendable establecerlos como clases aunque carezcan de instancias individuales.

- Confirmar la limitación del alcance. Es importante tener en cuenta que cuando se lleva a cabo una ontología no es necesario modelar todo el conocimiento, sino aquél que sea necesario para la aplicación que se le va a dar.

- Analizar las subclases disjuntas. Se dice que dos clases son disjuntas si no tienen instancias en común. El hecho de declarar las clases disjuntas permite valorar mejor la calidad de la ontología.

- Definir las propiedades de las clases: slots. Para que se pueda ofrecer una respuesta a

las preguntas de competencia establecidas en la primera etapa no basta con definir una clase, ésta deberá relacionarse con otras. Una vez seleccionadas las clases de la lista de términos confeccionada previamente la mayoría de los términos restantes serán, probablemente, propiedades de esas clases (slots). Debe determinarse qué describe cada una de esas propiedades. Además, una subclase hereda las propiedades de su clase padre. Existen distintos tipos de propiedades: *propiedades intrínsecas*, *propiedades extrínsicas*, *partes de un objeto*, que podrán ser físicas o abstractas; y *relaciones* con otros individuos.

- Definir las restricciones de las propiedades: facetas de los slots. Las propiedades pueden adquirir diferentes valores (tipo de valor, cardinalidad, ...). A continuación se observan las facetas más comunes: la cardinalidad describirá cuántos valores puede tener un slot, en algunas ocasiones será interesante registrar el valor mínimo y máximo que se puede almacenar. Un slot puede ser un string, boolean, instance, number, enumerated.... Cabe destacar los casos de: *enumerated* que se utilizará para listas de valores admitidos por ese slot; e *instance* que admite la definición de relaciones entre individuos, además, los slots que admitan este tipo de valor deberán definir una lista de clases admitidas. Se define como dominio de un slot las clases cuyas propiedades son descritas por un slot. Se define como rango de un slot a las clases admitidas para los slots de tipo instance.

  Cabe destacar ciertos aspectos cuando se definen las propiedades y restricciones: se deben evitar las relaciones inversas, esto es: no almacenar información en dos sentidos, cuando un valor depende de otro no es necesario especificar la relación en sentido contrario; existe la posibilidad de asignar un valor por defecto a una propiedad de una clase, así, cuando la mayoría de los slots de una clase contengan el mismo valor para una propiedad, ésta se rellenará por defecto al crear la instancia de la clase, dando la opción de ser modificada para aquellos valores que no son el establecido por defecto. Sin embargo, cabe destacar que esta última propiedad es diferente del valor del slot que no puede ser modificada.

- Crear instancias. En esta última etapa se definen las instancias individuales de cada clase. Para ello se debe elegir una clase, crear una instancia de ella y rellenar los valores de los slots correspondiente.

Por último cuando se desarrolla una ontología es recomendable seguir el mismo criterio a la hora de nombrar los objetos, se facilitan a continuación una serie de recomendaciones para ello:

- Mayúsculas, minúsculas y delimitadores: suelen escribirse en mayúsculas los nombres de clases y en minúsculas los nombres de slots. En el caso de que el nombre contenga varias palabras existen varias opciones a elegir por el usuario: utilizar espacios entre ellas; escribir todas las palabras juntas y comenzar cada palabra en mayúsculas; utilizar un delimitador como guiones, en este caso también se ha de decidir si se utilizan mayúsculas al comenzar cada nueva palabra.

- Singulares y plurales: en este caso no existe una alternativa mejor, pero si se recomienda ser consistente a lo largo de todo el desarrollo.

- Otras consideraciones: no se recomienda añadir las palabras: “clase”, “slot”, “propiedad”, tampoco abreviar los nombres ya que resultarían poco descriptivos, para los nombres de subclases también se recomienda coherencia a la hora de nombrarlos, si se se decide añadir el nombre de la clase se debe añadir en todas las subclases e igual para el caso de no añadirlo.

Cabe destacar que en ocasiones estas recomendaciones dependen de cómo funciona la herramienta con la que se va a desarrollar la ontología.

3.3. Desarrollo e implementación

En las próximas secciones se planteará el diseño y desarrollo de un sistema basado en ontologías y soportado sobre un servicio web. Se pretende clasificar un aceite de oliva a partir de los componentes y propiedades que facilite el usuario del servicio. Esto será posible gracias al conocimiento inferido por las ontologías diseñadas.

La primera parte será diseñar la ontología de dominio, a la que seguirá el diseño de la ontología de aplicación y se finalizará con el diseño del servicio web en el que serán integradas las partes anteriores.

3.3.1. Desarrollando la ontología de dominio

En esta sección se desarrolla la ontología de dominio que será la que realice la clasificación de los aceites de oliva según el conocimiento declarado previamente.

3.3.1.1. Preguntas de competencia

Como se viene comentando cuando se desarrolla una ontología lo primero que se debe hacer es delimitar el dominio y alcance de la misma. En nuestro caso el dominio elegido es la calidad del aceite de oliva, se planea utilizar la ontología como herramienta para la correcta clasificación de cualquier aceite. Gracias a ella los usuarios podrán conocer cómo se clasifica un aceite en función de sus componentes y propiedades organolépticas.

Se estima que el uso que se le dará será por parte de usuarios interesados en este ámbito: desde un usuario experto hasta un usuario lego interesado en conocer un poco más el ámbito del aceite de oliva.

Por esto se definen las siguientes preguntas de compentencia:

- ¿Cuál es el dominio a modelar?
- ¿Qué uso se le dará a la ontología?
- ¿Quiénes son los posibles usuarios?
- ¿Quién mantendrá la ontología?
- ¿Qué propiedades ha de tener un aceite de oliva?
- ¿Cuáles son los componentes más destacables del aceite de oliva?
- ¿Qué tipos de aceite de oliva existen?
- ¿Existen aceites de oliva regulados? ¿Cuáles son?
- ¿Qué se valora en un aceite de oliva?
- ¿Cómo sé de qué aceite hablo?

La ventaja que tienen las ontologías sobre cualquier dominio es que son ampliables y reutilizables, por ello, en el caso de querer ampliar el horizonte del conocimiento que se va a inferir, en nuestro caso, sobre el aceite de oliva, sería posible realizar ampliaciones en el futuro para el ámbito específico necesario y pudiendo reutilizar el conocimiento adquirido ya por la ontología realizada. En el trabajo que nos ocupa actualmente se pretende obtener la calidad del aceite de oliva. Estas preguntas de competencia cubren a la perfección ese objetivo.

3.3.1.2. Consideración de reutilización de ontologías

Una de las características más destacables de las ontologías es su reutilización, permitiendo a otros usuarios modificar o introducir más conceptos. Es posible que interese reutilizar

solamente una parte de la ontología cosa que sería posible.

En nuestro caso, después de examinar diferentes bancos de ontologías, se han encontrado algunas que tratan sobre el aceite de oliva, pero ninguna que tenga en cuenta su calidad de una manera más concreta. Por ello se ha desestimado la opción de reutilizar una de las ontologías encontradas.

3.3.1.3. Enumeración de términos importantes

Se ha de establecer una lista de términos que se consideran importantes a la hora de evaluar la calidad del aceite de oliva:

componentes mayoritarios, componentes minoritarios, ácidos grasos, ácidos grasos libres, acidez, propiedades organolépticas, atributos, sabor, sabor elemental, flavor, textura, grasa, grasa alimentaria, aceite, aceite vegetal, aceite de oliva, categorías comerciales, aceites de oliva vírgenes, aceites de oliva no vírgenes, aceites de orujo de oliva, aceite de oliva virgen extra, aceite de oliva virgen, aceite de oliva lampante, aceite de oliva, aceite de oliva refinado, aceite de orujo de oliva, aceite de orujo de oliva bruto, aceite de orujo de oliva refinado, variedades principales, arbequina, cornicabra, empeltre, hojiblanca, farga, lechín de Granada, lechín de Sevilla, manzanilla cacereña, manzanilla de Sevilla, morisca, morrut, picual, picudo, verdial de Badajoz, verdial de Huévar, almendra verde, almendrado, alpechín, atrojado, avinado, basto, cocido-quemado, dulce, esparto, frutado, frutado maduro, frutado verde, gusano, heno-madera, hierba, hojas verdes, lubricante, manzana, metálico, moho-humedad, pepino, picante, amargo, rancio, salmuera, tierra, untuoso.

Es posible que a lo largo del desarrollo surgan términos que es necesario introducir para poder dar al sistema la funcionalidad que se desea. En las próximas secciones se establecerá la jerarquía de clases entre los términos de la lista anterior.

Cuando se desarrolla una ontología es necesario conocer bien la terminología utilizada en el ámbito a modelar, en este caso, la terminología utilizada ha sido extraída gracias al proyecto Olivaterm (2.3.3). Para su correcta definición se ha seguido el proceso terminológico explicado en la sección 2.3.2

3.3.1.4. Definición de clases y jerarquía de clases y definición de las propiedades de las clases

Una vez seleccionados los conceptos que se van a utilizar se ha de establecer una jerarquía de clases. Para este caso se ha decidido tomar un enfoque top-down, partiendo de los conceptos más generales especificando poco a poco.

Teniendo en mente la lista de términos definida en la sección anterior (3.3.1.3) se deberán seleccionar aquellos que describen objetos de manera independiente estableciendo así cuáles son las clases. Una forma buena de distinguir clases de atributos es preguntándose sobre el objeto. Si la pregunta tiene un sólo argumento será una clase, si tiene dos será un atributo.

Después de realizar las acciones necesarias para establecer la jerarquía de clases, se ha determinado que las clases son: *componentes mayoritarios, componente minoritarios, ácidos grasos, ácidos grasos libres, acidez, propiedades organolépticas, atributos, sabor, sabor elemental, flavor, textura, grasa, grasa alimentaria, aceite vegetal, aceite de oliva, categorías comerciales, aceites de oliva vírgenes, aceites de oliva no vírgenes, aceites de orujo de oliva, aceite de oliva virgen extra, aceite de oliva virgen, aceite de oliva lampante, aceite de oliva, aceite de oliva refinado, aceite de orujo de oliva, aceite de orujo de oliva bruto, aceite de orujo de oliva refinado, variedades principales, arbequina, cornicabra, empeltre, hojiblanca, farga, lechín de Granada, lechín de Sevilla, manzanilla cacereña, manzanilla de Sevilla, morisca, morrut, picual, picudo, verdial de Badajoz, verdial de Huévar.*

Una vez establecidas estas clases se debe determinar la jerarquía entre ellas, así obten-

dremos las clases que son "tipo de" otra clase y las relaciones necesarias para conectar unas con otras. Es recomendable comprobar la consistencia de la ontología en este paso, pues si no está definida de manera correcta el conocimiento inferido no será el esperado. En nuestro este caso se tienen tres clases independientes:

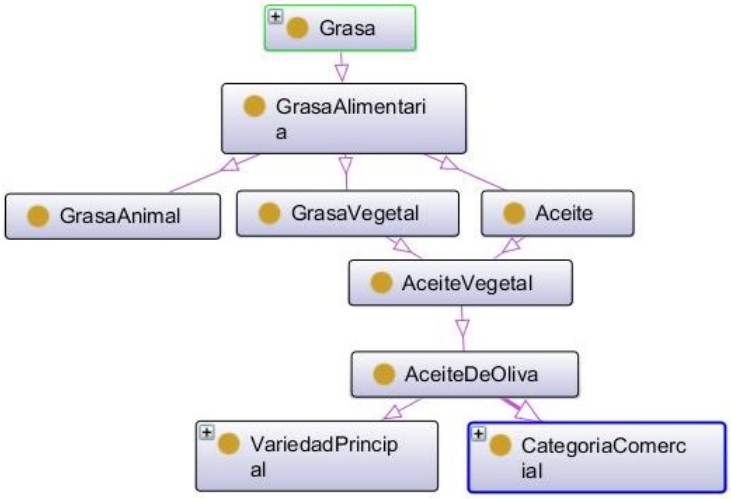

Figura 3.1: Jerarquía de subclases de la clase Grasa

La imagen 3.1 muestra la jerarquía de clases de la clase "Grasa". Esta clase es la más importante, pues contiene los principales elementos del sistema, que son las categorías por las que se clasificará el aceite de oliva según el conocimiento inferido. La clasificación nombrada, que será explicada más adelante, se corresponderá con las subclases de "VariedadPrincipal" y "CategoríaComercial". Existe un elemento destacable en este caso, la clase "AceiteVegetal" tiene dos padres, uno "Aceite" y otro "GrasaVegetal", se ha optado por definirlo así debido a que el aceite puede ser vegetal o animal, pero el caso que nos ocupa es el primero, esta es la manera en la que se ha decido especificar que el aceite vegetal es un aceite de origen vegetal.

En la imagen 3.2 se establece la jerarquía para la clase que define los principales componentes del aceite de oliva. Estos componentes son los que establecen las propiedades nutricionales del aceite de oliva y servirán para determinar la tipología del aceite de oliva.

Figura 3.2: Jerarquía de clases de la clase Componente

En la imagen 3.3 se observa la jerarquía de clases para los atributos que definen el sabor, olor, aroma, ..., del aceite de oliva. Esta jerarquía es importante a la hora de realizar la futura clasificación.

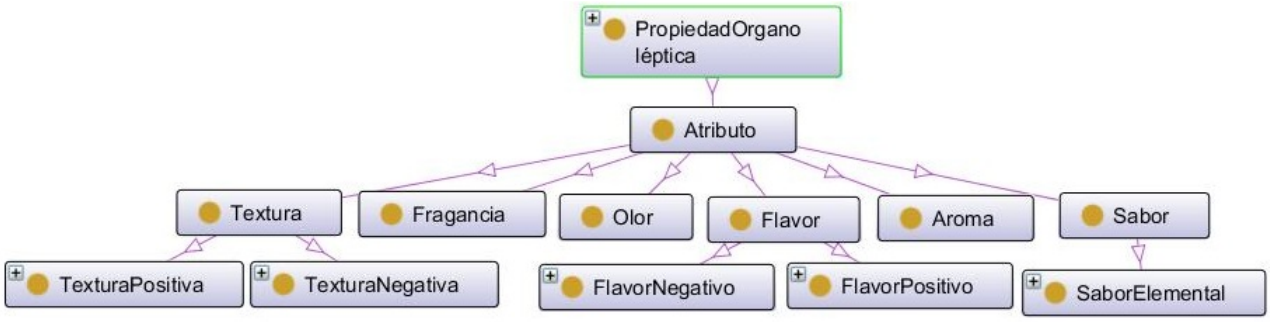

Figura 3.3: Jerarquía de clases de la clase Propiedad Organoléptica

Todas las subclases de las imágenes 3.1, 3.2 y 3.3 son relaciones del tipo “is-a”.

El siguiente paso es establecer relaciones entre estas clases aparentemente independientes. Nos centramos en la subclase “AceiteDeOliva”, como se ve en la imagen 3.4 el aceite de oliva “contiene” “Componente” y “EsEvaluadoPor” “PropiedadOrganoléptica” y está “hecho” por “VariedadPrincipal”. Estas relaciones serán heredadas por las subclases de “AceiteDeOliva”. Además, la clase “PropiedadOrganoleptica” se vincula con “VariedadPrincipal” por la relación define.

Figura 3.4: Relaciones entre las clases

Estos son los aspectos generales de la ontología definida. De manera más específica para poder establecer qué tipo de aceite de oliva se está tratando necesitamos conocer el nivel de acidez de un aceite y sus propiedades organolépticas, pudiendo realizar la siguiente clasificación:

	Acidez	Atributos	Clasificación
Apto para el consumo	≤0.8	Positivos, no tiene defectos	aceite de oliva virgen extra
	≤2	Positivos (con leves defectos)	aceite de oliva virgen
	≤1	Normalmente positivos, aunque puede tener negativos	aceite de oliva
	≤1		aceite de orujo de oliva
No apto para el consumo	≤0.3	Normalmente negativos, aunque puede tener positivos	aceite de oliva refinado
	≤0.3		aceite de orujo de oliva refinado
	≤3.3		aceite de oliva lampante
	Ilimitada		aceite de orujo de oliva bruto

Figura 3.5: Tabla de clasificación

Como se ve en la tabla 3.5 para poder clasificar un aceite de oliva será necesario conocer su índice de acidez y sus propiedades organolépticas. Se han establecido las subclases necesarias tal y como se muestra en la figura 3.6, además, en la taxonomía de componentes también se han definido las clases y atributos necesarios para conocer el índice de acidez del aceite de oliva 3.2, así como los atributos y valores que pueden tomar sus propiedades organolépticas 3.7.

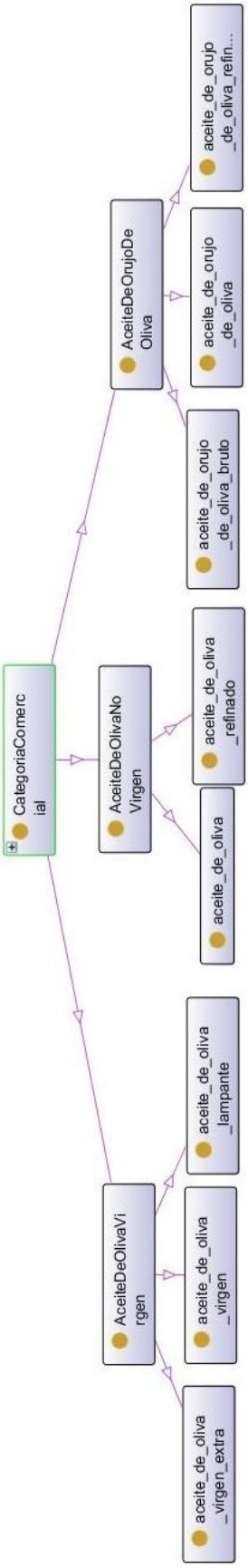

Figura 3.6: Categorías comerciales

Gracias a los atributos que se muestran en la imagen 3.7 es posible determinar de manera aproximada si el aceite de oliva se ha obtenido de alguna de las variedades principales definidas en la ontología, pudiendo incluso inferir que si no se corresponde con ninguno de los tipos definidos, probablemente se trate de un aceite obtenido por diversas variedades.

3.3.1.5. Definición de las restricciones de las propiedades

Las relaciones definidas previamente deben tener restricciones: la cardinalidad, dominio y rango, que serán definidas a continuación junto con una descripción para su mejor comprensión:

"Contiene". Es la encargada de conectar la parte de "Componente" con "AceiteDeOliva". Tiene cardinalidad 1:1. Así, un individuo de cualquiera de las subclases de "AceiteDeOliva" contendrá un individuo de cualquiera de las subclases de "Componente". Por ejemplo: aunque en este trabajo no se establecen individuos para las subclases de "AceiteDeOliva" supongamos cualquier marca de aceite de oliva virgen extra, contendrá una instancia de cualquier individuo de la clase componentes, un aceite de oliva virgen extra cuenta con una acidez (figura 3.8).

"Evaluado". Coordina "AceiteDeOliva" con "PropiedadOrganoleptica". Su cardinalidad es 1:N. Un individuo de una de las subclases de "AceiteDeOliva" estará evaluado por uno o varios individuos de cualquier subclase de "PropiedadOrganoléptica". Por ejemplo: un individuo ficticio instancia de "aceite_de_oliva_virgen_extra", estará evaluado por una o más instancias de un flavor positivo, frutado verde o frutado verde y almendra verde.(figura 3.9).

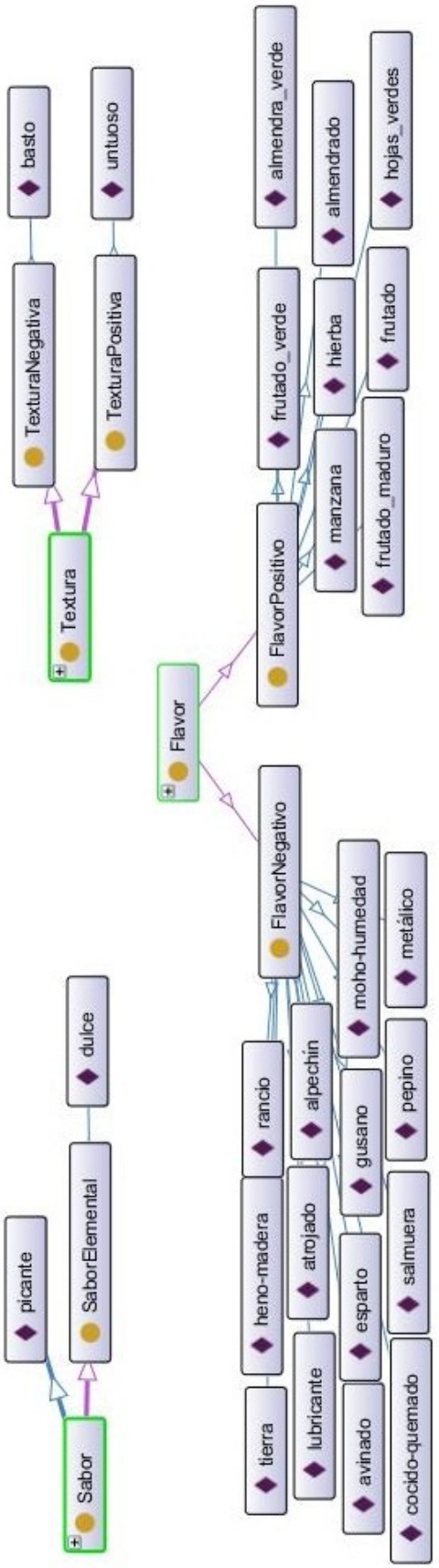

Figura 3.7: Atributos del aceite de oliva

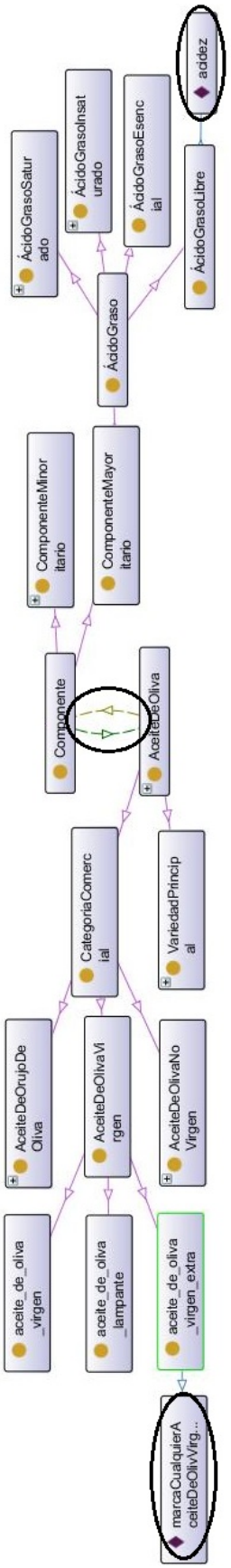

Figura 3.8: Relación "Contiene"

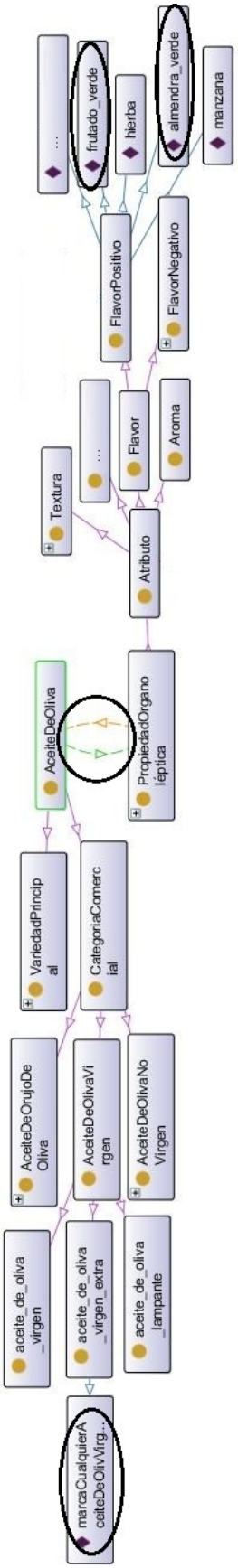

Figura 3.9: Relación "Evaluado"

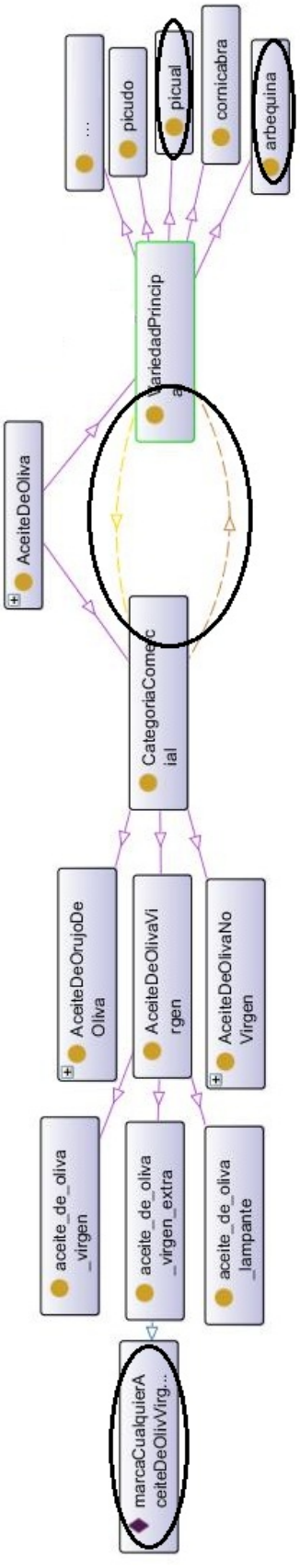

Figura 3.10: Relación "Hecho"

"Hecho". Vincula "CategoriaComercial" con "VariedadPrincipal". Su cardinalidad es 1:N. Cualquier individuo de una subclase de "CategoriaComercial" estará hecho por uno o varios individuos de "VariedadPrincipal". Por ejemplo: un individuo instancia de "aceite_de_oliva_virgen_extra", estará hecho por una o varias variedades principales, por ejemplo, picual o picual y arbequina.(figura 3.10).

"Definido". Une "VariedadPrincipal" con "PropiedadOrganoleptica". Su cardinalidad es 1:N. Un individuo de "VariedadPrincipal" estará definido por varios individuos de "PropiedadOrganoleptica". Por ejemplo: un individuo instancia de "VariedadPrincipal", picual por ejemplo, es definido por una o varias propiedades organolépticas: picante o frutado.(figura 3.11).

3.3.1.6. Creación de instancias

En el caso que nos ocupa las instancias que se tienen son los individuos de las clases, por ello, estas instancias son de manera general los atributos que se pueden percibir al realizar la cata de un aceite de oliva, mostrados en la figura 3.7: *almendra verde, almendrado, alpechín, atrojado, avinado, basto, cocido-quemado, dulce, esparto, frutado, frutado maduro, frutado verde, gusano, heno-madera, hierba, hojas verdes, lubricante, manzana, metálico, moho-humedad, pepino, picante, amargo, rancio, salmuera, tierra, untuoso.*

Estas instancias se utilizarán en el proceso de determinación del tipo de aceite de oliva que se está tratando y serán muy útiles para poder determinar si se trata de un aceite correspondiente a alguna variedad en concreto o a una combinación de variedades.

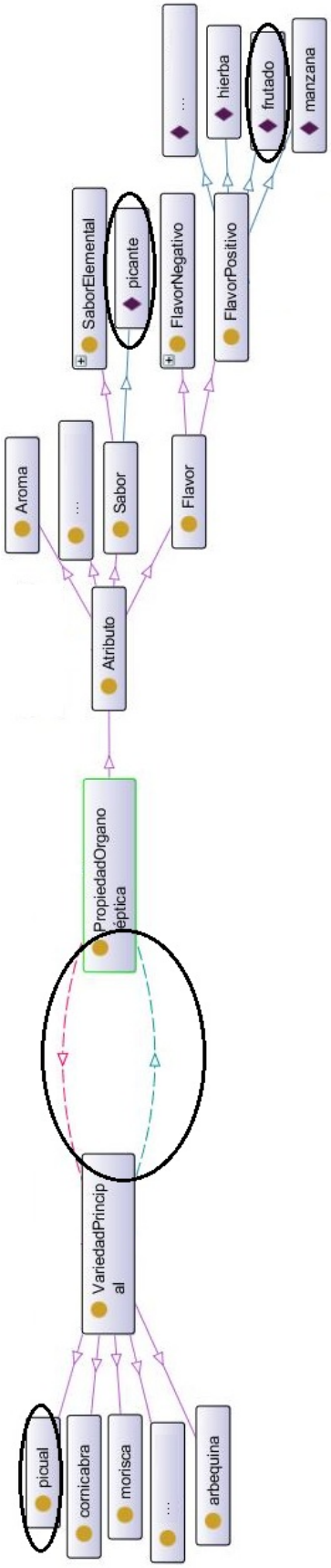

Figura 3.11: Relación "Definido"

3.3.2. Desarrollando e integrando el servicio web

Para poder integrar las ontologías definidas en las secciones 3.3.1 y **??** será necesario definir ciertos elementos que se describen a continuación:

SOAP

Ya se ha comentado que SOAP es un lenguaje etiquetado, todas las etiquetas referentes al lenguaje constan del prefijo: "soapenv". Los documentos SOAP tienen la siguiente estructura representada en la imagen 3.12

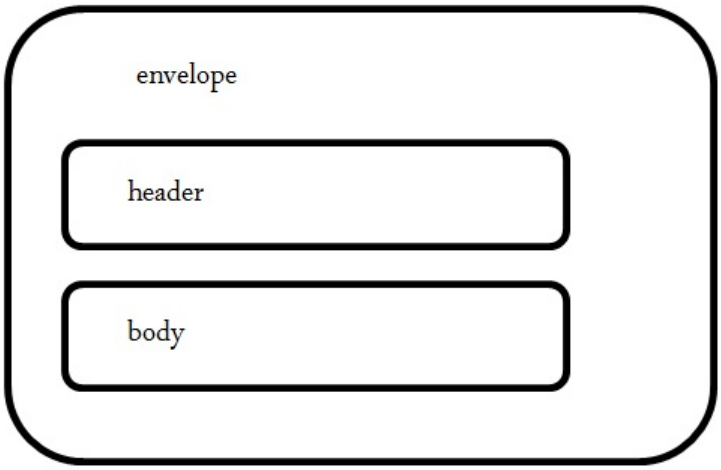

Figura 3.12: Estructura de un documento SOAP

La sección **"Envelope"** comienza con:

```
<soapenv:Envelope xmlns:SOAP-ENV=
        "http://schemas.xmlsoap.org/soap/envelope/">
```

En el código mostrado *"xmlns"* señala el espacio de nombres. El espacio de nombres en los servicios web tiene la misma funcionalidad que en las ontologías, gracias a él, el sistema se asegura la unicidad de las etiquetas. El nombre de ese espacio y la versión de SOAP utilizada es designado por el resto del código.

La sección **"Header"** es opcional, por lo que su formato no está especificado. Se utiliza para transmitir ciertos datos como las contraseñas de usuario. Consta de tres atributos:

- *soapenv:mustUnderstand* tendrá valor 1 cuando se acepte generar un mensaje de error si el servicio web no ha sido programado para manejar campos en esta sección.
- *soapenv:actor* el documento que señale debe visitarse para su procesamiento correcto.
- *soapenv:encodingStyle* indica los tipos de datos, aparecerá en cualquier elemento SOAP y se aplica a todos los elementos hijos.

En la sección **"Body"** se encuentra la parte útil del mensaje. Puede tratarse desde un documento XML que se va a transferir hasta una llamada a un procedimiento. Un ejemplo de esta sección, envía como pregunta a qué aceite de oliva corresponden ese nivel de acidez:

```
Request:
<?xml version="1.0" encoding="UTF-8"?>
 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
      <ns2:Clasificacion xmlns:ns2="http://pack_sw/">
          <acidez>0.7</acidez>
          <almendra_verde>false</almendra_verde>
          <almendrado>false</almendrado>
          <alpechin>false</alpechin>
          <basto>false</basto>
          <avinado>false</avinado>
          <cocido_quemado>false</cocido_quemado>
          <dulce>true</dulce>
          <amargo>true</amargo>
```

```
            <esparto>false</esparto>
            <frutado_maduro>false</frutado_maduro>
            <frutado_verde>false</frutado_verde>
            <gusano>false</gusano>
            <heno_madera>false</heno_madera>
            <hierba>true</hierba>
            <hojas_verdes>true</hojas_verdes>
            <lubricante>false</lubricante>
            <manzana>false</manzana>
            <metalico>false</metalico>
            <moho_humedad>false</moho_humedad>
            <pepino>false</pepino>
            <picante>true</picante>
            <rancio>false</rancio>
            <salmuera>false</salmuera>
            <tierra>false</tierra>
            <untuoso>false</untuoso>
        </ns2:Clasificacion>
    </S:Body>
</S:Envelope>
Response:
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
     <ns2:ClasificacionResponse xmlns:ns2="http://pack_sw/">
      <return>Se trata de un aceite de oliva virgen extra.
       Está extraído de la variedad 'manzanilla de Sevilla'
```

```
        </return>
    </ns2:ClasificacionResponse>
  </S:Body>
</S:Envelope>
```

Dentro de la sección "body" es posible señalar los errores que puedan producirse cuando se realiza una acción. Se marca con la etiqueta "fault". Aparecen una sóla vez, en los mensajes de respuesta y puede ser uno de los siguientes tipos: *faultcode* indica un código con el problema que se está produciendo; *faultstring* indica de manera legible el error que se está produciendo; *faultactor* señala el servicio en el que se produce el error; y *detail* informa del estado del servidor y las variables en el momento en el que se produce el error.

En cuando a los tipos de datos permitidos en SOAP, existen dos tipos: *ComplexType* se compone de varios SimpleType; y *SimpleType* cualquier tipo de dato sencillo: entero, real, cadena, ...

WSDL

Los doscumentos WSDL se dividen en dos secciones definidas claramente: funcional o concreta, en la que se encuentran los elementos orientados a la vinculación física con el cliente: service, port, binding; y no funcional o abstracta donde se hallan los elementos orientados a describir las capacidades del servicio web: types, message, operation, portType. Todas las etiquetas de este documento irán precedidas por "wsdl". La estructura es la siguiente:

Se comenta en primer lugar la sección **"definitions"**. Se trata del elemento raíz del documento. Cualquier espacio de nombres definido en esta sección será global, aunque luego se definan espacios de nombres locales.

La sección **"types"** define los tipos de datos usados en un servicio web utilizando la sintaxis de XML. SOAP permite solamente una entrada y una salida en los mensajes.

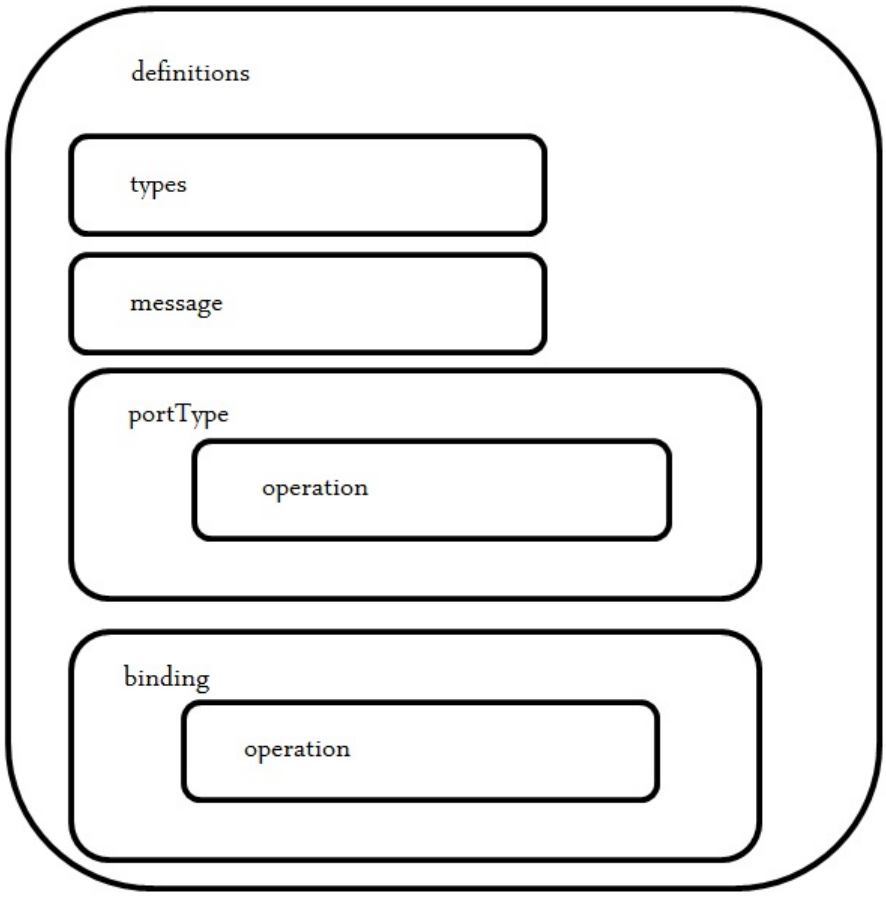

Figura 3.13: Estructura de un documento WSDL

La sección **"message"** concreta los elementos de datos en las operaciones de manera lógica. Puede tener más de una parte que son comparables a los parámetros de una llamada a un procedimiento en un lenguaje de programación tradicional.

La sección **"operation"** determina cada una de las operaciones de portType. Puede ser comparada a la operación "call" de Java. Por cada sección "operation" en el documento WSDL se tendrá definida una acción en el documento SOAP. Es posible declarar diversos tipos de operaciones: response/request, solicit/response, one-way, notification. La sección "operation" admite tres mensajes:

- De entrada: especifica el tipo de dato que el servicio web espera recibir.

- De salida: define el tipo de dato que el servicio web enviará.
- De error: mensaje que se enviará en caso de que se produzca un error.

El elemento más importante de este documento es la sección **"portType"**. Siguiendo con la comparación establecida con los lenguajes de programación tradicionales, estaríamos hablando de las bibliotecas. Es el conjunto de operaciones que puede ejecutar un servicio web además de los mensajes involucrados en éste. Gracias a este elemento el cliente puede obtener información sobre las operaciones del servicio. La estructura de este elemento es la siguiente:

```
<portType>
 <operation>...</operation>
  <...>...</...>
 <operation>...</operation>
</portType>
```

Se puede decir que cada una de las operaciones dentro de esta sección son funciones definidas. Establece el punto de conexión a un servicio web.

Vemos ahora la parte funcional del documento. La primera sección que se va a describir es **"binding"**. Se encarga de establecer el formato de los datos y el protocolo para cada "portType", obtiene la información que necesita el servicio web para conectarse de manera física con el cliente. Esta sección es la encargada de enlazar las dos partes del documento (funcional y no funcional). Tiene dos atributos: *name* que indica el nombre del binding y *type* que indica el puerto para establecer la conexión. Cuando se define una sección "portType", ésta puede aparecer en más de una sección "binding".

Por último una parte del documento WSDL:

```
<definitions
   xmlns:wsu="http://docs.oasis-open.org/wss/
    2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
   xmlns:wsp="http://www.w3.org/ns/ws-policy"
   xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://pack_sw/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   targetNamespace="http://pack_sw/" name="ont_ws">
 <types>
  <xsd:schema>
   <xsd:import namespace="http://pack_sw/"
    schemaLocation=
          "http://localhost:8080/WebServiceClas/ont_ws?xsd=1"/>
  </xsd:schema>
 </types>
 <message name="Clasificacion">
  <part name="parameters" element="tns:Clasificacion"/>
 </message>
 <message name="ClasificacionResponse">
  <part name="parameters" element="tns:ClasificacionResponse"/>
 </message>
 <portType name="ont_ws">
  <operation name="Clasificacion">
   <input wsam:Action="http://pack_sw/ont_ws/ClasificacionRequest"
```

```
      message="tns:Clasificacion"/>
    <output wsam:Action="http://pack_sw/ont_ws/ClasificacionResponse"
      message="tns:ClasificacionResponse"/>
   </operation>
 </portType>
 <binding name="ont_wsPortBinding" type="tns:ont_ws">
  <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
  <operation name="Clasificacion">
   <soap:operation soapAction=""/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </operation>
 </binding>
 <service name="ont_ws">
  <port name="ont_wsPort" binding="tns:ont_wsPortBinding">
  <soap:address
     location="http://localhost:8080/WebServiceClas/ont_ws"/>
  </port>
  </service>
</definitions>
```

Capítulo 4

Conclusiones y trabajo futuro

En el último capítulo de esta memoria se exponen las conclusiones extraídas de la investigación descrita y los resultados obtenidos. Además, se plantean líneas de investigación futuras que se podrán afrontar a partir de los resultados actuales.

Las ontologías son herramientas para la descripción semántica que facilitan la distribución de datos. Cuando se diseña una ontología se parte de la definición de términos a los que se les da significado semántico gracias a relaciones y axiomas. Todo ese conjunto garantiza un razonamiento que ofrecerá a los términos definidos significado semántico. El conocimiento generado será útil para integrarlo con cualquier tecnología, ofreciéndole un valor añadido que será un paso más en el desarrollo de la web semántica.

Tras realizar un análisis de las metodologías disponibles para desarrollar una ontología se llega a la conclusión de que son una tecnología de constante evolución que no tiene estandarización para su diseño metodológico, ninguno se considera mejor que otro, depende, en su mayor parte, del enfoque del diseñador y el uso que se le vaya a dar a la ontología. Aunque todas las metodologías estudiadas cuentan con etapas comunes, la mayoría de los grupos de trabajo establecen la suya propia para proceder, su propio lenguaje de implementación e incluso sus propias herramientas de desarrollo. Finalmente, si bien la metodología a seguir no está normalizada y puede ser establecida por el diseñador, los lenguajes de programación de

ontologías se han estandarizado dando lugar a los ya descritos RDF, OWL y DAML+OIL.

Cuando se realiza el diseño de una ontología, éste es totalmente subjetivo. No existen dos ontologías que modelen un mismo dominio y sean iguales si se han diseñado por personas diferentes, lo que no quiere decir que una esté mejor diseñada que la otra. Son puntos de vista diferentes y es por eso que son distintas.

El hecho de que una ontología se utilice para varias aplicaciones influye a la hora de realizar su diseño. No se puede comprobar nada que no se haya probado antes. Por ello aunque el diseño de la ontología sea correcto, a la hora de implementarse es posible que tenga que modificarse debido a el entorno de aplicación.

Las ontologías son una tecnología en constante desarrollo debido a su origen y finalidad. Es posible que esta misma ontología cuando esté implementada no se corresponda totalmente con la expresada en el diseño; lo que no significa que esté hecho de manera incorrecta, sino que a la hora de aplicar el conocimiento se ha establecido una mejor forma o el desarrollador se encuentra con barreras que en el diseño no estaban.

Los servicios web son herramientas de comunicación que ofrecen interfaces para la ejecución de servicios. Son independientes de la plataforma en la que se ejecuten, ayudan al usuario a realizar una tarea ya implementada y pueden integrarse con otras tecnologías de manera sencilla y clara para el usuario.

El estudio de esta teconología deja claro que supone una adaptación de las aplicaciones a cualquier ámbito de trabajo. Gracias a su estandarización es posible describirlos, localizarlos e invocarlos de manera sencilla por otros servicios a los que puedan dar uso.

El objetivo de la investigación era comprobar que la integración de ontologías con servicios web ofrecía el valor añadido esperado. El sistema planteado permite a los usuarios acceder al conocimiento expresado de manera sencilla y adaptable a las necesidades de cada uno.

Gracias al origen de reutilización de las ontologías que es compartido con el de los servicios web, cualquier usuario que necesite utilizar la ontología propuesta, en este caso, podrá hacerlo llevando a cabo una pequeña integración del servicio en la herramienta deseada.

En esta memoria se plantea el desarrollo de una ontología de dominio, la reutilización de una ontología de aplicación y la integración de éstas en un servicio web. Debido al desarrollo de la web semántica, después de realizar este estudio aún quedan varias líneas de investigación que se pueden abordar para completarlo (algunas de ellas están siendo desarrolladas actualmente):

- Finalización de la implementación y pruebas del sistema planteado, comprobando la correcta funcionalidad establecida.
- Ampliación de la ontología seleccionada, de manera que se amplíe el conocimiento que puede derivarse de ella, en este caso, la calidad del aceite de oliva puede depender también del método de recogida y del sistema de extracción.
- Hibridación del sistema propuesto con otras tecnologías demostrando que la integración de servicios web con ontologías no favorece únicamente a la web semántica, sino que es aplicable a cualquier herramienta en desarrollo.
- Integración de la ontología con un servicio web semántico. Los servicios web semánticos tratan esta tecnología desde el punto de vista de las ontologías, es posible que la integración con ese tipo de servicios web sea más sencilla.

Apéndice A

Glosario de términos de las ontologías

En este apéndice se presentan por orden alfabético todos los términos tanto de la ontología de dominio. Se comenta cuál es su utilidad, tipo (clase, instancia, relación, ...), ascendientes y descendientes en su caso, relaciones, cardinalidad en caso necesario, ...

Aceite: esta clase pertenece a la ontología de dominio. Subclase de "GrasaAlimentaria", describe el concepto de aceite, que podría ser aceite vegetal o animal, sin embargo para el propósito del trabajo solamente se despliega la clase "AceiteVegetal".

aceite_de_oliva: es una clase perteneciente a la ontología de dominio. Es una subclase de "AceiteDeOlivaNoVirgen", será una de las clases en la que se relizará la clasificación, siempre que cumpla los requisitos de la tabla 3.5. Se relaciona, por herencia con las clases "VariedadPrincipal", "PropiedadOrganoléptica" y "Componente" que son las clases necesarias para determinar si se trata de este tipo de aceite. No tiene subclases. Es disjunta de su clase hermana "aceite_de_oliva_refinado".

aceite_de_oliva_lampante: es una clase que pertenece a la ontología de dominio. Subclase de "AceiteDeOlivaVirgen" se trata de una de las clases clasificadoras del aceite, los aceites de oliva podrán ser clasificados en esta clase si cumplen los requisitos expuestos en la tabla 3.5. Hereda de su superclase las relaciones con las clases "VariedadPrincipal", "PropiedadOr-

ganoléptica" y "Componente". No tiene subclases y es disjunta de "aceite_de_oliva_virgen" y "aceite_de_oliva_virgen_extra", sus clases hermanas.

aceite_de_oliva_refinado: es una clase perteneciente a la ontología de dominio. Se trata de una subclase de "AceiteDeOlivaNoVirgen", es una de las clases en las que se clasifica el aceite. Los aceites de oliva se podrán clasificar en esta clase si cumplen los requisitos representados en la tabla de la imagen 3.5. Hereda de su clase padre las relaciones con "VariedadPrincipal", "PropiedadOrganoléptica" y "Componente" que ayudarán a determinar el tipo de aceite del que se trata. No tiene subclases. Es disjunta su clase hermana "aceite_de_oliva".

aceite_de_oliva_virgen: es una clase, pertenece a la ontología de dominio. Es una subclase de "AceiteDeOlivaVirgen" y será clasificadora para los aceites de oliva. Un aceite será aceite de oliva virgen si cumple los requisitos establecidos por las agencias reguladoras que se representan en la imagen 3.5. Hereda de su clase padre las relaciones con "VariedadPrincipal", "PropiedadOrganoléptica" y "Componente". No tiene subclases y es disjunta de sus clases hermanas: "aceite_de_oliva_virgen_extra" y "aceite_de_oliva_lampante".

aceite_de_oliva_virgen_extra: pertenece a la ontología de dominio. Es subclase de "AceiteDeOlivaVirgen". Es de una de las clases en las que se clasifica un tipo de aceite. El aceite clasificado de este tipo debe cumplir las características de componentes y propiedades organolépticas de la imagen 3.5. Hereda las relaciones con las clases "VariedadPrincipal", "PropiedadOrganoléptica" y "Componente" de su clase padre. No tiene subclases y es disjunta de "aceite_de_oliva_virgen" y "aceite_de_oliva_lampante".

aceite_de_orujo_de_oliva: perteneciente a la ontología de dominio es una sublcase de "AceiteDeOrujoDeOliva". Clasifica la categoría comercial de los aceites de oliva, un aceite será de orujo de oliva si tiene un determinado nivel de acidez y cumple las propiedades organolépticas represenatdas en la imagen 3.5. Hereda las relaciones de su clase padre: "VariedadPrincipal", "PropiedadOrganoléptica" y "Componente" No contiene subclases y es disjunta

de sus clases hermanas “aceite_de_orujo_de_oliva_bruto” y “aceite_de_orujo_de_oliva_refinado”.

aceite_de_orujo_de_oliva_bruto: pertenece a la ontología de dominio. Subclase de “AceiteDeOrujoDeOliva” es una de las clases en las que se pueden clasificar los aceites de oliva, en la tabla 3.5 se representan los requisitos que ha de cumplir un aceite para ser catalogado en esta clase. Hereda las relaciones con las clases “VariedadPrincipal”, “PropiedadOrganoléptica” y “Componente” de su clase padre. No tiene subclases. Es disjunta de sus clases hermanas “aceite_de_orujo_de_oliva” y “aceite_de_orujo_de_oliva_refinado”.

aceite_de_orujo_de_oliva_refinado: pertenece a la ontología de dominio. Subclase de “AceiteDeOrujoDeOliva”. Ordena a los aceites de oliva en esta categoría comercial, un aceite será clasificado en esta clase si cumple las características de la tabla 3.5. Relacionada por herencia con “VariedadPrincipal”, “PropiedadOrganoléptica” y “Componente” por su clase padre. No tiene subclases y es disjunta de sus clases hermanas “aceite_de_orujo_de_bruto” y “aceite_de_orujo_de_oliva_refinado”.

AceiteDeOliva: es una clase de la ontología de dominio. Subclase de “GrasaVegetal” que tiene como subclases “VariedadPrincipal” y “CategoríaComercial” y pretende definir la tipología en la que se pueden catalogar los aceites de oliva. Se relaciona con la “Atributo” por la relación esta_formado_por que tienen cardinalidad 1:1

AceiteDeOlivaNoVirgen: padre de las clases “aceite_de_oliva” y “aceite_de_oliva_refinado” y disjunta de sus clases hermanas “AceiteDeOlivaVirgen” y “AceiteDeOrujoDeOliva”, es una subclase de “CategoríaComercial” declarada en la ontología de dominio que define a los aceites de oliva no vírgenes. Hereda las relaciones de su clase padre: “es_obtenida_de” una o más “VariedadPrincipal”, “se_evalua_por” uno o varios “Atributo” y ´´está_formado_por” un “Componente”, la cardinalidad de estas relaciones es 1:N, 1:N y 1:1 respectivamente.

AceiteDeOlivaVirgen: definida en la ontología de dominio representa a los aceites de oliva vírgenes. Se trata de una subclase de “CategoríaComercial” que contiene tres subcla-

ses “aceite_de_oliva_virgen”, “aceite_de_oliva_virgen_extra” y “aceite_de_oliva_lampante”. Es disjunta de sus clases hermanas “AceiteDeOlivaNoVirgen” y “AceiteDeOrujoDeOliva”. Hereda, al igual que sus hermanas, las relaciones de su clase padre: “es_obtenida_de” una o varias “VariedadPrincipal”, “se_evalua_por” uno o varios “Atributo” y ´´está_formado_por” un “Componente”, la cardinalidad de estas relaciones es 1:N, 1:N y 1:1 respectivamente.

AceiteDeOrujoDeOliva: es una subclase de “CategoríaComercial” definida en la ontología de dominio que representa a los aceites de orujo de oliva. Dentro de ella se declaran tres subclases “aceite_de_orujo_de_oliva”, “aceite_de_orujo_de_oliva_bruto” y “aceite_de_orujo_de_oliva_refinado”. Igual que sus clases hermanas, “AceiteDeOlivaVirgen” y “AceiteDeOlivaNoVirgen”, de la que es disjunta, hereda las relaciones de su clase padre: “es_obtenida_de” una o varias “VariedadPrincipal”, “se_evalua_por” uno o varios “Atributo” y ´´está_formado_por” un “Componente”, la cardinalidad de estas relaciones es 1:N, 1:N y 1:1 respectivamente.

AceiteVegetal: esta clase tiene dos padres: “Aceite” y “GrasaVegetal”, representa los aceites extraídos de un origen vegetal, como los aceites de oliva, por lo que es disjunta de “GrasaAnimal”. Es la clase padre de “AceiteDeOliva”. Pertenece a la ontología de dominio.

acidez: es un atributo de la ontología de dominio. Es fundamental para la clasificación de los aceites, ya que según su valor será un tipo de aceite u otro. Es una instancia de la clase “ÁcidoGrasoLibre”.

ÁcidoGraso: es la única subclase definida dentro de “ComponenteMayoritario”, no es el único componente mayoritario que tienen los aceites de oliva, pero si el necesario para la aplicación. Tiene una subclase “ÁcidoGrasoLibre” que aunque tampoco es la única subclase definida en la ontología, si es la que se utiliza para realizar la clasificación del aceite en este caso.

ÁcidoGrasoLibre: subclase definida dentro de “ÁcidoGraso”, contiene un individuo: “acidez” que será necesario para la clasificación de un aceite. Se relaciona con “CategoríaCo-

mercial" a través de "conforma", que tiene cardinalidad 1:1. Aunque en el mundo real modelado no es el único tipo de ácido graso que pueden contener los aceites de oliva, si es el único que se ha incluido en este glosario por considerarse determinante para su clasificación, se deduce, por tanto, que tiene clases hermanas de las que es disjunta.

almendra_verde: pertenece a la ontología de dominio, se trata de una instancia de la clase "FlavorPositivo", define un flavor de los aceites de oliva. Deberá tenerse en cuenta para realizar una correcta clasificación de los aceites de oliva tanto en su categoría comercial como en el tipo de variedad.

almendrado: individuo de la clase "FlavorPositivo" en la ontología de dominio, define un flavor de los aceites de oliva. Se tendrá en cuenta para clasificar los aceites de oliva por categoría comercial y tipo de variedad.

alpechín: instancia perteneciente a la clase "FlavorNegativo" de la ontología de dominio, define un flavor negativo de los aceites de oliva. Deberá de ser tenida en cuenta a la hora de realizar la clasificación para las categorías comerciales, ya que no ayuda a la clasificación por variedad.

arbequina: variedad de la que se extraen los aceites de oliva, es una subclase de "VariedadPrincipal" en la ontología de dominio. Se clasificará en esta clase si cumple ciertos requisitos organolépticos.

Aroma: subclase definida en "Atributo" que representa un atributo de los aceites de oliva. Este atributo es necesario para la clasificación en esta aplicación, no obstante, al representar un atributo importante en el dominio se ha estimado oportuno representarlo en la ontología de dominio y explicarlo en ester glosario. Es disjunta de sus clases hermanas: "Flavor", "Fragancia", "Olor", "Sabor", "Textura".

Atributo: subclase de "PropiedadOrganoléptica" definida en la ontología de dominio

que representa los diferentes atributos que pueden observarse en los aceites de oliva. Un atributo ´´evalua" a uno o varios "AceiteDeOliva", ya que los atributos serán considerados para la clasificación tanto por categoría comercial como por variedad. Las subclases "Aroma", "Flavor", "Fragancia", "Olor", "Sabor" y "Textura" son los atributos que se consideran para catalogar a un aceite.

atrojado: individuo definido en la clase "FlavorNegativo" en la ontología de dominio. Es un flavor negativo de los aceites de oliva y se contará para llevar a cabo la clasificación para las categorías comerciales, no será necesario para clasificarlo por variedad.

avinado: es una instancia de la clase "FlavorNegativo" de la ontología de dominio, define un flavor negativo de los aceites de oliva. Se considerará cuando se haga la clasificación por categoría comercial de un aceite, ya que los flavores negativos no aportan información para la clasificación por variedad.

basto: se trata de un individuo definido en la clase "TexturaNegativa" en la ontología de dominio. Debido a su carácter negativo no se considera para determinar la variedad de la que se ha extraído el aceite, no obstante si es útil cuando se realiza la clasificación por categoría comercial.

CategoríaComercial: subclase definida en la ontología de dominio, hija de "Aceite-DeOliva". Es una de las clases más importantes de la ontología, ya que se encarga de la clasificación de los aceites de oliva según su nivel de acidez y sus atributos organolépticos. Se relaciona con "VariedadPrincipal" y "ÁcidoGrasoLibre" a partir de las relaciones denominadas "es_obtenida_de" con cardinalidad 1:N y "esta_formado_por" con cardinalidad 1:1 respectivamente; además, hereda de su padre la relación "se_evalua_por" que lo une con "Atributo". Es padre de las clases "AceiteDeOlivaVirgen", "AceiteDeOlivaNoVirgen" y "AceiteDeOrujoDeOliva".

cocido-quemado: es una instancia de la ontología de dominio, que pertenece a la clase

"FlavorNegativo", define un flavor negativo de los aceites de oliva. No se considera, debido a su carácter negativo, para la clasificación por variedad, pero si para se tendrá en cuenta cuando se clasifica por categoría comercial.

Componente: se trata de una clase definida en la ontología de dominio que representa los componentes químicos de los aceites de oliva. En el dominio modelado se distinguen "ComponenteMayoritario" y "ComponenteMinoritario". Es disjunta de las clases "Grasa" y "PropiedadOrganoléptica"

ComponenteMayoritario: clase de la ontología de dominio que representa los componentes mayoritarios de los aceites de oliva. Definida como subclase de "Componente" y disjunta de su hermana "ComponenteMinoritario". Es necesaria su representación en la ontología debido a la importancia de los atributos de sus subclases.

ComponenteMinoritario: clase definida en la ontología de dominio como hija de "Componente" y hermana disjunta de "ComponenteMayoritario". No contiene ningún atributo representativo para la aplicación, pero se ha estimado oportuno representarlar en la ontología debido a que es una jerarquía presente en el mundo real modelado.

cornicabra: es una subclase de "VariedadPrincipal" en la ontología de dominio que define la variedad de la que se pueden extraer los aceites de oliva. Un aceite será de esta clase si contiene ciertos atributos organolépticos. No es disjunta de sus clases hermanas porque pueden extraerse de diversas variedades.

dulce: definido en la clase "SaborElemental" de la ontología de dominio, expresa un sabor característico en los aceites de oliva, es considerado tanto para su clasificación como categoría comercial como para definir la variedad de la que se trata.

empeltre: subclase de "VariedadPrincipal" en la ontología de dominio; declara la variedad de la que se ha extraído el aceite y cumple unas determinadas características orga-

nolépticas. Debido a que los aceites de oliva pueden ser extraídos de varias variedades no es disjunta de sus clases hermanas.

esparto: instancia de "FlavorNegativo", clase perteneciente a la ontología de dominio. Se estimará para la clasificación en categoría comercial ya que los atributos negativos no se tienen en cuenta para clasificar un aceite por variedad.

farga: subclase de "VariedadPrincipal" que define una variedad de aceite. Está definida en la ontología de dominio; un aceite será instancia de esta clase si tiene unas determinadas propiedades organolépticas. No es disjunta de sus clases hermanas porque los aceites de oliva pueden extraerse de varias variedades.

Flavor: define a una subclase de "Atributo" en la ontología de dominio, que representa los posibles flavores que tienen los aceites de oliva. Tiene dos subclases encargadas de categorizarlos: "FlavorPositivo" y "FlavorNegativo". Es disjunta de sus clases hermanas: "Aroma", "Fragancia", "Olor", "Sabor" y "Textura". Hereda de su clase padre la relación "evalua"; un flavor "evalúa" a un "AceiteDeOliva", esta relación tiene cardinalidad 1:1.

FlavorNegativo: es una subclase de "Flavor" que representa solamente los flavores negativos que pueden tener los aceites de oliva. Está definida en la ontología de dominio y tiene como atributos a: ´´alpechín", ´´atrojado", ´´avinado", ´´cocido-quemado", ´´esparto", ´´gusano", ´´heno-madera", ´´lubricante", ´´metálico", ´´moho-humedad", ´´pepino", ´´rancio", ´´salmuera" y ´´tierra". Hereda de su padre la relación que lo une con "AceiteDeOliva". Es disjunta de "FlavorPositivo" su clase hermana.

FlavorPositivo: subclase de "Flavor" que representa los flavores positivos de los aceites de oliva. Definida en la ontología de dominio, tiene los siguientes atributos: ´´almendra_verde", ´´almendrado", ´´frutado", ´´frutado_maduro", ´´frutado_verde", ´´hierba", ´´hojas_verdes" y ´´manzana". Hereda la relación "evalua" de su padre, que lo une con la clase "AceiteDeOliva". Es disjunta de su clase hermana "FlavorNegativo".

Fragancia: es una subclase definida en "Atributo". Representa un atributo de los aceites de oliva que no se tiene en cuenta para la clasificación en esta aplicación, no obstante, al representar un atributo importante en el dominio se ha estimado oportuno representarlo en la ontología de dominio e incluir su descripción en este glosario. Es disjunta de sus clases hermanas: "Aroma", "Flavor", "Olor", "Sabor" y "Textura".

frutado: es una instancia de "FlavorPositivo", clase definida en la ontología de dominio; es un flavor positivo que se tiene en cuenta a la hora de clasificar los aceites de oliva para conocer su categoría comercial y tipo de variedad.

frutado_maduro: individuo de la clase "FlavorPositivo" de la ontología de dominio define un flavor en los aceites de oliva. Se considera cuando se realiza la clasificación tanto por variedad como por categoría comercial.

frutado_verde: definido en la ontología de dominio como instancia de "FlavorPositivo", especifica un flavor característico de los aceites de oliva. Se estima cuando se clasifica por categoría comercial y tipo de variedad.

Grasa: se trata de una sublcase de la ontología de dominio que representa el origen del aceite. Es padre de "GrasaAlimentaria" y disjunta de las clases hermanas definidas: "Componente" y "PropiedadOrganoléptica".

GrasaAlimentaria: es la clase hija de "Grasa", representa la grasa considerada comestible. Declara como hijas las clases 'GrasaAnimal", "GrasaVegetal" y "Aceite". Es definida en la ontología de dominio por ser importante en la jerarquía del mundo real modelado.

GrasaAnimal: subclase de "GrasaAlimentaria", representa la grasa de origen animal. En este trabajo no se ha tenido en cuenta, pero se ha estimado pertinente incluirla en la ontología por su origen contrario a la grasa vegetal que es de la que surgen los aceites de oliva. Es disjunta de "GrasaVegetal" y de la subclase de "Aceite" denominada "AceiteVegetal".

GrasaVegetal: clase padre de "AceiteVegetal", igual que "Aceite", es subclase de "GrasaAlimentaria" en la ontología de dominio. Representa las grasas que tienen un origen vegetal. Es disjunta de su clase hermana "GrasaAnimal".

gusano: instancia de la clase "FlavorNegativo" de la ontología de dominio, es un flavor negativo en los aceites de oliva. Deberá considerarse cuando se desea conocer la categoría comercial de un aceite si contiene este atributo, no es útil para clasificar la variedad de la que se trata.

heno-madera: definido en "FlavorNegativo", clase de la ontología de dominio, especifica un posible flavor negativo de los aceites de oliva. Será considerado cuando se quiera clasificar dicho aceite en una categoría comercial, pero no será útil para clasificarlo según su variedad.

hierba: individuo declarado en la clase "FlavorPositivo" de la ontología de dominio, especifica un flavor característico en los aceites de oliva y se estimará cuando se realice la clasificación tanto por categoría comercial como por tipo de variedad.

hojas_verdes: instancia declarada en la clase "FlavorPositivo", perteneciente a la ontología de dominio, que definirá un flavor de los aceites de oliva. Será tenido en cuenta cuando se clasifique por categoría comercial y tipo de variedad.

hojiblanca: representa una de las variedades principales de las que se pueden extraer aceites de oliva. Se declara como una subclase de "VariedadPrincipal", en la ontología de dominio. Un aceite será catalogado perteneciente a esta clase si tiene ciertas propiedades organolépticas. No es disjunta de sus clases hermanas ya que pueden extraerse de varias variedades.

lechín_de_Granada: simboliza una variedad de la que se pueden extraer aceites de oliva, declarado en la clase "VariedadPrincipal" de la ontología de dominio, clasificará a los aceites de oliva si contienen determinadas propiedades organolépticas; debido a que se extraen, en

ocasiones, de múltiples variedades no es disjunta del resto de sus clases hermanas.

lechín_de_Sevilla: es una subclase de "VariedadPrincipal" en la ontología de dominio. Representa una variedad de las que se extaen los aceites de oliva. Se clasifican en esta clase si contienen ciertas propiedades organolépticas. No se ha definido disjunta de sus clases hermanas debido a que los aceites de oliva pueden ser extraídos mezclando variedades.

lubricante: individuo definido en la ontología de dominio en la clase "FlavorNegativo". Simboliza un flavor negativo en un aceite de oliav que será considerado en la clasificación por categoría comercial pero no por variedad.

manzana: instancia definida en la clase "FlavorPositivo", en la ontología de dominio, define un flavor positivo característico en los aceites de oliva que permitirá su clasificación por variedad y ayudará en la clasificación por categoría comercial.

manzanilla_cacereña: representación de una variedad de los aceites de oliva, definida en la ontología de dominio como subclase de "VariedadPrincipal". Un aceite será catalogado en esta clase satisface ciertas condiciones organolépticas. No se ha definido como disjunta de sus clases hermanas debido a que puede extraerse de más de una variedad.

manzanilla_de_Sevilla: subclase de "VariedadPrincipal" definida en la ontología de dominio que representa una variedad de la que se pueden extraer aceites de oliva. Se clasificará en ella si contiene unas características organolépticas determinadas. Ya que los aceites de oliva pueden ser extraídos de diversas variedades no se ha definido como disjunta de sus clases hermanas.

metálico: instancia definida en la clase "FlavorNegativo" de la ontología de dominio, representa un flavor en los aceites de oliva. Será considerada a la hora de realizar la clasificación de un aceite en categoría comercial, ya que no ayuda a clasificar por variedad.

moho-humedad: representa un "FlavorNegativo" de los aceites de oliva y está definido

en la ontología de dominio. Se utiliza cuando se realiza la clasificación por categoría comercial, ya que no es determinante para la clasificación por variedad.

morisca: Declarada en la ontología de dominio, es una subclase de "VariedadPrincipal" que representa una de las variedades principales de las que se pueden extraer los aceites de oliva, que estarán catalogados en esta clase si se observan ciertas propiedades organolépticas en su cata. Los aceites de oliva se pueden extraer de diversas variedades, por ello no se ha definido como disjunta de sus clases hermanas.

morrut: variedad de aceites de oliva, definida como una subclase de "VariedadPrincipal" en la ontología de dominio. No es disjunta de sus clases hermanas por el carácter de los aceites de oliva (pueden ser extraído de más de una variedad). Un aceite se cataloga en esta clase si en su cata se advierten determinadas propiedades organolépticas.

Olor: atributo de los aceites de oliva representado como subclase de "Atributo" que, aunque no se utilizar para la clasificación en esta aplicación, se ha considerado oportuno explicarlo en el presente glosario e incluirlo en la ontología por estar presente en la jerarquía de conceptos del dominio representado. Es disjunta de sus clases hermanas: "Aroma", "Flavor", "Fragancia", "Sabor" y "Textura".

pepino: atributo negativo de los aceites de oliva representado como una instancia de la clase "FlavorNegativo" definida en la ontología de dominio. Se considera a la hora de clasificar un aceite por categoría comercial y no por variedad ya que no es un dato representativo para dicha clasificación.

picante: individuo declarado en "Sabor", clase de la ontología de dominio. Representa un sabor característico en los aceites de oliva y se tiene en cuenta cuando se evalúa un aceite para clasificar su categoría comercial como la variedad de la que se extrae.

picual: es una subclase definida en la ontología de dominio, tiene como padre a "Va-

riedadPrincipal" y representa una variedad de la que se puede extraer aceites de oliva. Un aceite será de esta variedad si tiene unas determinadas propiedades organolépticas. Debido a que los aceites de oliva pueden extraerse de diversas variedades no es disjunta de sus clases hermanas.

picudo: subclase de "VariedadPrincipal" definida en la ontología de dominio que representa una variedad en la que se podrán clasificar los aceites de oliva si cumple ciertos requisitos organolépticos. No es disjunta de sus clases hermanas debido a que un aceite puede ser extraído de diversas variedades.

PropiedadOrganoléptica: clase definida en la ontología de dominio que representa las diferentes propiedades organolépticas que pueden obtenerse en la cata de los aceites de oliva. Contiene ciertos atributos que no determinantes en este trabajo, pero importantes cuando se habla de aceite. Contiene una clase "Atributo" y es disjunta de "Componente" y "Grasa".

rancio: declarado en "FlavorNegativo", clase perteneciente a la ontología de dominio, representa un flavor negativo no deseable en los aceites de oliva, es considerado cuando se realizar una clasificación para establecer la categoría comercial, ya que no es un dato representativo cuando se realiza la clasificación por variedad.

RegistrarDHT: se encarga de actualizar el Distributed Hash Table (DHT) con los T-Box y A-Box de otros nodos. Está definida como subclase de "Accion" y por tanto hereda sus relaciones "recompensa", "castigo". Además, tiene una relación denominada "DHT_Asociado" que la une con "DHT" y se encarga de actualizar el DHT correcto.

Sabor: subclase de "Atributo", definida en la ontología de dominio, que representa los sabores de los aceites de oliva. Tiene una subclase: "SaborElemental". Contiene un sólo atributo en este caso: "picante". Es disjunta de sus clases hermanas: "Aroma", "Flavor", "Fragancia", "Olor" y "Textura". Hereda de su clase padre la relación "evalua"; un sabor "evalúa" a un "AceiteDeOliva", tiene cardinalidad 1:1.

SaborElemental: es la única subclase de "Sabor". Está definida en la ontología de dominio y representa, en este caso, el sabor elemental que puede tener un aceite, por lo que tiene un atributo "dulce". Hereda la relación "evalua" de su clase padre.

salmuera: individuo de "FlavorNegativo" en la ontología de dominio, establece un flavor negativo en los aceites de oliva. Es tenido en cuenta cuando se desea establecer la categoría comercial de un aceite de oliva, pero no es útil cuando se desea conocer la variedad de la que se ha extraído el aceite.

Textura: es una subclase de "Atributo", definida en la ontología de dominio, que representa las texturas que pueden tener los aceites de oliva. Tiene dos subclases que categorizan las posibles texturas: "TexturaPositiva" y "TexturaNegativa". Es disjunta de sus clases hermanas: "Aroma", "Flavor", "Fragancia", "Olor" y "Sabor". Hereda de su clase padre la relación "evalua"; una textura "evalúa" a un "AceiteDeOliva", esta relación tiene cardinalidad 1:1.

TexturaNegativa: subclase de "Textura", representa las texturas negativas que pueden tener los aceites de oliva. Está definida en la ontología de dominio y tiene, en este caso, un sólo atributo ´´basto". Hereda la relación "evalua" de su padre, que lo une con la clase "AceiteDeOliva". Es disjunta de su clase hermana "TexturaPositiva".

TexturaPositiva: es una subclase definida en la ontología de dominio hija de la clase "Textura". Representa las texturas de los aceites de oliva. Tiene un sólo atributo ´´untuoso". Hereda la relación "evalua" de su padre. Es disjunta de su clase hermana "TexturaNegativa".

tierra: declarado en la clase "FlavorNegativo" de la ontología de dominio, representa un flavor negativo de los aceites de oliva que se considera cuando se desea conocer la categoría comercial del aceite. No es útil cuando se realiza una clasificación por variedad.

untuoso: individuo definido en la ontología de dominio, en la clase "TexturaPositiva",

representa una textura deseable en los aceites de oliva, será considerada para obtener la categoría comercial de dichos aceites o para conocer la variedad de la que se extraen.

VariedadPrincipal: es una subclase importante dentro del ámbito que se trata en el trabajo, pues será la que clasifique el aceite según la variedad de la que se ha extraído gracias a sus atributos organolépticos, por ello hereda de su clase padre "AceiteDeOliva" la relación con "Atributo" denominada "se_evalua_por" que tiene cardinalidad 1:N. Además se relaciona con "CategoríaComercial" a través de "da_lugar_a" (cardinalidad 1:N). Tiene como subclases a cada una de las variedades principales definidas en el proyecto Olivaterm.

verdial_de_Badajoz: se trata de una subclase de "VariedadPrincipal". Está declarada en la ontología de dominio y se utiliza para clasificar a los aceites de oliva según la variedad de la que se han extraído, siempre que contengan determinadas propiedades organolépticas. No se ha definido como disjunta de sus clases hermanas porque un aceite puede estar hecho con diversas variedades.

verdial_de_Huévar: representa una variedad principal de la que se extraen aceites de oliva. Definida en la ontología de dominio como subclase de "VariedadPrincipal". Un aceite estará catalogado en esta clase si al catarlo contiene ciertas propiedades organolépticas. Declarada como disjunta de sus clases hermanas, ya que los aceites de oliva se pueden extraer de diversas variedades.

Cabe destacar que estas son las clases, subclases y atributos utilizados en la aplicación pero en la ontología de dominio han sido definidas otras clases presentes en la jerarquía real del dominio de los aceites de oliva y que no han sido incluidas en este apéndice por carecer de uso en la aplicación.

Bibliografía

[1] Darpa agent markup language. http://www.daml.org. Accesed October 2012.

[2] extensible markup language. http://www.w3.org/XML/. October 2012.

[3] Hypertext markup language. http://www.w3.org/MarkUp/html-spec/html-spec_1.html#SEC1. Accesed October 2012.

[4] Knowledge interchange format. http://logic.stanford.edu/kif/dpans.html. Accesed October 2012.

[5] World wide web consortium. http://www.w3c.es/. Accesed October 2012.

[6] Resource description framework schema. http://www.w3.org/TR/rdf-schema/, 2004. Accesed October 2012.

[7] The friend of a friend (foaf) project. http://xmlns.com/foaf/spec/, 2010. Accesed October 2012.

[8] Fipa. http://www.fipa.org/, 2012. [Accesed October 2012].

[9] K. Aberer et al. Gridvine: building internet-scale semantic overlay networks. Proceedings of the Third International Semantic Web Conference, 2004.

[10] J. Arpirez et al. (onto)2agent: An ontology-based www broker to select ontologies. Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, (ECAI-98), 1998.

[11] J. C. Arpírez et al. WebODE: a workbench for ontological engineering. First International Conference on Knowledge Capture., 2001.

[12] A. Bernaras, I. Laresgoiti, and J. Corera. Building and Reusing Ontologies for Electrical Network Applications. Proceedings of the European Conference on Artificial Intelligence., 1996.

[13] A. Bernaras, I. Laresgoiti, and J. Corera. Building and reusing ontologies for electrical network applications. Proceedings of the European Conference on Artificial Intelligence., 1996.

[14] T. Berners-Lee. *Weaving the web.* 2000.

[15] T. Berners-Lee, J. Hendler, and O. Lassila. *The Semantic Web.* 2001.

[16] M. Blázquez et al. Building ontologies at the knowledge level using the ontology design environment. *Knowledge Acquisition of Knowledge-Based Systems*, 1998.

[17] F. Bobillo, M. Delgado, and J. Gmez-Romero. Representation of context-dependant knowledge in ontologies: a model and an application. *Elsevier*, 2008.

[18] S. Borgo, N. Guarino, and C. Masolo. Stratified ontologies: the case of physical objects. ECAI-96 Workshop on Ontological Engineering, 1996.

[19] P. Borst et al. Engineering ontologies. *International Journal of Human Computer Studies*, 1997.

[20] M. T. Cabré. *La terminología. Teoría, metodología, aplicaciones.* 1993.

[21] O. Corcho, A. Gómez-Pérez, and M. Fernández-López. Methodologies, tools and languages for building ontologies. Where is there meeting point? *Data and Knowledge Engineering*, 2002.

[22] F. Curbera et al. Unraveling the Web Services Web An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 2002.

[23] L. Damas and C. Tricot. L'ontoterminologie pour la recherche d'information sémantique. Actes de la quatrième conférence TOTh, 2010.

[24] A. Díaz Negrillo and J. Fernández Domínguez. *Bases para la terminología multilingüe del aceite de oliva*, pages 37–50. 2010.

[25] B. Decker et al. Selforganized Reuse of Software Engineering Knowledge supported by Semantic Wikis. Proc. of Workshop on Semantic Web Enabled Software Engineering (SWESE), 2005.

[26] B. Diemert, M. H. Abel, and C. Moulin. Modélisation des dénominations ontologiques. Actes de la quatrième conférence TOTh, 2010.

[27] H. Felber. *Terminology manual*. 1984.

[28] D. Fensel. Ontology-Based Knowledge Management. *IEEE*, 2002.

[29] D. Fensel, M. Kerrigan, and M. Zaremba. *Implementing Semantic Web Services. The SESA Framework*. 2008.

[30] M. Fernández-López et al. Building a chemical ontology using methontology and the ontology design environment. Technical report, 1999.

[31] M. Fernández-López and A. Gómez-Pérez. Overview and analysis of methodologies for building ontologies. The Knowledge Engineering Review, 2002.

[32] A. García Jiménez. Instrumentos de representación del conocimiento: Tesauros versus Ontologías. 2004.

[33] M. R. Genesereth and Nilsson L. Logical Foundation of Artificial Intelligence. *AI. Contains definition of conceptualisation in extensionalist terms*, 1987.

[34] A. Gómez-Pérez, M. Fernández, and N. Juristo. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. 1996.

[35] A. Gómez-Pérez, M. Fernández-López, and Ó. Corcho. Ontological Engineering. 2004.

[36] A. Gómez-Pérez, N. Juristo, and J. Pazos. Evaluiation and assessment of knowledge sharing technology. 1995.

[37] Ó. González. *XML (edición revisada y ampliada 2005)*. 2005.

[38] T. Gradner. An Introduction to Web Services. *Ariadne*, 2001.

[39] S. Grimm et al. *Handbook of Semantic Web Technologies*, pages 509–579. 2011.

[40] M. Grüninger and M. S. Fox. Methodology for the Design and Evaluation of Ontologies. 1995.

[41] T. Gruber. *A translation approach to portable ontology specifications*. 1993.

[42] T. R. Gruber. Toward principles for the design of ontologies used for knowledege sharing. Technical report, 1993.

[43] N. Guarino. Formal Ontologies and Information Systems. 1998.

[44] N. Guarino et al. *What Is an Ontology? Handbook on Ontologies*. 2009.

[45] P. Haase et al. Bibster a semantics-based bibliographic peer-to-peer system. Proceedings of the Third International Semantic Web Conference, 2004.

[46] F. Hakimpour et al. Semantic web service composition in irs-iii: the structured approach. Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, 2005.

[47] R. M. Heery. *What Is...RDF?* 1998.

[48] G. Hench and D. Fensel. *Implementing Semantic Web Services*, pages 5–7. 1998.

[49] J. Hendler et al. Ontology-based web agents. 1997.

[50] I. Horrocks. Daml+oil: a description logic for the semantic web. Technical report, 2002.

[51] C. Kaiser. *Down the gopher hole.* 2007.

[52] P. D. Karp, Chaudhri V. K., and J. Thomere. XOL: An XML-based ontology exchange language. 1999.

[53] J. Kim, M. Spraragen, and Y. Gil. An intelligent assistant for interactive workflow composition. Proceedings of the Ninth Internatinoal Conference on Intelligent User Interfaces, 2004.

[54] O. Lassila and R. Swick. Resource description framework (RDF). *W3C Recommendation.*, 1999.

[55] F. Lécué, A. Delteil, and A. Léger. Applying abduction in semantic web service composition. Proceedings of the IEEE International Conference on Web Services, 2007.

[56] F. Lécué, A. Delteil, and A. Léger. Dl reasoning and ai planning for web service composition. Proceedings of the IEEE/WIC/ACM International Conference onWeb Intelligence and Intelligent Agent Technology, 2008.

[57] F. Lécué and A. Léger. A formal model for semantic web service composition. Proceedings of the Fifth International Semantic Web Conference, 2007.

[58] Y. Lei, E. Motta, and J. Domingue. Ontoweavers: supporting the design of knowledge portals. Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management, 2004.

[59] D. B. Lenat et al. Cyc: Toward programs with common sense. *Communications of the ACM*, 1990.

[60] D. B. Lenat and R. V. Guha. Building Large knowledge-Based systems. *Addison-Wesley*, 1990.

[61] D. B. Lenat and R. V. Guha. The evolutioin of CycL, The Cyc Representation Language. *SIGART Bulletin*, 1991.

[62] J. Lin, M. S. Fox, and T. Bilgic. A requirement ontology for engineering design. Technical report, 1996.

[63] P. Lord. Components of an ontology. ontogenesis. `http://ontogenesis.knowledgeblog.org/514`, 2010. [Accesed October 2012].

[64] R. M. MacGregor. Inside the LOOM Description Clasifier. *ACM*, 1991.

[65] A. Maedche et al. *Spinning the Semantic Web*, pages 317–359. 2003.

[66] E. Mäkelä et al. Ontoviews a tool for creating semantic web portals. Proceedings of the Third International Semantic Web Conference, 2004.

[67] V. Mayank, N. Kositsyna, and M. Austin. Requirements Engineering and the Semantic Web, Part II. Representation, Management, and Validation of Requirements and System-Level Architectures. Technical report, 2004.

[68] R. McEntire et al. An Evaluation of Ontology Exchange Languages for Bioinformatics. 1999.

[69] D. L. McGuinness et al. DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems*, 2002.

[70] E. Miller. An introduction to the resoruce description framework. *D-Lib Magazine*, 1998.

[71] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda. Task ontology for reusable problem solving knowledge, Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. *IOS Press*, 1999.

[72] A. Moreno Ortiz. *Investigar en terminología*, pages 25–70. 2002.

[73] P. Muschamp. An introduction to Web Services. *BT Technology Journal*, 2004.

[74] Y. Nakamura-Delloye and R. Stern. Extraction de relations et de patrons de relations entre entités nommées en vue de l'enrichissement d'une ontologie. Actes de la cinquième conférence TOTh, 2011.

[75] F. Nef. L'ontologie ou miroir de la Terminologie. Actes de la quatrième conférence TOTh, 2010.

[76] N. Noy and D. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford University*, 2005.

[77] N. F. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Standford Universtity*, 2001.

[78] V.L. Payne and D.P. Metzler. Representation of context-dependant knowledge in ontologies: a model and an application. Proceedings of the 18th International Symposium on Computer-Based Medical Systems, 2005.

[79] S. Potts and M. Kopack. *Sams Teach Yourself Web Services in 24 Hours*. Sams Teach Yourself. Sams, 2003.

[80] J. Rao, P. Küngas, and M. Matskin. Composition of semantic web services using linear logic theorem proving. Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, 2006.

[81] M. Roldán Vendrell. *Bases para la terminología multilingüe del aceite de oliva*, pages 1–15. 2010.

[82] M. Roldán Vendrell. Lingüística y gestión terminológica. En prensa.

[83] G. Rondeau. *Introduction à la terminologie.* 1983.

[84] E. Rosch. *Principles of Categorization. Cognition and Categorization*, pages 27–48. 1978.

[85] E. Rusty Harold and W. S. Means. *XML.*

[86] A. Ryman. Understandig Web Services. Technical report, 2003.

[87] J. C. Sager. *A practical course in terminology procesing.* 1990.

[88] K. D. Schmitz. Concepts as building blocks for knowledge organization - a more ontological and less linguistic perception of terminology. Actes de la cinquième conférence TOTh, 2011.

[89] H. Schnurr, Y. Sure, and R. Studer. On-To-Knowledge Methodology - Baseline Version. *Communications of the ACM.*

[90] J. Sinclair. *Corpus, concordance collection.* 1991.

[91] L. Sjberg. *Gopher: underground technology.* 2004.

[92] D. Skuce. Covention for reaching agreement on shared ontologies. Proceedings of the 9th Know ledge Acquisition for Know ledge Based Systems Workshop, 1995.

[93] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. 2001.

[94] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, 1998.

[95] O. Suominen et al. Healthfinland a national semantic publishing network and portal for health information. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2009.

[96] B. Swartout et al. Toward distributed use of large-scale ontologies. Symposium on Ontological Engineering of AAAI, 1997.

[97] P. Timmers. Business models for electronic markets. *Electronic Markets*, 1998.

[98] M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 1996.

[99] M. Uschold and M. King. Towards a Methodology for Building Ontologies. 1995.

[100] G. van Heijst. Using explicit ontologies in KBS development. *International Journal of Human-Computer studies*, 1997.

[101] W3C. Resource description framework. `http://www.w3.org/RDF/`, 2004. Accesed October 2012.

[102] W3C. Web ontology language. `http://www.w3.org/TR/owl2-overview/`, 2009. Accesed October 2012.

[103] M. Wooldridge. *An Introduction to MultiAgent Systems*, pages 15–46. 2002.

[104] B. Wouters, D. Deridder, and E. Van Paesschen. The Use of Ontologies as a Backbone for Use Case Management. European Conference on Object-Oriented Programming (ECOOP 2000), Workshop : Objects and Classifications, a natural convergence, 2000.

[105] E. Wüster. *l'Étude scientifique générale de la terminologie, zone frontalière entre la linguistique, la logique, l'ontologie, l'informatique et les sciences des choses*. 1981.

[106] L. Youseff, M. Butrico, and D. da Silva. Towards a unified ontology of cloud computing. Grid Computing Environments Workshop, 2008.

[107] L. Yu. *Introduction to the Semantic Web and Semantic Web Services*. 2007.

[108] G. Zheng and A. Bouguettaya. Discovering pathways of service oriented biological processes. Proceedings of the Ninth International Conference on Web Information Systems Engineering, 2008.