2. ONTOLOGÍA. ANOTACIÓN SEMÁNTICA. HERRAMIENTAS

2.1. Introducción

En este primer capítulo se establecen las bases sobre las que se asienta el presente proyecto, de esta forma se comienza proporcionando unas nociones básicas sobre ontologías. Se analiza la estructura de formación de éstas, así como las tipologías que se pueden encontrar.

La temática de las ontologías da lugar, posteriormente, a establecer una metodología de construcción de ontologías y para ello, será necesario proporcionar una base conceptual sobre el modelado de ellas, que resulta vital para establecer los modelos de los conceptos que se necesitan desarrollar a lo largo del proyecto. Se tendrán que desarrollar modelos ontológicos de la propia norma, así como de los productos y actividades que regula.

Para implementar este modelado ontológico se hace uso de UML, un lenguaje de modelado que nos permitirá crear todos los esquemas conceptuales necesarios y de esta manera ontologías sobre ellos.

Una vez que tengamos definido el concepto de las ontologías y su proceso de modelado mediante UML, se busca la relación de la creación de estas ontologías con el marcado de textos o lo que se denomina anotación semántica. Mediante lenguajes de marcado o herramientas de anotación externa se podrá enriquecer un texto que, en el caso que se analiza en este proyecto, será una sección del Código Técnico de la Edificación.

De esta manera, para hacer efectivas las anotaciones semánticas en el texto normativo, será necesario establecer unas bases de conocimiento sobre los lenguajes de marcado y por ello, se muestra en un apartado toda la información referente a XML y sus tecnologías. Se analizan tecnologías como XMLSchema, que ayudan a esquematizar el contenido de un documento XML y posteriormente tras la aplicación del lenguaje de marcado, XML, y basándonos en las ontologías modeladas, se podrá enriquecer el texto para posteriormente mediante otras tecnologías de XML, como Xquery (tecnología de consulta de documentos XML), poder implementar herramientas de búsqueda y consulta más eficientes que las que actualmente nos brinda un simple procesador de texto o explorador Web.

2.2. Ontologías

El término ontología se ha empleado desde hace muchos siglos en el campo de la filosofía y del conocimiento y hace ya varias décadas cobró especial relevancia en el campo de la biblioteconomía y la documentación. Hoy ha sufrido un nuevo impulso debido al desarrollo de la Web Semántica donde prima la idea de transformar la red no sólo en un espacio de información, sino también en un espacio de conocimiento.

En el presente proyecto, mediante la creación de ontologías y el uso de las mismas se alcanza una nueva perspectiva del almacenamiento y búsqueda de información normativa basado en la categorización de los conceptos recogidos en los textos normativos.

2.2.1. Definición de Ontología y Objetivos

Una ontología se puede definir en general como una especificación explícita y formal de una conceptualización compartida.Esta definición se puede completar diciendo que una ontología:

a) Es explícita porque define los conceptos, propiedades, funciones, axiomas y restricciones que la componen.

b) Es formal porque es interpretable por máquinas.

c) Es una conceptualización porque es un modelo abstracto y vista simplificada de fenómenos del dominio que se quiere representar.

d) Finalmente, es compartida porque la información ha sido consensuada previamente entre distintos grupos de expertos.

Una vez establecida una definición formal de una ontología, se debería citar a continuación cual es el objetivo inmediato del uso de las mismas dentro de los sistemas de conocimiento. Por ello se puede decir que una ontología tiene como finalidad:

a) Comprender mejor algún área del conocimiento.

b) Comprensión por parte de otros comprendan algún área del conocimiento.

c) Consensuar la interpretación del conocimiento.

d) Permitir que las máquinas utilicen el conocimiento en alguna aplicación.

e) Permitir el intercambio de conocimiento entre las maquinas.

2.2.2. Componentes de modelo de conocimiento de las Ontologías

Las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio.

a) Clases o Subclases:
Son las ideas a formalizar y representan los conceptos en el sentido más amplio. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

Las clases en una ontología se suelen organizar en taxonomías a las que se les pueden aplicar los mecanismos de herencia. El concepto taxonomía, procedente del campo de la zoología y la botánica, es aplicable al campo de las ontologías ya que permite que conceptos o clases relacionadas puedan agruparse formando categorías de manera que resulte más fácil encontrar el término correcto bien para buscar o bien para describir un objeto.

A los componentes clases se les puede añadir información auxiliar que limitan las propiedades y características de los conceptos a los que representan. Estos subcomponentes, denominados atributos, van a ayudar a definir las características de las clases.

b) Relaciones:
Representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.

c) Funciones:
Son un tipo concreto de relación en las se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignar fecha, etc.

d) Instancias:
Se utilizan para representar objetos determinados de un concepto. Se usan para representar elementos o individuos en una ontología.

e) Axiomas:
Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición C1, A es B", etc.

Los axiomas formales sirven para modelar sentencias que son siempre ciertas. Normalmente se usan para representar conocimiento que no puede ser formalmente definido por los componentes descritos anteriormente. Además, también se usan para verificar la consistencia de la propia ontología.

2.2.3. Tipos de Ontologías

Dentro de la clasificación tipológica que se puede hacer de las ontologías se va hacer una primera estructuración entendiendo al nivel de componentes integrados dentro de las mismas, de manera que según la carga informativa que contenga, se puede hablar de ontologías ligeras o de ontologías pesadas:

a) Ontologías Ligeras ("Light-weight Ontology")
Este grupo de ontologías incluyen los conceptos, las taxonomías de los conceptos, las relaciones entre conceptos y las propiedades que describen esos conceptos. De esta manera, se puede decir que esta tipología de ontologías ligeras está muy ligada con tesauros.

b) Ontologías Pesadas ("Heavy-weight Ontology")
Esta otra tipología de ontologías se caracteriza principalmente por superar el ámbito de los tesauros en la medida que añaden axiomas y restricciones. Por ello las ontologías pesadas incluyen todos los elementos que permiten utilizarlas para realizar inferencias sobre el conocimiento que contienen.

Por otro lado, las ontologías se pueden categorizar según su alcance y posibilidad de aplicación y según esto se identifican cuatro tipos de ontologías

Hay ontologías de nivel más alto, las de dominios, las de tareas y las de aplicaciones. Las primeras destinadas a describir todos los conceptos. Las segundas describen el vocabulario relacionado con un dominio genérico, mientras que las ontologías de tareas describen actividades, lo que puede resultar útil en las organizaciones:

a) Ontologías de la Aplicación
Usadas por la aplicación, escriben los conceptos conforme a un campo determinado o unas tareas concretas, que resultan, en muchas ocasiones, especializaciones de diversas ontologías. Un claro ejemplo de esta tipología son las ontologías de procesos de producción, de diagnóstico de fallos, etc.

b) Ontologías del Dominio
Específicas para un tipo de concepto u objeto. Describen todos los elementos generales tales como el espacio, el tiempo, la materia, el objeto, el hecho, la acción, etc. Un ejemplo de estas ontologías es la ontología del proceso de producción industrial, proceso constructivo de un edificio, etc.

c) Ontologías Técnicas básicas
Describen características generales de artefactos. Por ejemplo: componentes, procesos, funciones.

d) Ontologías Genéricas:
Describen las categorías de más alto nivel.

Otra posible clasificación de las ontologías se podría realizar en función de su punto de vista, como por ejemplo: físico, de comportamiento, funcional, estructural, topológico, etc.

Por otro lado, las ontologías también se podrían caracterizar teniendo en cuenta su estructuración, es decir, según cómo se hayan estructurado; si están muy estructuradas o no, etc.

2.2.4. Metodología, Lenguajes, y Herramientas para Ontologías

En este punto se va a exponer la metodología más adecuada para la elaboración de una ontología de cualquier concepto u objeto y por la que se ha guiado este proyecto para la elaboración de las ontologías de productos, actividades y del propio texto normativo.

a) Metodología

1) Identificación del propósito y alcance de los objetivos necesarios.

2) Captura:

- Identificación de los conceptos y relaciones claves en el dominio de interés.
- Producción de definiciones no ambiguas de conceptos y de sus relaciones.
- Identificación de términos para referirse a estos conceptos y relaciones.

3) Codificación:

Representación explícita de la conceptualización en un lenguaje formal:

- Términos básicos de especificación (a veces llamado meta-ontología)
- Lenguaje de representación adecuado.
- Codificación de este lenguaje.
- Integración de ontologías existentes: cómo, cuáles y si se va a usar alguna ontología existente.

4) Evaluación:

Se considera que la ontología construida va a ser reutilizada por lo tanto debe seguir unos principios básicos:

- Abstracción: lo más abstracto posible pero suficientemente concreto.
- Modularización: permite aislar conceptos.
- Jerarquización: debe seguir un orden.
- Estandarización.

b) Documentación:

Debe de hacerse de forma paralela a los puntos anteriores y debe de contener esta clase de puntos:

- Tener el tipo de mapeo en que se basa la nueva teoría.
- Contener diferencias semánticas con las ontologías seleccionadas.
- Justificación de las decisiones tomadas.
- Evaluación.
- Conocimiento adicional para usarla, etc.

Además la ontología construida debe ser indexada y ordenada con las ontologías existentes para su posterior reutilización.

c) Lenguajes de Ontologías

Tal y como se expresa en el punto iii de la metodología anteriormente expuesta, la ontología debe ser codificada en un lenguaje adecuado. Para ello el lenguaje XML, que será analizado más adelante en este proyecto, proporciona una forma de escribir datos que es independiente de lenguajes, plataformas y herramientas y que proporciona una estructura sintáctica para que los datos puedan ser interpretados por computadoras. XMLS (el lenguaje de esquemas de XML) permite la definición de gramáticas y etiquetas significativas para los documentos a través de namespaces o espacios de nombre. Sin embargo, XML o XMLS no son suficientes ya que aportan una estructura, pero no una semántica. La semántica es aparente para los humanos, pero no para las máquinas. La semántica estudia cómo los símbolos se refieren a los objetos. Es necesaria más expresividad para el procesamiento semántico y de esta forma se creó el lenguaje RDF, como un lenguaje para modelar los datos. El lenguaje RDF mediante recursos, propiedades (atributos y relaciones para describir recursos) y sentencias (combinación de recursos y propiedades) permite una representación explícita de la semántica de los datos.

RDF carece de poder expresivo (negación, implicación, cardinalidad, etc.). Por ejemplo, no es posible especificar las condiciones necesarias y suficientes para definir la pertenencia a una clase. Para lograr una mayor expresividad para el procesamiento semántico, se han desarrollado nuevos estándares para la representación de ontologías que constriñen los vocabularios de descripción de recursos basados en RDF y RDF Schemas (RDFS). Tales ontologías permitirán, entre otras cuestiones, distribuir definiciones autorizadas de vocabularios que soporten referencias cruzadas como los

tesauros. Por lo tanto, las ontologías de representación están pensadas para que tomen el papel que hasta ahora ocupaban los tesauros normalizados, pero es preciso un lenguaje estándar que especifique dichas ontologías con mayor precisión que los estándares ISO sobre tesauros. Con ese fin se ha creado el lenguaje OWL, una especificación del W3C para especificar ontologías. Por ejemplo, OWL sí permite definir las condiciones necesarias y suficientes para definir la pertenencia a una clase, luego, aunque basado en el lenguaje RDF, va más allá que RDF Schema.

Las ontologías requieren de un lenguaje lógico y formal para ser expresadas. En la inteligencia artificial se han desarrollo numerosos lenguajes para este fin, algunos basados en la lógica de predicados, como KIF y Cycl que ofrecen poderosas primitivas de modelado, y otros basados en frames (taxonomías de clases y atributos), que tienen un mayor poder expresivo, pero menor poder de inferencia; e incluso existen lenguajes orientados al razonamiento como Description Logic y Classic. Todos estos lenguajes han servido para desarrollar otros lenguajes aplicables a la Web. En un lenguaje de ontologías se pretenderá un alto grado de expresividad y uso.

De entro los principales lenguajes de ontologías, destacan los siguientes:

1) SHOE (Simple HTML Ontology Extensions): Fue el primer lenguaje de etiquetado para diseñar ontologías en la Web. Este lenguaje nació antes de que se ideara la Web Semántica. Las ontologías y las etiquetas se incrustaban en archivos HTML. Este lenguaje permite definir clases y reglas de inferencia, pero no negaciones o disyunciones.

2) OIL (Ontology Inference Layer): Este lenguaje, derivado en parte de SHOE, fue impulsado también por el proyecto de la Unión Europea On-To-Knoledge. Utiliza ya la sintaxis del lenguaje XML y está definido como una extensión de RDFS. Se basa tanto en la lógica descriptiva (declaración de axiomas) y en los sistemas basados en frames (taxonomías de clases y atributos). La principal carencia de este lenguaje es la falta de expresividad para declarar axiomas.

3) DAML y OIL: Este lenguaje nació fruto de la cooperación entre OIL y DARPA y unifica los lenguajes DAML (DARPA's Agent Markup Language) y OIL (Ontology Inference Layer). Se basa ya en estándares del W3C(es un consorcio internacional que produce estándares para la World Wide Web). El lenguaje DAML se desarrolló como una extensión del lenguaje XML y de Resource Description Framework (RDF) y para extender el nivel de expresividad de RDFS. DAML- OIL hereda muchas de las características de OIL, pero se aleja del modelo basado en clases (frames) y potencia la lógica descriptiva.

4) OWL (*OWL Web Ontology Language*) Es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del W3C, y puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos (ontologías). En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste. Se trata de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL surge como una revisión al lenguaje DAML-

OIL y es mucho más potente que éste. Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes (motores de búsqueda, agentes, etc.).

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

2.2.5. Aplicaciones para este proyecto

Mediante todos los conceptos expuestos en este punto se podrán crear metodologías de trabajo para generar las ontologías de aplicación a la temática que desea abordar el proyecto. Desde el punto de vista de necesidades, el proyecto requiere identificar los productos u objetos y actividades que normaliza el CTE. Una vez creadas estas ontologías, habrá que desarrollar la ontología en la que se basa el funcionamiento normativo de los textos, con la finalidad de categorizar la información que ordenan las disposiciones recogidas en las distintas secciones que componen el CTE.

Esta labor de modelado de la información se puede implementar desde un nivel inicial mediante un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software, permitiendo crear todas la ontologías necesarias que posteriormente serán formalizadas con los lenguajes de ontologías (XML y XMLS en este caso) expuestos anteriormente.

2.3. UML. Lenguaje de Modelado

A continuación se analizan los fundamentos del lenguaje de modelado UML necesarios para crear posteriormente los esquemas de ontologías referidos a los productos y actividades que se pueden aparecer y desarrollar, respectivamente, en un documento normativo. De la misma manera se aprovecha este lenguaje de modelado para crear una ontología de la estructura de un documento normativo.

2.3.1. Definición de UML y sus aplicaciones.

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar. A través del modelado se consiguen cuatro objetivos:

a) Los modelos ayudan a visualizar un sistema.

b) Los modelos permiten especificar la estructura o el comportamiento del sistema.

c) Los modelos proporcionan una plantilla que guía en la construcción del sistema.

d) Los modelos documentan las decisiones tomadas.

El modelado de sistemas mediante orientación a objetos pretende representar el mundo real, en el que se desarrolla la actividad del hombre, mediante la representación de objetos.

El principal bloque constructivo del modelado orientado a objetos es el objeto o la clase. Un objeto es una cosa, generalmente extraída del vocabulario del espacio del problema o del espacio de solución; una clase es una descripción de un conjunto de objetos similares, que contiene atributos y las operaciones sobre estos atributos que hacen que una clase tenga la entidad que se desea.

Una ventaja clave de un modelado orientado a objetos es la posibilidad de aumentar su funcionalidad, ampliando los componentes existentes, añadiendo nuevos objetos al sistema, e incorporando fácilmente nuevas visiones o planteamientos de problemas que afecten al mismo sistema.

En cada caso concreto, es necesario establecer una semántica y una sintaxis que permitan trabajar con los conceptos involucrados en el problema objeto de análisis.

El ámbito de aplicación del lenguaje de modelado UML en este proyecto está enfocado a la visualización, descripción y esquematización de las estructuras y subestructuras conceptuales que aparecen en el mismo. Nos referimos a estructuras conceptuales a todos aquellos elementos que se desean analizar tales como productos que regula la norma, actividades que describe la norma y la propia norma en sí, que también será modelizada para desarrollar la aplicación de clasificación y búsqueda de la información del contenido normativo.

2.3.2. Diagramas UML

UML se compone de muchos elementos de esquematización que representan las diferentes partes de un sistema. Los elementos UML se utilizan para crear diagramas, que representa alguna parte o punto de vista del sistema.

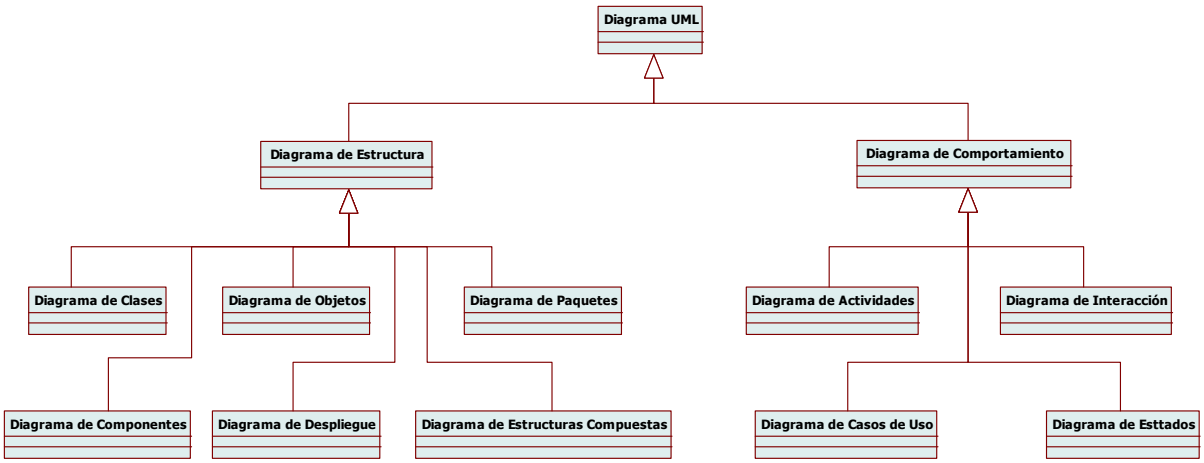


Fig. 2.1 - Jerarquía de Diagramas UML

Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente tal y como se puede apreciar en la figura 2.1 adjunta:

a) Diagramas de Estructura
Enfatizan en los elementos que deben existir en el sistema modelado:

1. Diagrama de clases.
2. Diagrama de componentes.
3. Diagrama de objetos.
4. Diagrama de estructura compuesta.
5. Diagrama de despliegue.
6. Diagrama de paquetes.

b) Diagramas de Comportamiento
Enfatizan en lo que debe suceder en el sistema modelado:

1. Diagrama de actividades.
2. Diagrama de casos de uso.
3. Diagrama de estados.
4. Diagramas de Interacción

   Son un subtipo de diagramas de comportamiento, que enfatizan sobre el flujo de control y de datos entre los elementos del sistema modelado:

   1. Diagrama de secuencia.
   2. Diagrama de colaboración.
   3. Diagrama de tiempos.
   4. Diagrama de vista de interacción.

En el presente proyecto sólo se va a hacer uso de tres tipologías de diagramas UML de los anteriormente citados. Los diagramas UML, que se emplean en el Capítulo 4 para modelar la estructura conceptual que relaciona a las disposiciones normativas, las Entidades de Producto y las Actividades, serán los Diagramas de Casos de Uso, los Diagramas de Clases y los Diagramas de Actividad

A continuación se dan unas nociones básicas de los diagramas que se emplearan en la elaboración de este proyecto:

a) Diagrama de Casos de Uso:
Un diagrama de casos de uso es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones. Todo sistema tiene como mínimo un diagrama principal de casos de uso, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso).

Un caso de uso, denotando un requisito funcional exigido al sistema, se representa en el diagrama por una elipse y un nombre significativo. En el caso del ejemplo se tienen como casos de uso de la máquina de café RecibirDinero, Pedir Azúcar, PedirProducto, DarVueltas y Cancelar.

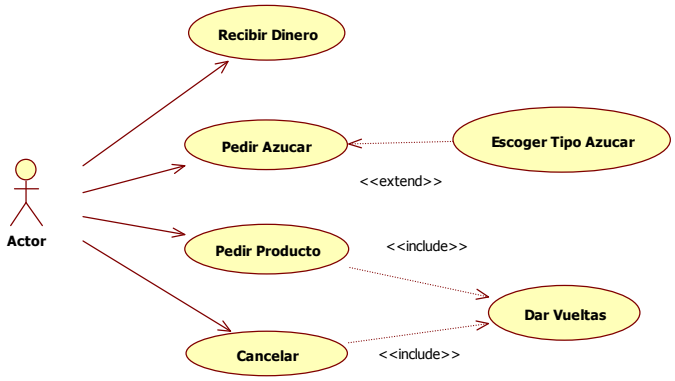


Fig. 2.2 - Ejemplo Máquina Café

En un diagrama de casos de usos solo existen tres relaciones posibles:

1)Inclusión (Include): Es una forma de interacción, un caso de uso dado puede "incluir" otro. El primer caso de uso a menudo depende del resultado del caso de uso incluido. Esto es útil para extraer comportamientos verdaderamente comunes desde múltiples casos de uso a una descripción individual.

2)Extensión (Extend): Es otra forma de interacción, un caso de uso dado, (la extensión) puede extender a otro. Esta relación indica que el comportamiento del caso de uso extensión puede ser insertado en el caso de uso extendido bajo ciertas condiciones. La notación es una flecha rayada desde el caso de uso extensión al caso de uso extendido, con la etiqueta «extend». Esto puede ser útil para lidiar con casos especiales, o para acomodar nuevos requisitos durante el mantenimiento del sistema y su extensión.

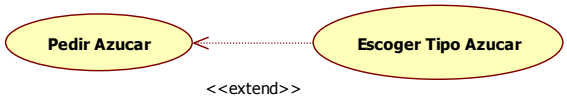


Fig. 2.3 - Ejemplo de Extensión

3) Generalización: En la tercera forma de relaciones entre casos de uso, existe una relación generalización/especialización. Un caso de uso dado puede estar en una forma especializada de un caso de uso existente. La notación es una línea solida terminada en un triángulo dibujado desde el caso de uso especializado al caso de uso general.

b) Diagrama de Clases:

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.).

Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño).

Los diagramas tendrán una parte de destinada a los elementos que los forman (clases y objetos) y las relaciones que existen entre los mismos (asociaciones).

Una clase se representa mediante una caja subdividida en tres partes: En la superior se muestra el nombre de la clase, en la media los atributos y en la inferior las operaciones. Una clase puede representarse de forma esquemática, con los atributos y operaciones suprimidos, siendo entonces tan solo un rectángulo con el nombre de la clase. Ejemplo de representación de una clase mostrado por la figura 2.4.

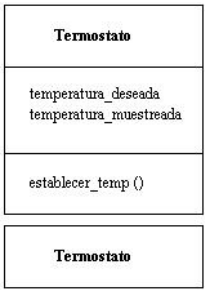


Fig. 2.4 - Ejemplo de Clase

Las relaciones entre clases se documentan con una descripción de su propósito, su cardinalidad (cuantos objetos intervienen en la relación) y su opcionalidad (cuando un objeto es opcional el que intervenga en una relación).

1) Asociación (conexión entre clases):
Una asociación es una relación estructural que describe una conexión entre objetos. Gráficamente se muestra como una línea continua que une las clases relacionadas entre sí.

Fig. 2.5 - Ejemplo de Asociación

En las asociaciones es muy importante definir la navegación de la relación de manera que, aunque suelen ser bidireccionales es deseable hacerlas unidireccionales. Gráficamente, cuando la asociación es unidireccional, la línea termina en una punta de flecha que indica el sentido de la asociación, tal y como se puede ver en la figura 2.6:

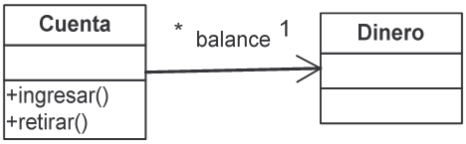


Fig. 2.6 - Asociación Unidireccional

En caso de que la relación sea bidireccional la línea de unión no presenta ningún tipo de flecha que indique la dirección, pues las dos son validas o posibles.

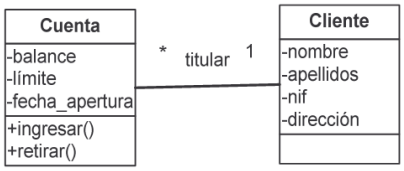


Fig. 2.7 - Asociación Bidireccional

Otra característica importante de las asociaciones es la multiplicidad de las relaciones (ver tabla 2.1). Las asociaciones tienen dos multiplicidades (una en cada extremo de la relación) y cuando se indica la multiplicidad de una asociación se debe indicar la multiplicidad mínima y la multiplicidad máxima (*mínima..máxima*).

Especificación de Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
n..m	Desde n hasta m
*	varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)

Tabla 2.1 - Multiplicidades

Hay que destacar que dentro de la tipología de relación asociación encontramos dos casos particulares en los que la relación se da entre un todo y sus partes:

2) Agregación:

En este caso las partes pueden formar parte de distintos agregados. Gráficamente (ver figura 2.8) se representan con un rombo vacio en el extremo donde representamos el todo.

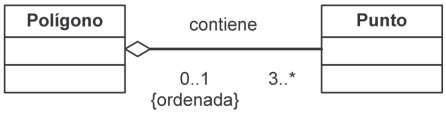


Fig. 2.8 - Ejemplo Agregación

3) Composición:

Se tratan de agregaciones separadas y estrictas. Las partes solo existen asociadas al componente principal. Gráficamente se representan con un rombo lleno en el extremo donde representamos el componente principal, tal y como se representa en el ejemplo de la figura 2.9:

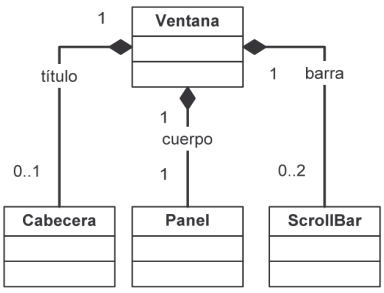
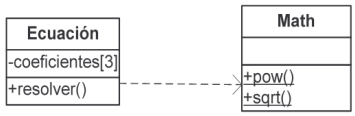


Fig. 2.9 - Ejemplo Composición

4) Dependencia (Relaciones de uso):

Es una relación más débil que una asociación, que muestra la correspondencia entre un cliente y un proveedor. Se puede decir que el cliente es el que solicita el servicio y el servidor o proveedor es el objeto que proporciona el servicio solicitado. Gráficamente, la dependencia se representa con una línea discontinua con una punta de flecha que establece la dirección cliente-proveedor.

Como se puede apreciar en el ejemplo de la figura 2.10 para realizar la actividad +resolver() hay que recurrir a la clase Math que contiene la función +sqrt() para calcular la raíz cuadrada.



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Fig. 2.10 - Ejemplo de Dependencia

5) Generalización (Relaciones de Herencia):

Por último, la generalización es una relación entre una superclase y una subclase. Mediante este tipo de relación se pueden clasificar objetos (clases) que comparten atributos u operaciones.

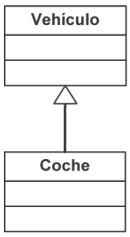


Fig. 2.11 - Ejemplo de Generalización

Mediante las generalizaciones es posible crear lo que se denomina jerarquías de clases, en las que los atributos u operaciones de una categoría más general son aplicables a una categoría más particular. Las subclases, tal y como se puede ver en el ejemplo de la figura 2.11 heredan las características de las case de las que se derivan y añaden características especificas que las diferencian del resto de clases de la jerarquía.

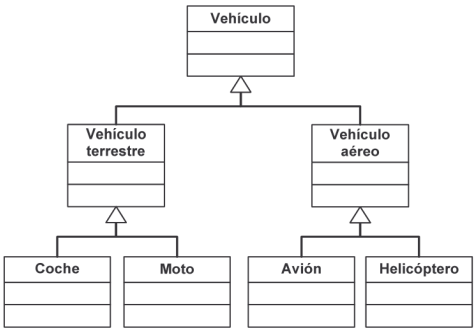


Fig. 2.12 - Ejemplo de Jerarquía

Es así como las clases se organizan mediante esta estructura de jerarquías formando lo que se denominan taxonomías.

c) Diagrama de Actividades:
Un diagrama de actividades puede considerarse como un caso especial de un diagrama de estados en el cual casi todos los estados son estados acción (identifican una acción que se ejecuta al estar en él) y casi todas las transiciones evolucionan al término de dicha acción (ejecutada en el estado anterior). Un diagrama de actividades puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto.

2.4. Anotación Semántica

Al principio, los documentos Web se hacían a mano y nacían orientados al intercambio de información entre las personas. Todos estos documentos constituyen una enorme cantidad de texto, imágenes e incluso sonido, sin significado para una computadora. El usuario era el encargado de extraer e interpretar la información relevante. Actualmente, dado el asombroso crecimiento de la información contenida en Internet, resulta imposible que un único usuario realice estas tareas en un tiempo aceptable. Al mismo tiempo, han surgido nuevas tecnologías que facilitan la gestión y recuperación de la información y, con ellas, aparece también la generación semi-automática de documentos Web.

2.4.1. Definición de Anotación Semántica y sus aplicaciones.

Según el diccionario de la RAE, anotación "constituye la acción y efecto de anotar", y anotar, en su principal acepción, "poner notas en un escrito, una cuenta o un libro". En lingüística computacional una anotación es una nota añadida a una parte específica de un texto. Existen dos alternativas para realizar anotaciones: anotaciones incrustadas en el texto (en las páginas Web) creadas mediante los lenguajes de marcado (XML) o anotaciones externas, no incrustadas en las páginas Web.

Una anotación se puede considerar como una información sobre las entidades o conceptos de una ontología que aparecen en un texto y su situación en el mismo, o también las referencias que hay en un texto sobre un repositorio semántico en el que hay más conocimiento.

En la actualidad, la presentación en la Web de este tipo de documentos tiende a tratarse junto con su contenido, principalmente mediante la utilización de XML, RDF, RDF Schema u OWL. Pero aunque con ello se facilite el procesamiento automático de la información, un ordenador no puede llevar a cabo por sí solo las tareas de acceso, extracción e interpretación de la información relevante. Conseguir que las máquinas entiendan el significado, la semántica, de los textos escritos y de las páginas Web es uno de los pilares principales que sustenta el desarrollo de la Web Semántica. En este contexto, la "anotación semántica" de páginas Web, que hace explícito el significado de un concepto para un ordenador, se ha convertido en un punto clave y de tremenda aplicación para cualquier tipo de documento de las mismas características al que se le quiera aplicar la misma filosofía de procesamiento.

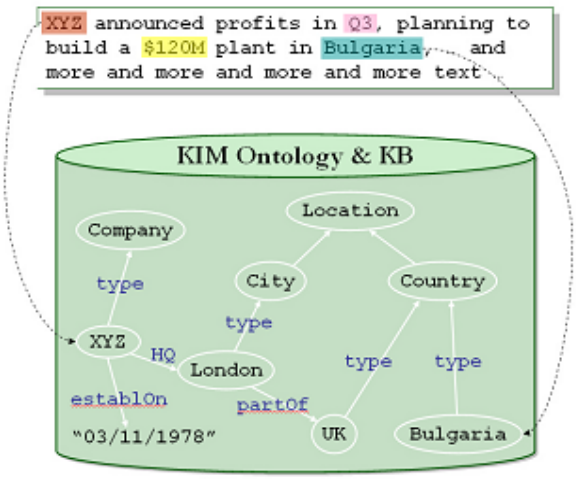


Fig. 2.13 - Ejemplo de Anotación Semántica

2.4.2. Herramientas de anotación.

Las herramientas de anotación permiten añadir contenido semántico a las páginas Web de Internet. Es decir, estas herramientas permiten estructurar la información publicada mediante su clasificación en base a conceptos semánticos. Este es el primer paso para permitir el procesamiento automático de la información de la Web por parte de las máquinas, es decir, para la creación de la Web semántica.

Alguno de los requisitos que deben cumplir las herramientas de anotación semántica son los siguientes:

a) Consistencia. Las estructuras semánticas deben de adherirse a una ontología dada para permitir un mejor intercambio de conocimiento.

b) Referencias propias. Los identificadores de instancias (por ejemplo de personas, empresas, organizaciones, etc.) deben de ser únicos.

c) Evitar redundancias. Debe de ser posible que los usuarios logren identificar si un documento ya tiene anotaciones, y en su caso permitir la reutilización de material previamente desarrollado.

d) Metadatos relacionales. Al igual que la información en HTML, la cual está diseminada en la web, la marcación del conocimiento (knowledge markup) puede estar distribuida, pero debe de estar semánticamente relacionada.

e) Mantenimiento. La herramienta debe de permitir labores de mantenimiento de una manera clara y simple.

f) Facilidad de uso y eficiencia. Obviamente el ambiente de anotación debe de ser fácil de usar para ser realmente útil. La eficiencia en la creación de metadatos permitirá que los usuarios creen una mayor cantidad de metadatos.

Una vez establecidos los requisitos que han de cumplir las herramientas de anotación semántica, se pueden englobar en dos grupos:

a) Herramientas de Anotación Externa:
Estas herramientas permiten asociar meta información a páginas Web que ya están presentes en la Web. La meta información no se almacena dentro de la Web, sino que se almacena de forma externa en un repositorio destinado específicamente a mantener las anotaciones. Estos repositorios suelen ser bases de datos RDF. Las anotaciones se guardan clasificadas como objetos o propiedades correspondientes a un concepto de una ontología. Por tanto, como herramientas complementarias a las de anotación se encuentran los editores de ontologías, que facilitan la tarea de definición de ontologías, la unión de diferentes ontologías, y el desarrollo distribuido de ontologías.

Las herramientas de anotación externa son útiles para realizar comentarios personales sobre páginas Web, para mantener discusiones en grupo sobre estas páginas, para compartir información, etc.

Ejemplos de herramientas de anotación externa que se han evaluado son:

1) COHSE (The Conceptual Open Hypermedia Project): Proyecto de investigación sobre métodos que mejoren significativamente la calidad, consistencia y la amplitud de documentos Web enlazados mientras se recuperan (cuando los lectores navegan sobre los documentos) y al mismo tiempo que se crean (cuando los autores crean los documentos). COHSE utiliza tres tecnologías: un servicio de razonamiento de ontologías que se utiliza para representar un sofisticado modelo conceptual de términos documentales y sus relaciones, un servicio abierto de enlaces hipermedia basado en Web, que sea escalable, y la integración de estos dos para poder enlazar documentos vía meta datos describiendo sus contenidos.

2) Annotea: Proyecto que pretende mejorar el ambiente colaborativo de la W3C (el "Consorcio de la World Wide Web") a través del uso de anotaciones compartidas; se entiende que una anotación puede ser un comentario, una nota, una explicación o cualquier texto que se pueda adjuntar a un documento Web externamente, es decir, sin necesidad de tocar el documento mismo. El usuario recibe las notas de un servidor o varias adjuntas a un documento junto con éste y puede ver qué piensan sus compañeros del documento.

3) Yawas (Yet Another Web Annotation System): YAWAS es una herramienta que permite a los usuarios realizar anotaciones para expresar sus opiniones y personalizar los documentos a medida que analizan la información en páginas Web.

b) Herramientas de anotación de autor:
Estas herramientas permiten incluir la información estructurada dentro de las propias páginas. Esta información se incluye normalmente en alguno de los lenguajes de marcado promovidos por el W3C (Consorcio World Wide Web), como son XML o RDF.

Las herramientas de anotación de autor están más orientadas a la Web semántica, y para que sean de utilidad para el usuario final, requieren el desarrollo adicional de agentes inteligentes que entiendan las ontologías, sean capaces de realizar búsquedas inteligentes y de extraer información.

Las herramientas de anotación de autor que se han evaluado son:

1) MnM: MnM es una herramienta de anotación basada en ontologías que permite anotar páginas Web con contenidos semánticos de forma automática y semi-automática. MnM integra un navegador Web con un editor de ontologías y proporciona unas APIs abiertas para enlazar MnM con servidores de ontologías y para integrar MnM con herramientas de extracción de información. MnM trabaja con diferentes lenguajes de ontologías como RDF, DAML+OIL y OCML.

2) OntoMat-Annotizer: Herramienta de anotación de páginas Web interactiva y amigable. Ayuda al usuario en la tarea de crear y mantener ontologías basadas en DAML+OIL, por ejemplo, para crear instancias, atributos y relaciones. Incluye un navegador de ontologías para explorar la ontología y las instancias y un navegador HTML que visualiza las partes anotadas del texto.

3) SHOE Knowledge Annotator: Herramienta que permite realizar anotaciones en documentos Web sin tener que preocuparse de los códigos HTML pues añade las etiquetas necesarias automáticamente. Las etiquetas están divididas en dos categorías: para la construcción de ontologías y para anotar documentos Web para suscribirlos a una o más ontologías, declarar entidades de datos y realizar afirmaciones sobre las entidades según las normas establecidas por las ontologías.

Al ser la Web semántica una arquitectura que acaba de emerger, las herramientas de anotación existentes no están del todo maduras por ello para hacer un uso más generalizado de ellas no se ha recurrido a ninguna herramienta de las anteriormente expuestas sino que se ha utilizado la base en las que su arquitectura se desarrolla; los lenguajes de marcado. Uno de los lenguajes de marcado más utilizados y extendidos es XML sobre el que se desarrolla el próximo apartado.

2.5. Lenguaje de Marcado Semántico; XML

2.5.1. Lenguajes de Marcado

Un lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. El lenguaje de marcado más extendido es el HTML, fundamento de la World Wide Web. Los lenguajes de marcado suelen confundirse con lenguajes de programación, sin embargo, estos no son lo mismo, ya que el lenguaje de marcado no tiene las funciones aritméticas que tienen los lenguajes de programación, como las variables.

Históricamente, el marcado se usaba y se usa en la industria editorial y de la comunicación, así como entre autores, editores e impresores.

2.5.1.1. Clases de lenguajes de marcado

Se suele diferenciar entre tres clases de lenguajes de marcado, aunque en la práctica pueden combinarse varias clases en un mismo documento.

a) Marcado de Presentación

El marcado de presentación es aquel que indica el formato del texto. Este tipo de marcado es útil para maquetar la presentación de un documento para su lectura, pero resulta insuficiente para el procesamiento automático de la información. El marcado de presentación resulta más fácil de elaborar, sobre todo para cantidades pequeñas de información. Sin embargo resulta complicado de mantener o modificar, por lo que su uso se ha ido reduciendo en proyectos grandes en favor de otros tipos de marcado más estructurados.

Se puede tratar de averiguar la estructura de un documento de esta clase buscando pistas en el texto. Por ejemplo, el título puede ir precedido de varios saltos de línea, y estar ubicado centrado en la página. Varios programas pueden deducir la estructura del texto basándose en esta clase de datos, aunque el resultado suele ser bastante imperfecto.

b) Marcado de Procedimientos

El marcado de procedimientos está enfocado hacia la presentación del texto, sin embargo, también es visible para el usuario que edita el texto. El programa que representa el documento debe interpretar el código en el mismo orden en que aparece. Por ejemplo, para formatear un título, debe haber una serie de directivas inmediatamente antes del texto en cuestión, indicándole al software instrucciones tales como centrar, aumentar el tamaño de la fuente, o cambiar a negrita. Inmediatamente después del título deberá haber etiquetas inversas que reviertan estos efectos. En sistemas más avanzados se utilizan macros o pilas que facilitan el trabajo.

Algunos ejemplos de marcado de procedimientos son nroff, troff, TeX, y PostScript. Este tipo de marcado se ha usado en aplicaciones de edición profesional, manipulados por tipógrafos cualificados, ya que puede llegar a ser extremadamente complejo.

c) Marcado Descriptivo

El marcado descriptivo o semántico utiliza etiquetas para describir los fragmentos de texto, pero sin especificar cómo deben ser representados, o en qué orden. Los lenguajes expresamente diseñados para generar marcado descriptivo son el SGML y el XML.

Una de las virtudes del marcado descriptivo es su flexibilidad: los fragmentos de texto se etiquetan tal como aparecen en el contexto y no siguiendo una regla de etiquetado externa a la semántica del mismo. Estos fragmentos pueden utilizarse para más usos de los previstos inicialmente. Por ejemplo, los hiperenlaces fueron diseñados en un principio para que un usuario que lee el texto los pulse. Sin embargo, los buscadores los emplean para localizar nuevas páginas con información relacionada, o para evaluar la popularidad de determinado sitio web.

El marcado descriptivo también simplifica la tarea de reformatear un texto, debido a que la información del formato está separada del propio contenido. Por ejemplo, un fragmento indicado como cursiva (texto), puede emplearse para marcar énfasis o bien para señalar palabras en otro idioma. Esta ambigüedad, presente en el marcado presentacional y en el procedimental, no puede soslayarse más que con una tediosa revisión a mano. Sin embargo, si ambos casos se hubieran diferenciado

descriptivamente con etiquetas distintas, podrían representarse de manera diferente sin esfuerzo.

En resumen, los lenguajes de marcado son la herramienta fundamental en el diseño de la web semántica, aquella que no solo permite acceder a la información, sino que además define su significado, de forma que sea más fácil su procesamiento automático y se pueda reutilizar para distintas aplicaciones. Esto se consigue añadiendo datos adicionales a los documentos, por medio de dos lenguajes expresamente creados: el RDF (Resource description framework-Plataforma de descripción de recursos) y OWL (Web Ontology Language-Lenguaje de ontologías para la web), ambos basados en XML.

2.5.2. XML, usos y ventajas

XML proviene de un lenguaje inventado por IBM en los años setenta, llamado GML (Generalized Markup Language), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje gustó a la ISO, por lo que en 1986 trabajaron para normalizarlo, creando SGML (Standard Generalized Markup Language), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros sistemas para almacenar información.

XML, sigla en inglés de Extensible Markup Language («lenguaje de marcado extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

2.5.2.1. Tipos de Documentos XML

Lo primero que se debe analizar es que existen dos tipos de documentos XML: validos y bien formados. Este es uno de los aspectos más importantes de este lenguaje, así que esto requiere mostrar bien cuál es la diferencia:

a) Bien formados:
   Los documentos denominados como "bien formados" (del inglés well formed) son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, ser analizados correctamente por cualquier analizador sintáctico (parser) que cumpla con la norma.

b) Válidos:
Son aquellos documentos XML que, además de estar bien formados, siguen una estructura y una semántica determinada por un DTD o XML-schema: sus elementos y sobre todo la estructura jerárquica que define el DTD o e Schema, además de los atributos, deben ajustarse a lo que estos dicten.

2.5.2.2. Estructura de un Documento XML

La tecnología XML da solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de pedazos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez con notas. Estas partes se llaman elementos y se las señala mediante etiquetas.

Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.

Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.

Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.

Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos "entendibles" por las personas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

A continuación se muestra un ejemplo referido al proceso de marcado semántico de un texto administrativo que contienen los datos personales de un estudiante universitario y será analizado detalladamente para entender la estructura de un documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<matricula xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance " xsi:noNamespaceSchemaLocation="prueba.xsd">
<!--- Datos de Alumno y Matricula --->
        <matricula>
                <personal>
                        <dni>52945813C</dni>
                        <nombre>José Estévez Moñux </nombre>
                        <titulacion>Ingeniería Industrial</titulacion>
                        <curso_academico>2007/2008</curso_academico>
                        <domicilios>
                                <domicilio tipo="familiar">
                                        <nombre>C/ Principal nº1</nombre>
                                </domicilio>
                                <domicilio tipo="habitual">
                                        <nombre>C/ Secundaria nº2</nombre>
                                </domicilio>
                        </domicilios>
                </personal>
                <pago>
                        <tipo_matricula>Matrícula ordinaria</tipo_matricula>
                </pago>
        </matricula>
```

Fig. 2.14 - Ejemplo Documento XML

2.5.2.3. Partes de un documento XML

Un documento XML está formado por dos partes claramente diferenciadas; por el prólogo y por el cuerpo del documento.

a) Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<matricula xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance " xsi:noNamespaceSchemaLocation="prueba.xsd">
<!--- Datos de Alumno y Matricula --->
```

Fig. 2.15 - Prólogo de Documento XML

El prólogo contiene:

1) Una declaración XML. Es la sentencia que declara al documento como un documento XML en ella se recoge la siguiente información:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Fig. 2.16 - Declaración

i. *version*: Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluya.

ii. *encoding*: La forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser que analiza el texto (Proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical con respecto a una gramática formal dada). Por defecto es UTF-8, aunque podrían usarse

otras, como UTF-16, US-ASCII, ISO-8859-1, etc. No es obligatorio salvo que sea un documento externo a otro principal.

iii. *standalone*: Indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"); en principio no hay porque ponerlo, porque luego se indica el DTD si se necesita.

2) Una declaración de tipo de documento. Enlaza el documento con su DTD o en el caso de que queramos definir la estructura del documento mediante un XML-Schema se hace referencia al mismo mediante la siguiente línea de comandos tal y como se muestra a continuación.

```
<matricula xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance " xsi:noNamespaceSchemaLocation="prueba.xsd">
```

Fig. 2.17 - Declaración Tipo Documento

3) Comentarios. Uno o más comentarios e instrucciones de procesamientos o aclaración del contenido que queda marcado en el texto y se expresa mediante la siguiente sintaxis. Los comentarios comienzan por la cadena "<!--" y terminan con "-->".

```
<!--- Datos de Alumno y Matricula --->
```

Fig. 2.18 - Comentario

b) Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML. Es la parte principal del documento XML. Es obligatorio crear por lo menos un elemento en el documento, el cual ayude a reconocer, de manera general, de que tratarán los datos marcados. En este documento el elemento documento es el elemento "<matricula>". Esta característica es indispensable para que el documento esté bien formado.

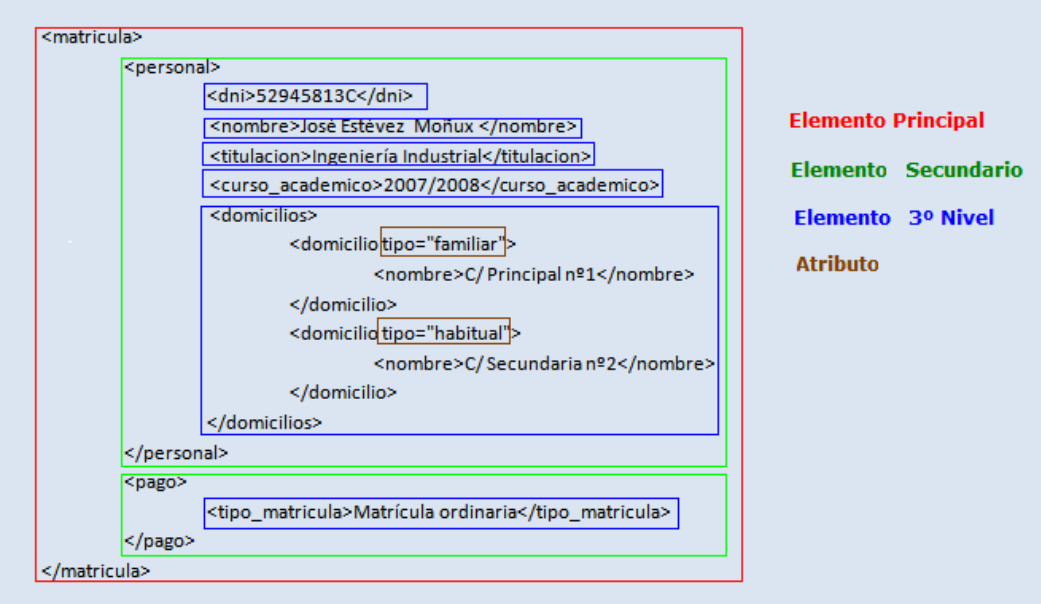


Fig. 2.19 - Cuerpo de Documento XML

1) Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos. Siempre empieza con una <etiqueta> que puede

contener atributos o no, y termina con una </etiqueta> que debe tener el mismo nombre. Hay que tener en cuenta que el símbolo "<" siempre se interpreta como inicio de una etiqueta XML. Si no es el caso, el documento no estará bien-formado. Para usar ciertos símbolos se usan las entidades predefinidas, que se explican más adelante.
En el ejemplo expuesto se contempla un elemento general denominado <matricula> el cual a su vez acoge a otros dos elementos; <personal> y <pago>.

2) Atributos
Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben de ir entre comillas.
Por ejemplo, un elemento "domicilio" puede tener un atributo "tipo" en el cual se almacena la naturaleza del domicilio que posee el alumno (familiar o habitual, en este caso)

```
<domicilio tipo="familiar">
        <nombre>C/ Principal nº1</nombre>
</domicilio>
```

Fig. 2.20 - Declaración Atributo

En una Definición de Tipo de Documento (más adelante se mostrara este concepto), se especifican los atributos que puede tener cada tipo de elemento, así como sus valores y tipos de valor posible.

3) Entidades predefinidas
En XML 1.0, se definen cinco entidades para representar caracteres especiales y que no se interpreten como marcado por el procesador XML. Es decir, que así se puede usar el carácter "<" sin que se interprete como el comienzo de una etiqueta XML, por ejemplo:

Entidad	Carácter
&amp;	&
&lt;	<
&gt;	>
&apos;	'
&quot;	"

Tabla 2.2 - Entidades Predefinidas

4) Secciones CDATA
Existe otra construcción en XML que permite especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como marcado XML. La razón de esta construcción llamada CDATA (Character Data) es que a veces es necesario para los autores de documentos XML, poder leerlo fácilmente sin tener que descifrar los códigos de entidades. Especialmente cuando son muchas.

Como ejemplo, el siguiente (primero usando entidades predefinidas, y luego con un bloque CDATA):

```
<parrafo>Lo siguiente es un ejemplo de HTML.</html>
<ejemplo>
&lt;HTML>
&lt;HEAD>&lt;TITLE>Rock &amp; Roll&lt;/TITLE>&lt;/HEAD>
</ejemplo>

<ejemplo>
<![CDATA[
<HTML>
<HEAD><TITLE>Rock & Roll</TITLE></HEAD>
]]>
</ejemplo>
```

Fig. 2.21 - Ejemplo Documento XML

Como se ha visto, dentro de una sección CDATA se puede poner cualquier cosa, que no será interpretada como algo que no es. Existe empero una excepción, y es la cadena "]]>" con la que termina el bloque CDATA. Esta cadena no puede utilizarse dentro de una sección CDATA.

5) Comentarios
Es conveniente insertar comentarios en el documento XML, que sean ignorados por el procesado de la información y las reproducciones del documento, pero que proporcionen en futuras revisiones del mismo información sobre el marcado realizado y sus elementos. Los comentarios comienzan por la cadena "<!--" y terminan con "-->".

2.5.2.4. Ventajas del uso de XML

Las principales ventajas del uso de XML son:

a) XML es un lenguaje independiente de la plataforma sobre la que se trabaje.

b) Es UNICODE, lo que hace que pueda ser utilizado en múltiples lenguajes.

c) Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

d) El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

e) XML sigue un estándar. Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

2.5.3. Esquemas de Contenido; XMLSchema

Un documento XML puede contener muchos tipos de información. Es decir, pueden haber muchos lenguajes escritos en XML para cualquier colectivo de usuarios. Por ejemplo:

- Si lo utiliza el colectivo de ingenieros podría crear un lenguaje en XML específico para almacenar resultados de los diseños. Este lenguaje se podría llamar IngenierosML.

- Si los arquitectos utilizan XML podrán crear sus propios lenguajes para guardar la información de los edificios. Este lenguaje se podría llamar ArquitectosML.

Como se ve, se pueden crear infinitos lenguajes a partir del XML. Para especificar cada uno de los usos de XML, o lo que es lo mismo, para especificar cada uno de los sublenguajes que se pueden crear a partir de XML, se utilizan unos lenguajes propios.

Son unos lenguajes que sirven para definir otros lenguajes, es decir, son metalenguajes. Se definen especificando qué etiquetas se encuentran en los documentos HTML, en qué orden, dentro de qué otras, además de especifican los atributos que pueden o deben tener cada una de las etiquetas.

Hay dos metalenguajes con los que definir los lenguajes que se puede obtener a partir de XML, el DTD y el XML Schema.

El DTD, Definition Type Document, tiene una sintaxis especial, distinta de la de XML, que es sencilla, aunque un poco rara si nunca se ha visto un documento similar.

Para evitar el DTD, que tiene una sintaxis muy especial, se intentó encontrar una manera de escribir en XML la definición de otro lenguaje XML. Se definió entonces el lenguaje XML Schema y funciona bien, aunque puede llegar a ser un poco más complicado que especificarlo en DTD. Simplemente nos ahorramos de aprender un nuevo lenguaje con su sintaxis particular.

De XML Schema se puede decir que es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así, una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C) y alcanzó el nivel de recomendación en mayo de 2001.

La principal aportación de XML Schema es el gran número de los tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

2.5.3.1. Componentes

Los esquemas se construyen a partir de diferentes tipos de componentes:

a) Elemento (element)

b) Atributo (attribute)

c) Tipo simple (simple type)

d) Tipo complejo (complex type)

e) Notación (notation)

f) Grupo modelo nombrado (named model group)

g) Grupo de atributos (attribute group)

h) Restricción identidad (identity constraint)

Un ejemplo de definición con XML Schema sería el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema " xml:lang="ES">
     <xs:element name="matricula" type="tMatricula"/>

     <xs:complexType name="tMatricula">
       <xs:sequence>
          <xs:element name="personal" type="tPersonal"/>
          <xs:element name="pago" type="tPago"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="tPersonal">
          <xs:all>
                <xs:element name="dni" type="xs:string"/>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="titulacion" type="xs:string"/>
                <xs:element name="curso_academico" type="xs:string"/>
                <xs:element name="domicilios" type="tDomicilio"/>
          </xs:all>
     </xs:complexType>

     <xs:complexType name="tPago">
          <xs:all>
               <xs:element name="tipo_matricula" type="xs:string"/>
          </xs:all>
     </xs:complexType>

     <xs:complexType name="tDomicilio">
      <xs:sequence>
       <xs:element name="domicilio" maxOccurs="unbounded">
        <xs:complexType>
         <xs:all>
          <xs:element name="nombre" type="xs:string"/>
         </xs:all>
         <xs:attribute name="tipo" type="xs:string"/>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:schema>
```

Fig. 2.22 - Ejemplo de XMLSchema

Para comprender mejor la estructura de un esquema se adjunta una figura con la representación de cada elemento definido en el mismo.

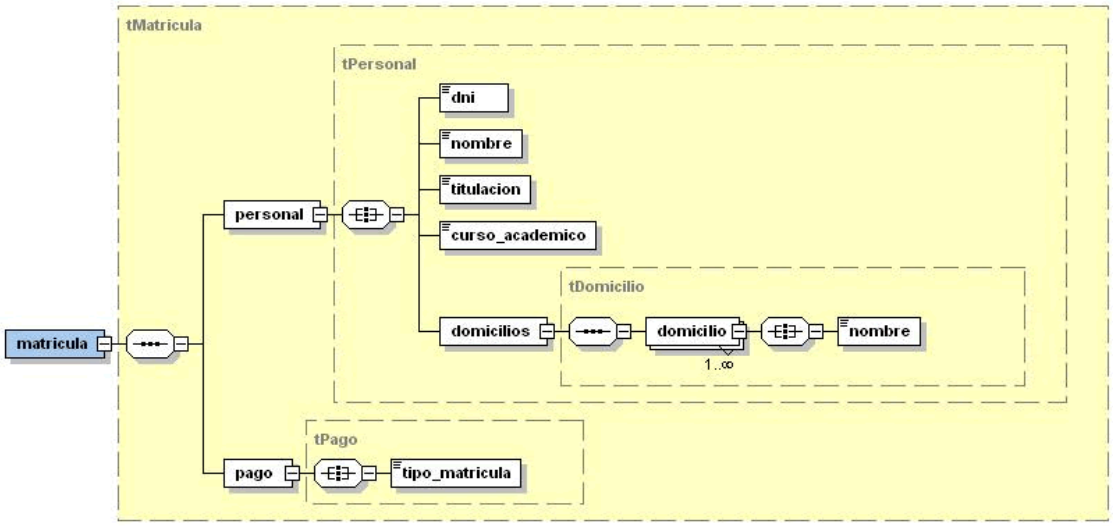


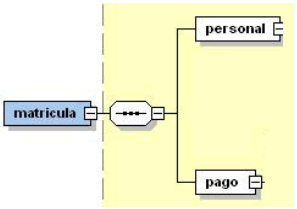
Fig. 2.23 - Mapa de un XMLSchema

Se puede ver cómo, de la misma que en un documento XML, se inicia la declaración indicando la versión de XML que se va a utilizar y la codificación que se usa. Estos dos campos son necesarios para poder interpretar el esquema. Además, en la siguiente línea de código se puede ver como se redirecciona al usuario a la página que ofrece las pautas de creación de XML Schema en las que se basa la descripción del esquema.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema " xml:lang="ES">
```

Fig. 2.24 - Prólogo

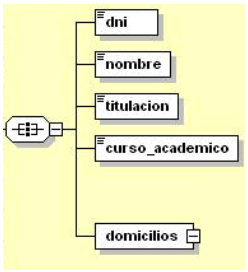
A continuación se comienza describiendo todos los elementos posibles que tendrá el documento XML. El elemento raíz se llama "matricula" y tiene dos elementos anidados, tal y como se puede ver en la figura 2.24; personal y pago, los cuales se definen mediante una relación de secuencia. Este tipo de relación establece que los datos en el documento XML deberán aparecer tal y como se definen en el esquema, de esta manera no se podrá establecer un marcado de texto de un elemento tipo "pago" sin haber definido antes un elemento de tipo "personal". Como su nombre, indica una secuencia de generación de elementos.




```
<xs:sequence>
   <xs:element name="personal" type="tPersonal"/>
   <xs:element name="pago" type="tPago"/>
</xs:sequence>
```

Fig. 2.25 y 2.26 - Esquema y Código de Elementos

Los elementos anidados a su vez definen más elementos. Por una parte se puede encontrar al elemento personal del que nace una relación de tipo "all". Este tipo de relaciones, no determinan un orden estricto de generación de elementos en el documento XML pero si obliga a que aparezcan todos.




```
<xs:all>
        <xs:element name="dni" type="xs:string"/>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="titulacion" type="xs:string"/>
        <xs:element name="curso_academico" type="xs:string"/>
        <xs:element name="domicilios" type="tDomicilio"/>
</xs:all>
```

Fig. 2.27 y 2.28 - Esquema y Código de Relación tipo "all"

De la misma manera que se definen los elementos de un documento XML, se puede también predefinir los atributos que tendrán esos elementos así como el tipo de datos que almacenan.

Es el caso, en nuestro ejemplo, del atributo "tipo" que indica que naturaleza tiene el domicilio que posee el alumno matriculado.

```
<xs:element name="domicilio" maxOccurs="unbounded">
 <xs:complexType>
  <xs:all>
   <xs:element name="nombre" type="xs:string"/>
  </xs:all>
  <xs:attribute name="tipo" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

Fig. 2.29 - Código de Atributo

De una manera práctica se ha expuesto el uso de XMLSchema, el cual será de gran aplicación en parte del proceso de elaboración de este proyecto ya que esta tecnología de XML nos permitirá realizar el esquema de contenido de una sección normativa correspondiente al Código Técnico de la Edificación.

2.5.4. Otras Tecnologías de XML

XML Schema no es la única tecnología de XML que permite al usuario agilizar las tareas de marcado de textos. Hay muchas otras tecnologías de XML que permiten otro tipo de aplicaciones de gran utilidad tales como:

a) XLink
Define la forma estándar de añadir enlaces dentro de un documento XML. Mediante esta especificación se podrá:

- Utilizar cualquier elemento del XML como origen de enlace.
- Enlazar a más de un documento.
- Crear enlaces fuera del documento.
- Definir el comportamiento del enlace, etc.

b) XPointer:
En esta especificación se define la sintaxis que nos permitirá enlazar al interior de un documento XML. Con esta nueva norma los enlaces ya no sólo serán a elementos ya marcados sino que se permitirán enlaces del tipo:

- Enlaza a la primera aparición de la palabra Internet'.

- Enlaza al tercer párrafo del segundo capítulo', etc.

c) XFragments:
Que define como poder hacer referencias a partes dentro del documento XML. Es como las URL, pero haciendo referencia a partes dentro del documento XML.

d) XSL (eXtensible StyleSheet Language)
Define un estándar para las hojas de estilo de XML. Es la ampliación y modificación de las CSS. XSL está basado en XSLT.

e) Xquery
Es un lenguaje de búsqueda diseñado para consultar colecciones de datos XML.

De entre todas estas tecnologías expuestas y la anteriormente comentada XML Schema, se ha encontrado también especial interés en el uso de XQuery, ya que será el motor de búsqueda para la aplicación de consulta normativa que se desarrolla en este proyecto. Con ella una vez creado el mapa del documento XML se podrá consultar la información marcada.

2.5.5. Consulta de Documentos XML. XQUERY.

XQuery proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML, como por ejemplo Bases de Datos Relacionales o documentos ofimáticos.

También incluye la posibilidad de construir nuevos documentos XML a partir de los resultados de la consulta. Se puede usar una sintaxis similar a XML si la estructura (elementos y atributos) es conocida con antelación, o usar expresiones de construcción dinámica de nodos en caso contrario. Todos estos constructores se definen como expresiones dentro del lenguaje, y se pueden anidar arbitrariamente.

El lenguaje se basa en el modelo en árbol de la información contenida en el documento XML, que consiste en siete tipos distintos de nodo: elementos, atributos, nodos de texto, comentarios, instrucciones de procesamiento, espacios de nombres y nodos de documentos.

El sistema de tipos usado por el lenguaje considera todos los valores como secuencias, asumiéndose un valor simple como una secuencia de un solo elemento. Los elementos de una secuencia pueden ser valores atómicos o nodos. Los valores atómicos pueden ser números enteros, cadenas de texto, valores booleanos, etc. La lista completa de los tipos disponibles está basada en las primitivas definidas en XML Schema.

2.5.5.1. Estructura XQuery

En XQuery las consultas pueden estar compuestas por cláusulas de hasta cinco tipos distintos. Las consultas siguen la norma FLWOR (leído como flower), siendo FLWOR las siglas de For, Let, Where, Order y Return. A continuación, en la tabla 1, se describe la función de cada bloque:

a) For: Vincula una o más variables a expresiones escritas en XPath, creando un flujo de tuplas en el que cada tupla (secuencia ordenada de objetos) está vinculada a una de las variable.

b) Let: Vincula una variable al resultado completo de una expresión añadiendo esos vínculos a las tuplas generadas por una cláusula for o, si no existe ninguna cláusula for, creando una única tupla que contenga esos vínculos.

c) Where: Filtra las tuplas eliminando todos los valores que no cumplan las condiciones dadas.

d) Order: by Ordena las tuplas según el criterio dado.

e) Return: Construye el resultado de la consulta para una tupla dada, después de haber sido filtrada por la cláusula where y ordenada por la cláusula order by.

2.5.5.2. Usos XQuery

Una relación de algunos ejemplos de uso de XQuery:

a) Extraer información de una base de datos para usarla en un Servicio Web

b) Generar un resumen de la información almacenada en una base de datos XML

c) Realizar búsquedas textuales en la web y compilar los resultados de la misma

d) Seleccionar y transformar datos de XML a XHTML de forma que se puedan publicar en la Web

e) Obtener datos desde diferentes fuentes con vistas a ser integradas por la aplicación

f) Dividir un documento XML que representa una serie de múltiples transacciones en varios documentos XML, uno por cada transacción

2.6. Conclusiones

Se ha conseguido establecer, mediante el estudio de las ontologías, una base solida para crear los modelos de aplicación al presente proyecto. De esta manera tras los diagramas de modelado UML se podrá establecer criterios para crear una semántica ontológica.

Una vez creada la semántica ontológica, y como punto de partida para la anotación de los textos normativos que contienen información ontológica, se pretende conseguir que esta anotación permita un acceso a la información de manera inteligente mediante las herramientas y tecnologías de consulta (Xquery), que faciliten la búsqueda y consulta de las normas.

Pero para crear y modelar todas las ontologías presentes en un documento normativo y posteriormente aplicarle el proceso de marcado semántico basado en las ontologías, es preciso primero estudiar la estructura de los documentos normativos, así como los conceptos que en ellas se regulan. Esta tarea será la que se aborde en el próximo capítulo de este documento.