UNIVERSIDAD DE VALLADOLID

DEPARTAMENTO DE INFORMATICA

TESIS DOCTORAL

PRINCIPIOS PARA LA EXPLOTACIÓN DINÁMICA DE RELACIONES ENTRE DOCUMENTOS EN LAS BIBLIOTECAS DIGITALES: APLICACIÓN AL ENTORNO JURÍDICO

María Mercedes Martínez González

Valladolid 2001

No sé si la sabiduría está en la experiencia o en la activa esperanza; donde no está es en la resignación. Si alguien os demostrara que sí, renunciad a tal sabiduría.

ANTONIO GALA. *Progresar. Carta a los herederos.* 1995

Agradecimientos

Quiero expresar mi agradecimiento a las personas que de algún modo me han ayudado a sacar adelante y mejorar este trabajo. Espero no olvidar a nadie, pero si así fuera, mis disculpas. A mis directores, Pablo de la Fuente y Jean-Claude Derniame, les doy las gracias por la confianza y apoyo que me han prestado durante este tiempo. También quiero agradecer a Jacques Ducloy haberse interesado desde el comienzo por esta tesis, así como su constante apoyo. A Samuel Cruz-Lara y Jean-Charles Lamirel, mi gratitud por sus desvelos y las cuidadosas lecturas y correcciones de este documento.

A los doctores Nieves Brisaboa, Hilario Canós, Arlindo Oliveira, Jean-Charles Lamirel y César Llamas, mi agradecimiento por haberme hecho el honor de formar parte de mi tribunal e interesarse por este trabajo.

A César LLamas y Carlos Enrique Cuesta les doy las gracias por haberme ayudado durante la fase final de puesta a punto de este documento. A Belar y Carlos Enrique, por haber encontrado tiempo para revisar mis resúmenes y capítulos. También a las personas que durante este tiempo han respondido mis preguntas o han compartido conmigo discusiones que han enriquecido mis conocimientos y aclarado mis dudas: Jordi Barrat, Carlos Enrique Cuesta, Yolanda Diebold, Patrice Bonhomme y Arturo González.

A las personas de los servicios administrativos que me han ayudado a superar los obstáculos que en ocasiones suponen las cuestiones administrativas, Rosa Ventoso, Virginie Antoine, Danielle Marchand, Nadine Beurné, y el personal del negociado de Tercer Ciclo de la Universidad de Valladolid, gracias por su tiempo y colaboración.

Han sido una inestimable ayuda las personas que han participado en las traducciones: mi hermana, Harald y Margarita. Gracias por colaborar en una tarea tan poco agradable. A mi corrector de inglés, Alan, gracias por la exquisita profesionalidad, paciencia y buen humor con que ha revisado mi memoria. Del mismo modo, quiero agradecer a Jean-Charles la revisión de mi resumen francés. A mi compañera de docencia en este último curso, Carmen Hernández, le quiero agradecer lo mucho que me ha facilitado esta tarea, permitiéndome así dedicar algo más de tiempo a mi tesis en una etapa en que tanta falta me hacía.

A los ocupantes (o casi) del despacho B127 del LORIA, Gégé, Rim, Jean-Charles, Manu y Garti, les quiero agradecer la cálida acogida que me han proporcinado siempre, dándome además el regalo de su amistad. A los amigos que me han escuchado y animado, Bego, Toñi, Belar, Yolanda, Cristina, Silvio, Margarita, Carlos Enrique, Pilar y Nines quiero decirles aquí lo valioso que esto ha sido para mi.

Por último, mi familia, un elemento imprescindible, que me ha infundido en cada momento la fuerza que necesitaba. A mi familia en Valladolid, que siempre ha tenido su puerta abierta para mi, gracias por vuestro cariño. A mi hermana, que siempre sabe qué decir y cuándo, gracias por infundirme ánimos tantas veces con tu estupendo entusiasmo. A mis padres les podría agradecer muchas cosas, pero simplemente les diré que son el pilar imprescindible sobre el que se asienta esta tesis. De no haber contado con vosotros, no creo que ahora estuviese escribiendo ningún tipo de agradecimiento.

Universidad de Valladolid
Departamento de Informática

Principios para la explotación dinámica de relaciones entre documentos en las bibliotecas digitales: aplicación al entorno jurídico

Memoria de tesis doctoral presentada por

María Mercedes Martínez González

Bajo la dirección de los doctores

Dr. Pablo de la Fuente Redondo

Dr. Jean-Claude Derniame

Febrero de 2001

Índice General

Índice de Algoritmos

Índice de Figuras

Índice de Tablas

1

Introducción

Índice General

1.1 Motivación

Las bibliotecas digitales son sistemas de información complejos y avanzados que complementan las bibliotecas tradicionales, amplían sus recursos y servicios y posibilitan el desarrollo de nuevas soluciones para los problemas planteados por los usuarios. Aportan además dos ventajas muy valoradas en la sociedad de la información: un soporte barato para grandes cantidades de información y la posibilidad de acceso masivo a ésta para amplias comunidades de usuarios, que de otro modo no tendrían capacidad para disponer de ellos. Los recursos de una biblioteca digital no sufren el tan temido "desgaste" que padecen sus equivalentes físicos, con lo cual no hay necesidad de restringir el acceso para preservar su integridad. Además, no se requiere el desplazamiento físico del usuario allí donde se encuentra ubicada la biblioteca; más bien es la biblioteca la que se "desplaza" al punto donde se encuentra el usuario, ofreciéndole sus recursos vía interfaces, tan populares en algunos casos como el de los navegadores Web.

No obstante, estas ventajas llevan pareja una interesante complejidad: las bibliotecas digitales son multidisciplinares, los datos que albergan son heterogéneos en su naturaleza, formatos, tipos, etc. y las aplicaciones en las que se apoyan también suelen ser heterogéneas. Hablar de bibliotecas digitales es hablar de campos de interés tan variados como la búsqueda de información, el filtrado de información, la distribución de recursos y/o aplicaciones, heterogeneidad, manipulación de información y documentos, interacción hombre-máquina, o los aspectos relacionados con la seguridad y derechos legales en el acceso a los recursos que ofrecen. Comparten, pues, algunos de sus problemas y soluciones con la gran base de información disponible en Internet, lo cual contribuye a potenciar aún más el interés y la actividad en torno a ellas.

La funcionalidad de una biblioteca digital se expresa mediante una serie de servicios. Algunos son "clásicos", como el de *búsqueda* -que permite a los usuarios localizar aquellos documentos que contienen uno o más términos clave- y la *recuperación* de documentos. Otros servicios completan éstos, convirtiéndolas así en sistemas avanzados que ofrecen las siguientes facilidades: clasificar los documentos y/o resultados de las consultas por uno o varios criterios designados por el usuario, la manipulación de documentos y consiguiente generación de otros nuevos. También cabe incluir aquí la posibilidad para el usuario de interaccionar con la biblioteca en sesiones durante las cuales se "mejora" el conocimiento del usuario sobre la biblioteca a la vez que el de la biblioteca sobre el usuario.

Entre las posibles cualidades deseables en una biblioteca, en esta tesis reciben atención especial las siguientes:

- que la biblioteca sea capaz de ofertar a través de sus servicios acceso a documentos heterogéneos (en formato, naturaleza, estructura, ...),
- que permita descubrir información adicional sobre los documentos (por ejemplo, relaciones entre ellos),
- que dé la posibilidad al usuario de abstraerse o no (según las circunstancias lo requieran) de la existencia de múltiples versiones del mismo documento.

Normalmente los documentos de una biblioteca están relacionados entre sí y dicha información debe ser accesible, susceptible de ser manipulada automáticamente y aprove-

chable por el usuario y las aplicaciones. Además, la biblioteca debería ser capaz de proporcionar a sus usuarios cualquier versión de un documento. Todo ello sin mermar la funcionalidad básica de cualquier biblioteca, lo cual equivale a decir que los nuevos servicios han de poder *integrarse* en una biblioteca "clásica". Cada uno de estos puntos se desarrolla con más detalle a continuación.

La heterogeneidad en los documentos

Además de la variedad y complejidad de los servicios ofertados a los usuarios, es característica la heterogeneidad de la información. Una biblioteca digital puede contener datos heterogéneos en

- su nivel de estructuración (información estructurada, semiestructurada, no estructurada),
- su naturaleza (texto, vídeo, audio, ...),
- formatos (.doc, .tex, .fm, ...),
- etc.

Centrándose en los documentos, éstos pueden subdividirse a su vez en documentos *estructurados* (compuestos por elementos bien delimitados entre los cuales existe una jerarquía de inclusión) y *no estructurados*. Por ejemplo, una memoria de tesis doctoral suele ser un documento estructurado: está compuesto por *capítulos*, cada uno de ellos contiene a su vez *secciones*, cada una de las cuales puede subdividirse asimismo es *subsecciones*, etc., cuyos límites están claramente definidos. A partir de aquí, los documentos estructurados centran la atención de esta tesis.

Los documentos estructurados pueden ser de distintas clases: el tipo de los elementos (divisiones) que aparecen varía de unos documentos a otros, al igual que la jerarquía de inclusiones permitida entre ellos. Para ilustrar esto se puede considerar un caso sencillo: no son las mismas las divisiones que un lector espera encontrar en una tesis doctoral (mencionadas en este mismo párrafo) que en un artículo científico (normalmente, este último no contendrá *capítulos*). No obstante, la heterogeneidad en los documentos estructurados puede deberse a la propia naturaleza de la información que contienen, pero también a la diversidad en las estructuras que presentan [66]. Algunas estructuras respetan divisiones inherentes a la información que contiene el documento, mientras que otras responden a criterios variables, adoptados por aquellos que manipulan los documentos. Por ejemplo, una referencia bibliográfica puede dividirse en campos que cualquier persona reconocería fácilmente (título, autor, ...), o en otros cuyo significado resulta oscuro. Véase como ejemplo la comparación entre las estructuras obtenidas para la misma referencia bibliográfica utilizando BibTeX en un caso y HTML en otro: mientras que en la referencia BibTeX hay la certeza de que los campos autor, título, editor, etc., son fácilmente reconocibles, en el caso de HTML no hay certeza de que no se den situaciones en que elementos, como por ejemplo autor y editor, no se hayan fundido en un solo campo, para facilitar la presentación en un navegador. Las posibilidades de explotación que ofrece disponer de documentos digitales cuya estructura refleje fielmente la estructura semántica de la entidad abstracta asociada son múltiples.

Algunas serán presentadas cuando se comente el acceso a fragmentos internos, la detección de referencias y la modelización y explotación de estas relaciones. Este aspecto ha sido infravalorado hasta el momento en muchas de las implementaciones de bibliotecas digitales existentes. En estas implementaciones se han utilizado masivamente formatos y estructuras donde se da relevancia a la presentación del documento, por ejemplo, HTML [112]. Estas estructuras, introducidas de modo artificial, satisfacen adecuadamente el objetivo con el que fueron pensadas, pero tienen el inconveniente de ocultar la estructura semántica del documento (ver [81, 80, 78, 74]), que puede llegar incluso a quedar inaccesible [65]. Este problema no se debe tanto a la falta de tecnología que lo soporte (tanto SGML [75], inicialmente, como XML [117] posteriormente, permiten la definición de estructuras y etiquetado a voluntad), como al hecho de que, en lo que a documentos se refiere, la urgencia de facilitar el acceso masivo a ellos ha relegado a segundo plano los aspectos relacionados con su semántica, estructura y manipulación.

Las relaciones entre documentos

Los servicios genéricos que se pueden encontrar en cualquier biblioteca son aquellos que permiten la búsqueda y recuperación de documentos, mientras que los servicios avanzados están definidos en cada caso en función de las necesidades o posibilidades de cada biblioteca particular. La clasificación de los resultados de las consultas por su afinidad con respecto al tema, procedencia, autor u otro criterio son algunos ejemplos. También la "navegación" por las colecciones de la biblioteca es una funcionalidad deseable e incorporada a muchas bibliotecas: el usuario comienza la exploración en algún documento guía en el cual las referencias a documentos se encuentran agrupadas en colecciones, las cuales pueden a su vez subdividirse en más colecciones. Cualquiera de las clasificaciones mencionadas (las colecciones son el resultado de algún proceso de clasificación) es el reflejo de la existencia de relaciones semánticas entre los documentos agrupados bajo la misma categoría: documentos con el mismo tema, documentos creados en la misma fecha, documentos citados por los mismos autores, etc.

En la mayoría de las bibliotecas estas relaciones se expresan mediante enlaces creados manualmente [26] y son francamente escasos aquellos casos en los que se ha conseguido algún tipo de detección automática de relaciones de cualquier tipo. Esto de por sí ya es una limitación. Pero además las posibilidades de extraer la información implícita en las relaciones se ven restringidas por la forma en que están modeladas. Las opciones más utilizadas han sido la ya citada, consistente en modelar estas relaciones como enlaces insertados manualmente en los documento, o como metainformación asociada a ellos [23].

La inserción de enlaces en los documentos da lugar a hipertexto, lo cual implica que la forma de "consultar" las relaciones es la navegación a través del hipertexto. En muy pocos casos se ha planteado (y en menos implementado) algún tipo de consulta que no requiera dicha navegación. Se puede concluir, por tanto, que las relaciones entre documentos es un aspecto poco explotado hasta el momento. No obstante, es en la metainformación (las relaciones son un tipo particular de metainformación) donde se encuentran las posibilidades para avanzar hacia una mejor explotación semántica de las bibliotecas digitales.

La manipulación de documentos

Los documentos presentes en una biblioteca pueden ser reutilizados para crear nuevos documentos. La manipulación de documentos puede consistir en una conversión de formato, de modo que se obtenga una copia con el mismo contenido pero distinto formato; por ejemplo, la generación de páginas HTML para la presentación final en una interfaz de usuario es una conversión de formato. Otro ejemplo es la composición de colecciones, series o revistas por inclusión de artículos. Esta reutilización es en realidad una composición de documentos que se puede expresar como un conjunto de vínculos entre el documento patrón y sus fragmentos [109]. Más interesante aún es la reutilización de fragmentos de documentos en vez de documentos completos. La manipulación de fragmentos requiere la caracterización y acceso a los fragmentos de otros documentos, lo cual es posible si los documentos de trabajo están *estructurados*.

Las versiones de los documentos

Los documentos son susceptibles de ser modificados en el tiempo, lo cual da lugar a nuevas versiones del documento modificado. Estas versiones históricas, que deben su nombre al factor temporal de los cambios, coinciden parcialmente en su contenido, difiriendo en la parte afectada por la modificación. En algunos casos es interesante disponer de (o ser capaces de obtener) todas las versiones de un documento; este es el caso, por ejemplo, de las versiones históricas de los documentos normativos en el ámbito jurídico, que deben ser leídas tal cual se encontraban en el momento en que una sentencia fue dictada para que esta última sea comprensible.

Se da además la circunstancia de que en algunos casos estas modificaciones son redactadas por los autores como un nuevo documento o parte de otro, de forma que "coexisten" la versión original del documento, la versión modificada, y el que contiene la modificación -que es un documento distinto, con entidad propia-. Para cada modificación, el autor cita el fragmento de documento afectado e indica cómo modificar dicho fragmento (eliminándolo, sustituyéndolo, ...).

La existencia de múltiples versiones de una cierta obra implica una relación entre ellas, lo cual ha sido abordado como una cuestión de mantenimiento de la base de documentos y de los enlaces entre las versiones del mismo documento almacenadas [38]. Las dificultades de este enfoque surgen del riesgo de explosión de tamaño de la base de documentos si las modificaciones son frecuentes o los documentos versionados muy grandes, y en el mantenimiento de los enlaces: cada enlace que afecta a un documento es candidato a afectar a todas sus versiones.

La arquitectura de las bibliotecas digitales

La arquitectura de una biblioteca digital viene definida en base a las funcionalidades requeridas en ella. Debe facilitar la distribución de tareas entre los servicios de la biblioteca. Si se tienen en cuenta las funcionalidades básicas de cualquier biblioteca (consulta, recuperación de documentos y navegación[1]), es posible extraer un *modelo de referen-*

[1]La navegación es posterior en el tiempo a los servicios de búsqueda y recuperación de documentos. Pero su expansión en las bibliotecas (en grados de sofistificación variable) y el hecho de que las bibliotecas creadas en los últimos años incorporan algún tipo de navegación permiten considerarla ya

cia[2]. También existen propuestas centradas en servicios orientados al tratamiento de relaciones: detección de relaciones semánticas de tipo cita [67], gestión y mantenimiento de los enlaces de la biblioteca [32, 97], y otros que facilitan a los usuarios el proceso de creación de enlaces [33].

La integración de bases de datos y servicios requiere un protocolo que regule la interacción entre los servicios participantes en cada operación. La incorporación de nuevos servicios supone una modificación sobre el sistema: su interacción con los servicios existentes obliga a enriquecer el protocolo.

Posteriormente, se avanza a la fase de implementación. En esta estapa se evoluciona desde el modelo de referencia obtenido inicialmente hacia una *arquitectura de referencia* (conjunto de componentes software)[3]. En este caso, en prácticamente cualquier biblioteca es posible encontrar junto a las bases de datos (documentos, metadatos, otros) componentes de indexación y búsqueda, así como una interfaz de usuario, que implementan los servicios básicos. En los casos de bibliotecas heterogéneas (y/o distribuidas) aparecen además componentes integradores. Por último, también se debe tener en cuenta que los documentos (o datos en general) que conforman la biblioteca pueden estar a su vez distribuidos en distintos elementos y bases, lo cual obliga a considerar la arquitectura de los datos. La incorporación de un nuevo servicio sobre el modelo de referencia supone la incorporación o sustitución de algún componente o conjunto de componentes en esta arquitectura, cuya cooperación implemente el total de los servicios de la biblioteca: el nuevo servicio y los que existían previamente.

Para concluir, es destacable que los servicios destinados a la manipulación y explotación de relaciones son escasos y se orientan en su mayoría hacia la gestión de los enlaces [33] o la ya citada detección de referencias. La explotación de relaciones y manipulación de documentos se encuentran aún integradas en pocas bibliotecas o protocolos.

1.2 Objetivos

El objetivo de esta tesis es ofrecer propuestas que posibiliten la integración en una biblioteca digital de servicios que permitan explotar las interrelaciones entre documentos, además de acceder a cualquier versión (histórica) de un documento.

Se trata, pues, de permitir a los usuarios de las bibliotecas obtener información sobre las interrelaciones entre documentos. Por otro lado, la información que proporcionan las relaciones también será utilizada como base del proceso de generación de nuevos documentos (concretamente de versiones de documentos). Finalmente, se pretende ofrecer una propuesta arquitectónica tal que sea posible incorporar a una biblioteca digital servicios "orientados a las relaciones", cada uno de los cuales será accesible a través de una serie de interfaces que forman parte del protocolo de interacción entre estos servicios y los demás presentes en la biblioteca.

Las propuestas que se ofrecerán para dar respuesta a estos objetivos se fundamentan

clásica.

[2]Se entiende *modelo de referencia* según la definición de [18]: descomposición estándar de un problema en partes que cooperan para resolverlo.

[3]También se sigue aquí la definición de [18].

en los argumentos que se exponen a continuación. Somos de la opinión de que es factible conseguir una arquitectura basada en un modelo de servicios básicos, de modo que las nuevas propuestas se integren con éstos sin afectar a los servicios existentes. Además, partimos de que son los documentos estructurados los que permiten la reutilización de fragmentos en la composición de nuevos documentos. Consideramos, no obstante, que para obtener realmente servicios avanzados y conseguir documentos semánticamente coherentes en las operaciones de generación de documentos, es imprescindible que la estructura de los documentos digitales almacenados en las bases de la biblioteca refleje fielmente la estructura semántica de la entidad abstracta que la copia digital representa. Sólo teniendo en cuenta la estructura lógica asociada al documento abstracto es factible obtener una copia digital tal que sea posible modelar y explotar las relaciones entre documentos, con un nivel de granularidad máximo, a la vez que crear nuevos documentos donde quedarán reflejadas las modificaciones históricas sufridas. Dado el origen semántico de la estructura necesaria, su obtención se plantea a partir del *contenido* del documento. Se considera que el modo más adecuado de conseguirlo es el más natural: el mismo que permite a un lector crear una representación mental de dicha estructura durante una lectura de inicio a fin de cualquier copia del documento.

No se considera necesario almacenar las copias correspondientes a las distintas versiones de un documento: es posible generarlas, siempre y cuando se hayan representado convenientemente las relaciones de modificación entre documentos. La posibilidad de obtener una representación adecuada para las relaciones entre documentos y sus fragmentos, y la (previa) detección de estas modificaciones se encuentran íntimamente ligadas a la disponibilidad de la estructura del documento. Una representación adecuada de dicha estructura permite establecer el vínculo entre los fragmentos mencionados en los textos y los fragmentos que forman parte de los documentos almacenados. Además, si es posible direccionar dichos fragmentos, se puede mantener la integridad del documento en el que se encuentran, a la vez que reutilizar dichos fragmentos en tantos tratamientos como sea necesario. Consideramos posible generar las versiones deduciendo las reglas de composición del documento virtual que se está creando, en vez de almacenar directamente las reglas de composición. La hipótesis de que partimos es que, dado un grafo que contiene la información sobre la estructura de un documento y las modificaciones que le afectan, es posible plantear un recorrido de este grafo tal que la nueva versión se crea a medida que se recorre dicho grafo.

Con todo ello, se puede desglosar el objetivo planteado al principio de esta sección en la siguiente serie de subobjetivos:

- Mejorar la explotación de relaciones en las bibliotecas digitales, permitiendo consultas sobre ellas (recursos relacionados, naturaleza de la relación, tipos de relaciones en las que participa cada recurso, etc.). Se debe modelar las relaciones entre documentos de modo que se facilite este objetivo.

- Ser capaces de componer automáticamente, de forma dinámica, nuevos documentos, aprovechando la información que proporcionan las relaciones entre los documentos que deben participar en dicho proceso.

- Disponer de copias de los documentos cuya estructura sea tal que se facilite el acceso a los recursos implicados en las relaciones.

- Integrar estas funcionalidades en las bibliotecas digitales, sin alterar el normal funcionamiento de aquellos servicios que ya se encuentran operativos.

Las aportaciones con las que se responde a estos objetivos se comentan brevemente en la sección 1.5, en el momento en que se presenta el contenido de cada capítulo.

1.3 Prototipo de implementación

La explotación de las interrelaciones y el acceso a múltiples versiones de un documento son necesidades especialmente relevantes en ciertos entornos donde los documentos manipulados se caracterizan por una rígida estructura semántica. Este es el caso de los documentos jurídicos, que cumplen todas las características que interesan en esta tesis: están fuertemente estructurados, están altamente relacionados y para un jurista es necesario acceder a la versión de un documento (por ejemplo, una ley) tal como se encontraba redactada en un cierto momento. Las diferencias entre distintas versiones consisten en modificaciones parciales al contenido del documento, de modo que cada modificación da lugar a una nueva versión. Dado su carácter temporal son consideradas modificaciones históricas.

El prototipo donde se implementan las propuestas teóricas de esta tesis es una biblioteca digital de información legislativa. Es un entorno especialmente interesante, con la peculiaridad adicional de que las modificaciones se encuentran expresadas a su vez como parte del contenido de un documento donde se cita el documento afectado, para incluir a continuación la modificación.

1.4 Evolución del trabajo

La evolución temporal de esta tesis comenzó en una etapa de definición de objetivos, influida por las necesidades, en lo que a funcionalidades se refiere, que se plantean en un entorno como el del prototipo. Fue así como surgieron la necesidad de acceder a las versiones de un documento, de explotar las relaciones entre documentos y de disponer de una copia semánticamente estructurada de cualquier documento que se quiera versionar automáticamente. A partir de aquí, el trabajo ha avanzado en una serie de etapas que se corresponden de modo casi unidireccional con la organización en capítulos de esta memoria:

- La etapa siguiente a la definición de objetivos fue la expresión de estos requisitos como un conjunto de servicios. El principal requisito de partida ha sido plantearlos como servicios que puedan *incorporarse* en una biblioteca, enriqueciendo así sus funcionalidades.

- Posteriormente se ha investigado sobre la posibilidad de disponer de algún medio de expresar la estructura semántica de un documento: obtener una copia del documento que refleje dicha estructura y permita acceder a sus fragmentos. El resultado de esta etapa ha sido un algoritmo de obtención de una copia de un

documento cuya estructura lógica refleja las divisiones semánticas del documento abstracto. La entrada a dicho algoritmo es alguna de las copias del documento cuya estructura lógica no refleja las divisiones semánticas. El algoritmo analiza el contenido del documento para detectar las divisiones semánticas.

- Una vez se disponía de los documentos convenientemente estructurados, se planteó la representación de las modificaciones e interrelaciones como enlaces heterogéneos. La información asociada se almacena en una base de enlaces etiquetados, a partir de la cual se puede recomponer el grafo necesario para la generación de nuevos documentos. Un algoritmo de generación de versiones se encarga de recorrer este grafo, apoyándose en el árbol estructural del documento que es versionado y construyendo progresivamente la nueva versión en cada paso de su recorrido.

- Por último, se ha implementado en el prototipo la detección automática de las relaciones, sobre la base de que dichas relaciones son detectables como citas entre documentos y que existen experiencias positivas en la detección de referencias en documentos jurídicos.

1.5 Organización de la tesis

Los capítulos 2 a 4 tratan cada uno de los puntos mencionados en la sección anterior. Cada uno de ellos incluye una revisión del estado del arte en el campo abordado en el capítulo, nuestra propuesta, y una parte final donde se compara dicha solución con otras propuestas anteriores y se analizan los aspectos relevantes de la propuesta presentada.

El capítulo 2 introduce las diferencias y similitudes entre la entidad "documento abstracto" y sus correspondientes copias digitales: aquellas que tienen el mismo contenido, pero distintos formatos, etc. De esta forma, es posible ubicar la propuesta del capítulo: un algoritmo de obtención de copia digital de un documento cuya estructura lógica refleje fielmente la estructura semántica implícita en el documento conceptual. Dado que los documentos estructurados son los únicos en los que vamos a poder explotar las interrelaciones que nos interesan (aquellas que afectan a porciones de documentos y no sólo a documentos completos), nos centramos en dicho aspecto y abordamos un algoritmo que nos permite obtener un documento semánticamente estructurado a partir de un documento cuya estructura no responde a criterios semánticos. Proponemos la utilización de XML como el idóneo entre los estándares que permiten modelar la estructura lógica de un documento y planteamos la implementación del algoritmo conversor sobre una herramienta de manipulación de documentos XML, que simplifique la parte de análisis léxico.

La existencia de múltiples tipos de relaciones entre los documentos da lugar en esta tesis a un grafo etiquetado, en el cual los vértices no son meros documentos sino fragmentos de documentos afectados por la relación. La explotación de este grafo, la obtención de grafos parciales y subgrafos a partir del grafo original, y la consulta de dicho grafo permiten generar nuevos documentos. En el capítulo 3 nos centramos en algunos tipos de relaciones: estructurales, citas y modificaciones. Las relaciones estructurales las hemos obtenido en el capítulo anterior, ya que son las relaciones de jerarquía en el árbol

asociado a la estructura lógica del documento. Junto a esta propuesta para modelar las relaciones de modo que se facilite su consulta, aportamos una explotación original del grafo de relaciones: un algoritmo de generación dinámica de versiones históricas. Este algoritmo se basa en un tratamiento recursivo del grafo (*versioning graph*) obtenido a partir del árbol asociado a un documento estructurado y los enlaces que representan las modificaciones que afectan a fragmentos de dicho árbol. Facilitamos además la implementación de este algoritmo, aprovechando el esfuerzo previamente realizado para posibilitar la consulta de relaciones: se almacena la información sobre este grafo como una base de enlaces, de modo que los recorridos en el grafo se convierten en consultas a la base de enlaces. Aprovechamos en la implementación de este algoritmo los lenguajes que XML aporta para direccionar los subárboles en los documentos (XPath [115]) así como sus aportaciones para el modelado de enlaces (XLink [119], XPointer [116]).

Una breve revisión de los servicios más comunes en las bibliotecas digitales se encuentra en el capítulo 4. Este capítulo incluye también la puesta al día sobre las funcionalidades ofertadas en algunas bibliotecas digitales, su expresión como servicios y su impacto en la arquitectura y protocolos de interacción utilizados. En este capítulo proponemos una arquitectura en la cual se incorporan servicios clásicos en las bibliotecas digitales con los nuevos servicios objeto de este trabajo: explotación de interrelaciones, manipulación de documentos y consulta de interrelaciones. También se integran los servicios de traducción de documentos, que permiten obtener un documento estructurado en base a su estructura semántica. Los servicios propuestos en este capítulo se corresponden con los algoritmos descritos en los capítulos anteriores.

El capítulo 5 se dedica al prototipo. Se presentan las bases de documentos sobre los cuales se han probado las propuestas anteriores. Los documentos elegidos son heterogéneos respecto a su tipo, siendo algunos de estos tipos fuertemente estructurados. Son por ello los candidatos preferentes para la experimentación de los algoritmos presentados en los capítulos 2 y 3. En este capítulo se comenta también la posibilidad de detectar automáticamente algunas de las relaciones utilizadas en el capítulo 3, lo cual daría lugar a un tratamiento totalmente automático de las relaciones consideradas. No obstante, esta detección constituye en sí misma un área de investigación, restringiendo su inclusión en este trabajo a la factibilidad de dicha automatización en ciertos campos con condiciones muy bien definidas, como es el de nuestro prototipo.

Por último, el capítulo 6 resume las conclusiones extraídas en este trabajo de tesis.

2

Los documentos y su semántica en las bibliotecas digitales

Índice General

Los *documentos* son los elementos principales de las bibliotecas digitales. Dado que no existe una definición de este término que se adapte a lo que en informática se entiende como tal, es necesario establecer una diferencia clara entre el documento abstracto (concepto intelectual) al cual se refiere un usuario y los objetos digitales que se almacenan en la biblioteca. La razón de tal necesidad deriva del hecho de que, mientras los usuarios hacen referencia a entidades abstractas, éstas se encuentran implementadas en copias digitales cuya equivalencia con la entidad abstracta no tiene por qué ser uno a uno. En algunos casos para obtener el documento (abstracto) solicitado por el usuario es necesario realizar una *composición* de documentos (digitales)[1]. En estos casos la tarea se ve facilitada enormemente si los documentos con los que se trabaja son *estructurados*. Los documentos estructurados centran la atención de este capítulo, ya que sólo en ellos es posible hacer referencia a los distintos fragmentos lógicos del documento. Estos documentos tienen una *estructura lógica*, que describe el documento como una composición de piezas o fragmentos que albergan otras piezas, las cuales a su vez pueden albergar más piezas. Los documentos con estructuras similares conforman una *clase*, y tanto los documentos como las reglas (gramática) que caracterizan una clase pueden describirse con la ayuda del estándar XML o estándares a él asociados.

Sin embargo, a un mismo documento abstracto pueden corresponder varias estructuras lógicas distintas. Esto se debe a que, en el momento de la creación de las copias digitales correspondientes, los criterios usados por cada autor pueden diferir (en muchos casos se trata de criterios influenciados por la que será la presentación final para los usuarios de los documentos, como ocurre, por ejemplo, en algunas bibliotecas accesibles a través de servidores Web). No obstante, algunos documentos abstractos tienen una fuerte estructuración implícita: su contenido se estructura en divisiones semánticas claramente definidas. Esta estructura (abstracta), que se denominará en adelante "estructura semántica", es la que utilizan habitualmente los usuarios. Surge así la necesidad de disponer de una copia digital del documento cuyas divisiones lógicas se correspondan con las divisiones semánticas, imprescindible para automatizar los procesos de búsqueda y manipulación que tengan en cuenta las divisiones semánticas a que el usuario se refiere en sus consultas. Por ello, la aportación de este capítulo es un algoritmo que permite reconocer esta estructura, a partir de una copia digital del documento cuya estructura lógica no refleja la estructura semántica de la entidad abstracta. El resultado de aplicar este algoritmo será otra copia, cuya estructura lógica refleje lo más fielmente posible su estructura semántica. A tal estructura la denominaremos "estructura lógico-semántica". La disponibilidad de dicha copia facilitará enormemente tareas como la obtención y explotación de enlaces que se presentará en el capítulo 3.

Hay tres posibilidades a considerar en lo que se refiere a copias digitales de las cuales se pueda extraer la estructura semántica. La primera es que la copia esté estructurada (etiquetada), y se conozcan las reglas asociadas a su clase (DTD en el caso de documentos XML). La segunda es que la copia esté estructurada, pero no se conozca la

[1] El término "documento" es ampliamente utilizado en cualquiera de las dos situaciones planteadas (se habla de "documento" para referirse a una obra intelectual completa, o también se observa el uso de "documento" para referirse a "documentos XML", "documentos Web", etc, que no tienen por qué corresponder con una entidad semántica). En esta tesis se mantendrá su uso en ambos sentidos, para adaptarse a la terminología habitual, confiando en que la ambigüedad quede resuelta por el contexto en el cual se emplea el término.

DTD de la clase a la cual pertenece. La tercera posibilidad consiste en trabajar con copias que no tienen ninguna estructura lógica (sin ningún tipo de etiqueta). El caso considerado en la propuesta de este capítulo es el segundo: las copias sobre las que trabajará el algoritmo propuesto están etiquetadas, pero se desconoce su DTD. Los documentos etiquetados son cada vez más frecuentes debido a la expansión de lenguajes como XML (*markup languages*), lo cual abre un amplío abanico de aplicación para este algoritmo. Respecto a la primera posibilidad (DTD de entrada conocida), éste es un problema de transformación entre DTD, distinto al que aquí se plantea, sobre el cual se pueden encontrar referencias en [27]. En cuanto a la última posibilidad (documentos no etiquetados), se comentará en la discusión de este capítulo (sección 2.6) por qué se trata de una simplificación del caso tratado en esta tesis.

2.1 Documentos

2.1.1 Documentos abstractos, copias y versiones

No hay definición exacta del concepto *documento* tal cual se utiliza en informática. Según el *Diccionario de la Real Academia de la Lengua Española* un *documento* es *algo que da testimonio de algo* (algo que da la prueba de algo). Esta definición no es muy conveniente al hablar de una biblioteca digital que tiene documentos, se considera una de las características requeridas en una biblioteca digital (presentada en el capítulo 4): los usuarios de la biblioteca digital desean trabajos intelectuales; es decir, acceden a la biblioteca con el objetivo de manipular documentos que se correspondan con trabajos intelectuales, no ficheros digitales). Sin embargo, la idea de qué es un documento puede ser diferente en varias comunidades. Por ejemplo, un documento puede ser una entidad abstracta para un usuario de la biblioteca, y un fichero o una página Web para el administrador de la biblioteca. Por otra parte, la entidad abstracta del usuario puede corresponder a varias copias en la biblioteca, o a la composición de los fragmentos que provienen de varios documentos de la biblioteca.

Ideas similares, que pueden servir para aclarar este punto, se pueden encontrar en el informe IFLA [72][2] , que diferencia entre *trabajo* (*work*), *copia* (*copy*), y *manifestación* (*manifestation*):

> A *work* is an abstract entity; there is no single material object one can point to as the *work*. We recognise the *work* through individual realizations or *expressions* of the *work*, but the *work* itself exists only in the commonality of content between the various *expressions* of the *work*. When we speak of Homer's *Iliad* as a *work*, our point of reference is not a particular recitation or text of the *work*, but the intellectual creation that lies behind all the various *expressions* of the *work*[3].

[2]Una clasificación europea equivalente proviene de *indecs* [73] (*in*teroperabilidad de *d*atos en los *s*istemas *e*-commerce initiative).

[3]Una *obra* es una entidad abstracta. No existe un único objeto que se corresponda con una *obra*. Se accede a una *obra* a través de sus realizaciones indivuales o *expresiones*, pero la *obra* existe en realidad en el contenido común entre sus distintas *expresiones*. Cuando se habla de la *Ilíada* de Homero como

Consideraciones similares se han tenido en cuenta en diversos contextos de biblioteca digitales. El diseño de la arquitectura de la biblioteca digital NCSTRL [42] (presentada en el capítulo 4) incluye la arquitectura de *datos*, para reflejar el hecho de que de un mismo documento pueden existir varias copias, correspondientes a diversos formatos. Ocultan esta información al usuario durante todo el proceso de búsqueda, y sólo le informan de la existencia de tales copias en el momento de la recuperación del documento, para preguntarle cuál de ellas prefiere. Otros autores [14] diferencian entre *copias* (diversos formatos) e *interpretaciones* (*renditions*), que son versiones de un documento donde el contenido ha sufrido modificaciones.

En esta tesis, dado que se hablará de *documentos*, se adaptan las definiciones que se consideran fundamentales para entender el resto de este manuscrito. Un *documento abstracto* es la entidad abstracta concebida por un autor o usuario (corresponde al *work*). De cada documento puede haber varias *copias* correspondientes a diferentes formatos (.doc, .pdf, plain text, .xml, etc.); todas las copias de *documento* comparten el mismo contenido. La aplicación de *modificaciones* al contenido de un documento da lugar a *versiones* (que en la ontología IFLA se denominan *expresions*).

Ejemplo 1. Los documentos legales, como los textos normativos, son entidades de las cuales existen múltiples copias. El aspecto de tales copias es irrelevante, puesto que la información que los usuarios buscan es aquella que se encuentra en el contenido, que todas ellas comparten. □

Ejemplo 2. El concepto de *versión* es habitual cuando se habla de canciones, obras de teatro, manuales técnicos, etc. Todos ellos son documentos que suelen sufrir modificaciones que dan lugar a distintas versiones. □

2.1.2 El Identificador Lógico de Documento

En la sección anterior, se explicó que puede haber varias *copias* y *versiones* de un documento dado. La pregunta ahora es "*¿A qué se refiere el usuario cuando accede a una biblioteca digital con la intención de recuperar un documento: al documento abstracto, a la copia, o a la versión?. ¿Cómo se gestionan las versiones en una biblioteca digital? ¿Son las propiedades de una copia o versión diferentes de las propiedades de un documento abstracto? Y, si no es así, ¿cómo las diferencia el sistema?*".

Se puede distinguir un primer conjunto de propiedades que son propias del objeto intelectual y, por tanto, comunes a todas sus versiones o copias. Estas propiedades son *metadatos* que proporcionan información adicional acerca del objeto o documento. Por ejemplo, el *autor* de una novela es una valiosa información sobre dicha obra.

Cuando un usuario habla de un documento, lo primero que se identifica habitualmente es el objeto intelectual (*La Biblia*, la *Ilíada*, la *LRU*, ...). Con esta descripción normalmente es suficiente para identificarlo. Sólo en algunos casos es necesario puntualizar la *versión* (la versión de la *LRU* en el momento actual) o la *copia* (el fichero *postscript*, la copia *ascii*, ...).

Para facilitar la manipulación de esta metainformación y los documentos, así como

una *obra*, aquello que se referencia no es una implementación específica de dicha *obra*, sino la creación intelectual que representan todas y cada una de sus *expresiones*.

la persistencia de los identificadores de documentos (que facilitarán el mantenimiento de la biblioteca digital), se pueden pedir algunas propiedades a estos últimos. El primer requisito es que haya un identificador único asociado a un documento (con el cual se pueda acceder a todas sus copias). Las ventajas de tal política son numerosas: identificadores más comprensibles para el usuario, independientes de la ubicación física de las copias, etc. (una explicación detallada que completa esta enumeración se puede encontrar en [31]). Las características de tales identificadores incluyen *unicidad* y *resolución múltiple*. La unicidad garantiza que un identificador no se pueda asociar a dos documentos distintos en el espacio de nombres donde se define. Si los espacios de nombres están bien gestionados, la unicidad será preservada en universos más grandes, añadiendo el identificador del espacio de nombres al del documento. La *resolución múltiple* garantiza el acceso a todas las versiones/copias de un documento a partir de su identificador.

La cuestión de los identificadores provoca un gran interés en el entorno que se ocupa de enlazar referencias (*Reference linking*)[4] [31]. Existen varios estándares que proponen cuáles deben ser las propiedades de estos identificadores, así como sus reglas de formación (*Digital Object Identifier* [94], *Serial Item and Contribution Identifier* [87], *International Standard Serial Number*, etc.).

2.2 Documentos estructurados

2.2.1 Estructuras asociadas a un documento

En cualquier documento o copia de documento se pueden reconocer dos tipos de estructura [7, 5, 59, 66, 30, 106]:

- La estructura *lógica*, correspondiente a la jerarquía de inclusión entre sus partes lógicas.
- La estructura *física*, que divide la representación visible en áreas físicas, como por ejemplo páginas.

La estructura lógica representa la división del documento en objetos abstractos y la jerarquía de inclusiones entre éstos; relaciones como las correspondientes a referencias cruzadas en los textos *no* forman parte de la estructura lógica. A cada una de las divisiones se le asocia un *tipo* [58]. Por ejemplo, un artículo o libro puede tener divisiones de tipo *capítulo*, *sección*, etc. La estructura lógica de un documento se puede representar en un árbol, donde los nodos representan las divisiones del documento y los ejes las relaciones de inclusión entre ellas. A los documentos que tienen una estructura lógica asociada se les denomina *documentos estructurados*, para diferenciarlos de aquellos donde no existe tal estructura.

Ejemplo 3. La figura 2.1 muestra la estructura lógica de un artículo científico. Este artículo concreto está compuesto por un título (*title*), la información acerca del autor

[4] *Reference linking* es el término general utilizado para referirse a todos aquellos trabajos que investigan los modos de enlazar los objetos que contienen referencias entre ellos. La mayor parte se ocupan de las relaciones derivadas de las referencias (llamadas también *citation*) en revistas, razón por la cual se le denomina también *Citation linking*.

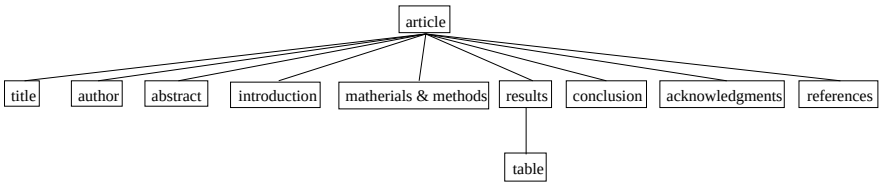


Figura 2.1: Estructura lógica de un artículo científico. No se representa los niveles más bajos de la jerarquía (párrafos y contenido) para preservar la claridad de la figura. (El texto de las figuras se encuentra en inglés, para que coincidan exactamente con las que se encuentran en la versión inglesa de la memoria de tesis)

(*author*), un resumen (*abstract*) de su contenido, una introducción (*introduction*), la presentación de los materiales y métodos utilizados en la experimentación (*material & methods*), los resultados (*results*) obtenidos , una discusión (*discussion*) sobre ellos, los agradecimientos (*acknowledgements*) y, por último, las referencias (*references*) utilizadas. A su vez, los resultados (*Results*) incluyen una tabla (*table*) descriptiva. □

Las estructuras lógicas pueden ser *orientadas al formato* (o aspecto físico) u *orientadas al contenido*. Las estructuras *orientadas al formato* son aquellas en las cuales la división en partes lógicas se ha realizado pensando en el aspecto que finalmente tendrá el documento cuando el usuario lo visualice. Un buen ejemplo de este caso son los documentos HTML, cuya estructura lógica obedece al aspecto que se desea tenga el documento en el momento de su visualización en un navegador. Las estructuras *orientadas al contenido* requieren de tratamientos lingüísticos para su detección, dado que en muchos casos esta estructura se puede reconocer a partir del contenido [106]. Las estructuras orientadas al formato dificultan las búsquedas automáticas basadas en estructura, ya que los usuarios se suelen expresar en términos de la semántica de los documentos y no de su formato.

Ejemplo 4. El artículo del ejemplo 3 podría encontrarse en una página HTML, cuyas divisiones fuesen de los tipos `h4, h3, p`, u otros definidos en este estándar. La estructura lógica, sería en este caso una estructura orientada al formato. □

Las ventajas de disponer de la estructura de un documento son varias. Se puede acceder a los elementos (divisiones) del documento, e incluso reutilizarlos para componer nuevos documentos [14, 96]. Los enlaces son más útiles cuando discriminan los elementos participantes [121, 109, 53], permitiendo, entre otros, mejorar las posibilidades de navegación por los documentos [40]. También las consultas pueden mejorar aumentando su precisión, ya que se puede especificar cuál es el elemento en el que debe aparecer un término para que realmente interese al usuario [1]. El aprovechamiento que se obtiene en esta tesis de la estructura en la expresión de relaciones mediante enlaces y para la consulta de éstos mismos se mostrará en el capítulo 3.

Multiplicidad de estructuras

La estructura lógica divide un documento en partes distinguibles ("tienen sentido") para su autor. Dado que la estructura lógica de un documento depende de su creador, puede haber varias estructuras lógicas asociadas a un mismo documento abstracto.

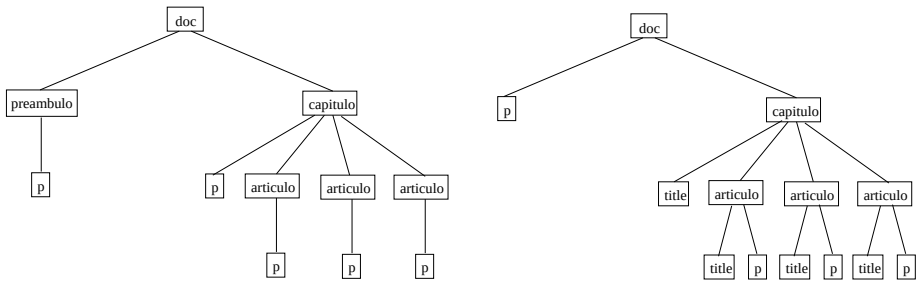


Figura 2.2: Dos estructuras lógicas diferentes para el mismo documento. El contenido (que no aparece en la figura) no varía, aunque sí la organización en divisiones internas.

Ejemplo 5. Los dos árboles de la figura 2.2 son dos estructuras diferentes para el mismo texto normativo español. Son muy parecidas, aunque tienen algunas diferencias. La estructura de la izquierda divide el documento en dos elementos: un `preambulo` y un `capitulo`. En el árbol de la derecha, las dos divisiones principales son un elemento de tipo `p` y un `capitulo`. Es interesante observar la composición de los elementos `articulo` en cada caso. Mientras que en la opción de la izquierda todo el texto del elemento `articulo` se encuentra en el interior de un elemento `p`, en la derecha se distinguen dos subelementos en el interior de cada `articulo`: un `title`, y un `p`. Sin embargo, el contenido es el mismo en ambos casos. Es decir, se trata de dos copias del mismo documento, con estructuras diferentes. □

Un motivo habitual de multiplicidad de estructuras es la creación de copias creadas pensando en el momento de su presentación al usuario (orientadas a la presentación). Estas estructuras son pobres desde un punto de vista semántico y además, como ya se ha comentado, dificultan algunos tratamientos automáticos que aprovechan la semántica del documento.

2.2.2 Reconocimiento de la estructura lógica en documentos

El reconocimiento de la estructura lógica de los documentos se ha orientado en la mayor parte de los casos como un reconocimiento de la gramática común a un conjunto de documentos (representada en la DTD[5] si se trata de documentos SGML o XML) [70, 106, 38, 123, 107, 19]. En otros casos se ha intentado descubrir la jerarquía existente entre los elementos, que eran ya conocidos (HTML) [83], o se ha planteado el reconocimiento de conceptos en los documentos [102], lo cual podría dar lugar a una división distinta a la de la copia original. El problema del que se ocupa esta tesis (reconocer la estructura lógico-semántica de un documento) pertenece a esta segunda categoría, ya que las reglas gramaticales de la clase de salida son conocidas. La comparación de la propuesta que se presentará en la sección 2.4 con las aproximaciones que se acaban de comentar se discutirá en la sección 2.6.

[5] *Document Type Definition (Definición de Tipo de Documento).*

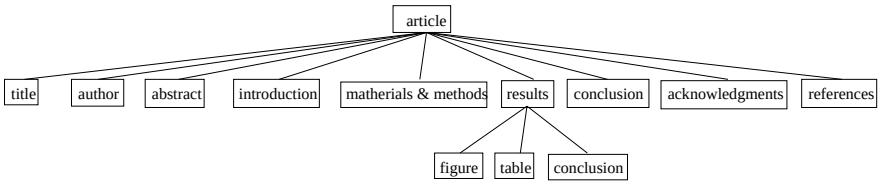


Figura 2.3: Estructura lógica general para la clase "artículo científico". El nivel inferior (párrafos) no está representado. El ejemplo de la figura 2.1 es una instancia de esta clase.

2.2.3 Clases de documentos

¿Qué es una clase?

Los documentos con estructura lógica similar constituyen una clase. Las reglas que definen los elementos permitidos en las instancias de la clase, así como la jerarquía de inclusión posible dan lugar a una estructura lógica general de la clase, la cual se puede describir usando una gramática [69, 37] o mediante una estructura arborescente [59]. La estructura lógica general de una clase describe cómo se construyen documentos de esa clase. Esta estructura consiste en *definiciones* de objetos, mientras que cada documento contiene *instancias* de esos objetos. El árbol asociado proporciona una imagen visual que permite obtener rápidamente una idea sobre cuáles son los elementos permitidos y la jerarquía de inclusión entre ellos.

Ejemplo 6. Los artículos científicos comprenden una serie de partes estándar. Al título (*title*) y la información sobre el autor (*author*), les sigue un resumen (*abstract*). Después aparece una introducción (*introduction*) que describe el estado del arte en el campo, y la presentación de objetivos. Seguidamente se incluye la descripción del material y métodos (*material and methods*) utilizados en los experimentos. El listado de resultados (*results*) debe ir obligatoriamente detrás. La interpretación y comentarios de los resultados se hacen en la conclusión (*conclusion*). Esta última parte es, en realidad, opcional, ya que puede aparecer como un elemento independiente, o dentro de los resultados (*results*). Cerrando el artículo se encuentran los agradecimientos (*Acknowledgements*) y las referencias bibliográficas (*references*).

Esta estructura general (cuyo árbol puede verse en la figura 2.3) es común a cualquier artículo científico, independientemente de cuál sea su tema. Puede que algunos artículos tengan una figura, otros varias, ...; pero todos ellos comparten el conjunto de reglas básicas que se han descrito. Es decir, los artículos científicos conforman una *clase* de documentos, caracterizada por su estructura lógica general. □

Ejemplo 7. Los textos normativos españoles se ajustan a una gramática bien definida que marca cuáles son los tipos de divisiones que se pueden encontrar en cualquiera de ellos, así como las inclusiones permitidas entre ellas. Estas restricciones se pueden expresar con la gramática de la figura 2.4. El árbol de la figura 2.5 también representa estas reglas.

Todos los documentos de esta clase comienzan con un texto, al cual sigue una secuencia de elementos, que pueden ser de cualquiera de los tipos `libro, titulo, capitulo, seccion, articulo`, cuya aparición es opcional. Para cada tipo de elemento se definen

cuáles son los tipos que puede contener. Por ejemplo, un elemento de tipo `libro` puede estar compuesto por varios elementos de tipo `titulo, capitulo, seccion` o `articulo`. En el último nivel de la jerarquía de inclusión se encuentran las divisiones de tipo párrafo (`p` en la figura). Nunca puede ocurrir que un elemento incluya a otro de mayor nivel (por ejemplo, un elemento de tipo `capitulo` nunca puede aparecer en el interior de un `articulo`), lo cual queda perfectamente expresado en la estructura *arborescente* de la figura 2.5. □

$$
\begin{array}{lcl}
< norma > & ::= & < p >^*(< libro > | < titulo > | \\
 & & < capitulo > | < seccion > | < articulo >)^+ \\
 & & < disposicion >^* \\
< libro > & ::= & < title >?(< titulo > | < capitulo > | < seccion > | \\
 & & < articulo >)^+ \\
< titulo > & ::= & < title >?(< capitulo > | < seccion > | < articulo >)^+ \\
< capitulo > & ::= & < title >?(< seccion > | < articulo >)^+ \\
< seccion > & ::= & < title >? < articulo >^+ \\
< articulo > & ::= & < title >? < p >^+ \\
< disposicion > & ::= & < title >? < p >^+
\end{array}
$$

Figura 2.4: Gramática asociada a los textos normativos (se utiliza de modo genérico el denominador *title* para el título de cualquier apartado, para establecer la diferencia con las divisiones de tipo *título* de los textos normativos).

Beneficios de la estructura lógica general de una clase

Los beneficios de disponer de la gramática asociada a una clase son múltiples, pero uno de los más importantes es la posibilidad de disociar en los documentos la semántica de los aspectos de formato, a la vez que asociar al conjunto de la clase las instrucciones que permiten dar formato a cualquier documento de la clase. Los editores pueden beneficiarse del conocimiento de la estructura lógica general en el momento de la presentación de los documentos y para gestionar las inclusiones o eliminaciones de elementos que realizan los usuarios [28, 84, 85]. También se pueden definir las reglas de transformación entre clases diferentes[6] [76, 60]. La composición de documentos [57, 6] se ve facilitada cuando se conoce la estructura lógica general, ya que se pueden definir reglas de composición de aplicación general a la clase.

2.2.4 Estándares para documentos estructurados

La estructura lógica de un documento es una información adicional que se ha almacenado de maneras distintas en períodos distintos. En general, se pueden distinguir dos alternativas: almacenar la estructura independientemente del contenido del documento, o hacerlo junto al contenido. En los primeros trabajos se almacenaba aislada,

[6] En el caso de SGML o XML -que se explican en el apartado 2.2.4- se conoce como transformación de DTD.

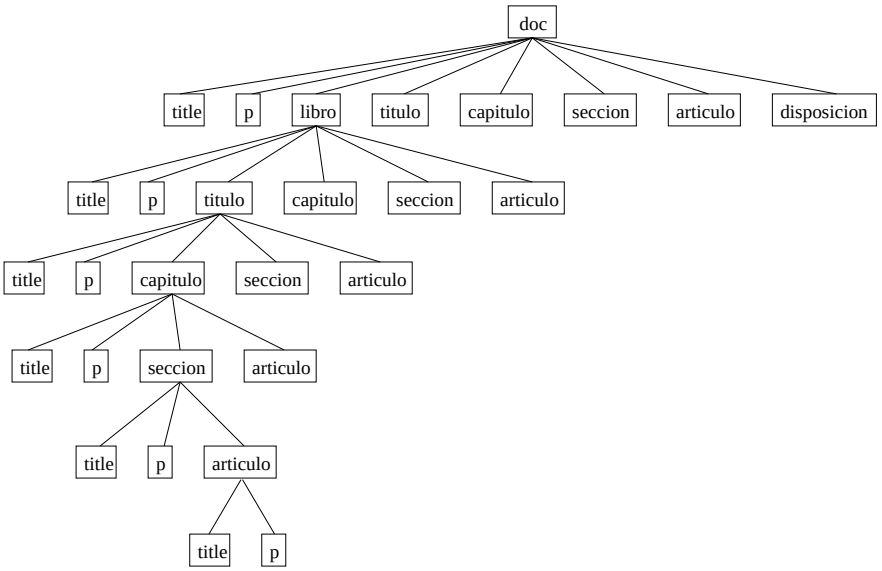


Figura 2.5: Representación parcial (el árbol sólo está completamente expandido en la rama situada más a la izquierda) del árbol asociado a la clase de documentos "texto normativo español". Las relaciones de inclusión entre elementos de la clase se corresponden con las relaciones de ascendencia entre nodos del árbol.

ya que no se disponía de un modo sencillo de hacerlo de otro modo [3, 99]. Sin embargo, la llegada de lenguajes de descripción de documentos basados en la utilización de *etiquetas* introdujo la esperada solución para almacenar conjuntamente estructura y contenido. Los más populares entre estos lenguajes son SGML y XML, que tienen en cuenta además requisitos tan importantes en lo que a manipulación de documentos heterogéneos se refiere como son interoperabilidad en los datos, descripción semántica de la información, o legibilidad.

SGML y XML

SGML *Standard Generalised Markup Language (SGML)* [75] es un estándar para la representación de textos en modo electrónico, independiente del dispositivo o sistema utilizado. En un documento SGML se entremezclan contenido y etiquetado; el etiquetado consiste en etiquetas de apertura y cierre que rodean el contenido de un elemento, delimitando así su principio y final. SGML permite la definición de distintos etiquetados y gramáticas en los documentos, representados por la DTD; es decir, SGML provee las herramientas necesarias para describir la estructura lógica general de una clase. La más popular de estas DTD es HTML (*HyperText Markup Language)* [112], ampliamente utilizada en entornos relacionados con Internet.

XML (*Extensible Markup Language*) [117] es una simplificación de SGML, que comparte sus principios básicos, pero que elimina algunos de los problemas que se habían detectado con los documentos SGML, debidos en su mayoría a la excesiva flexibilidad de sus normas, que ha dado lugar a gran cantidad de documentos cuyo etiquetado es tan pobre e irregular que no es posible crear aplicaciones capaces de analizarlos au-

tomáticamente[7]. La Recomendación XML vigente data de 1998. XML se creó con la intención de que los documentos que se ajusten a sus normas sean legibles, sencillos, analizables por una aplicación automática, y para garantizar la interoperabilidad en los datos. Facilita la construcción de herramientas automáticas definiendo reglas sintácticas más restrictivas que las de SGML. Pero tiene otras cualidades adicionales a las que debe su éxito.

XML es *extensible*: permite definir etiquetas a medida del creador del documento. Esto permite, entre otros, crear documentos etiquetados en base a la semántica de la información que contienen. Resultado de esto es que se puede conseguir etiquetas *descriptivas*: las etiquetas que rodean una porción de contenido no dice cómo presentar dicho texto o qué hacer con él, sino qué es. Esta es una cualidad que se explica perfectamente si se compara con HTML. En HTML el nombre de un elemento está más relacionado con su aspecto en el momento de la presentación al usuario que con su semántica; con XML se puede disociar por completo la presentación del etiquetado del elemento, utilizar las etiquetas semánticamente más convenientes y postergar las cuestiones de presentación para el momento de la visualización, en que se pueden aplicar hojas de estilo que asocian reglas a todos los documentos de una clase, en base a la estructura lógica general de la clase (CSS [110] y XSL [111]). Otras características de XML son que es sencillo, fácil de utilizar y un buen soporte para conseguir interoperabilidad en los datos.

XML no se limita al estándar que define las reglas de descripción de la estructura lógica (y estructura lógica general) de un documento. Existen otros estándares asociados, que sirven para aumentar su potencia descriptiva. XLink [119], XPointer [116] y XPath [115] permiten direccionar elementos en el interior de los documentos y enlazar, no sólo documentos, sino también fragmentos internos. Otras utilidades, como RDF [118], permiten describir metadatos, y también es posible definir espacios de nombres particulares que permitan abordar convenientemente las cuestiones de interoperabilidad semántica [114]. El conjunto de estándares, propuestas y lenguajes asociados a XML se puede consultar en la página que el W3C (*World Wide Web Consortium*) dedica a XML[8].

Aspecto de un documento XML

Los documentos XML están formados por *elementos*, cada uno de los cuales está delimitado por una etiqueta de apertura y la correspondiente etiqueta de cierre. Estos elementos pueden tener *atributos*, que se incluyen dentro de la etiqueta de apertura y se permite la inclusión de elementos dentro de otros elementos. También se permite el uso de *entidades*, que pueden referenciar archivos externos, o consistir en abreviaturas de constantes de cadena.

Además las reglas sintácticas son estrictas, y todo documento XML *bien formado* (que pueda ser procesado por un analizador XML) debe cumplirlas. Estas reglas son:

- *Todas* las etiquetas aparecen en el texto del documento.

[7] En [29] se puede encontrar una comparación detallada de SGML y XML, donde se ahonda en los problemas detectados en la utilización de SGML y cómo resuelve XML dichos problemas.

[8] http://www.w3c.org/xml

- Las etiquetas están correctamente *equilibradas*: toda etiqueta de apertura está cerrada, toda etiqueta de cierre fue abierta previamente y el anidamiento, en caso de existir, es correcto. Una excepción a esta norma son los elementos *vacíos* (*empty elements*), que no requieren etiqueta de cierre.
- Todos los valores de atributos están entre comillas.
- Todas las entidades utilizadas han sido declaradas previamente.

Si además el documento se ajusta a la estructura lógica general de una clase, que se habrá descrito en una DTD, se dice que es *válido*. La DTD sirve para describir la estructura lógica general de una clase y se comenta en el apartado siguiente. No obstante, cualquier documento XML bien formado puede ser procesado por un analizador XML (*parser XML*), lo cual es una diferencia importante respecto a SGML, donde cualquier documento debe ser válido (debe tener una DTD asociada).

La DTD

Tanto SGML como XML permiten asociar una DTD a un documento. La DTD contiene las reglas que determinan cuáles son los elementos permitidos en los documentos de una clase, así como las relaciones de inclusión posibles entre ellos. De este modo, se describe la estructura lógica general de la clase de forma que las aplicaciones automáticas son capaces de manipularla: todos los documentos de una clase tienen asociada la misma DTD. Los atributos de los elementos y las entidades que se utilicen en el documento deben estar definidas en la DTD. La declaración de DTD puede encontrarse al comienzo del documento o residir en un fichero separado, cuya inclusión se especifica en el inicio del documento.

Ejemplo 8. La figura 2.6 muestra un fragmento de la DTD asociada a la clase "texto normativo español". Incluye una declaración de entidad, `libro.mdl`, que se utilizará en la declaración del tipo de elemento `libro`. Si se sustituye el contenido de la entidad en la declaración de elemento, se obtiene que los elementos de tipo `libro` están compuestos por un elemento opcional (`title`), seguido de una secuencia de uno o más elementos de tipo `articulo`, `capitulo`, `seccion` o `titulo`. El único atributo de los elementos de tipo `libro` es el identificador. El atributo se declara como `IMPLIED`, para expresar que es opcional y además no se le asigna ningún valor por defecto. □

```
<!ENTITY  % libro.mdl
        "articulo|capitulo|seccion|titulo">

<!ELEMENT libro       (title?, (%libro.mdl;)+)>
<!ATTLIST libro       id   ID  #IMPLIED>
```

Figura 2.6: Fragmento extraído de la DTD de la clase "texto normativo español".

La DTD más popular en SGML es HTML, que se utiliza para la creación de páginas Web. De este modo ha contribuido a aumentar la popularidad de SGML. Pero al mismo tiempo, la flexibilidad de HTML ha dado lugar a grandes cantidades de páginas mal

estructuradas, de las cuales es difícil (y en ocasiones incluso imposible) extraer de modo automático algún tipo de información semántica. Por ejemplo, es imposible realizar búsquedas en el interior de los resúmenes de documentos cuando éstos se codifican con HTML, ya que no existe ningún tipo de etiqueta para designar dicho concepto semántico. Esto es así porque los documentos HTML se etiquetan en base a su presentación. Actualmente existe una propuesta conforme a las reglas de XML, denominada XHTML [113], en la cual se evitarían los problemas sintácticos que provocan problemas de ejecución a los analizadores; sin embargo, el problema semántico, lógicamente, persiste.

Existen otras DTD de ámbito general cuyo propósito es ofrecer una estructura lógica general aplicable a un amplio rango de documentos. *DocBook* [120] es una DTD SGML, de cuyo mantenimiento se ocupa el comité *DocBook Technical Committee* de *OASIS*. Está orientada a su utilización con libros, artículos o informes informáticos. Otra DTD que surgió al amparo de SGML es la TEI (*The Text Encoding Initiative*) [103]. Esta iniciativa comenzó a trabajar en la definición de una DTD de ámbito general en 1987, cuando XML aún no existía, ni siquiera como propuesta. Por ello, originalmente es una DTD SGML, si bien en la actualidad existe una versión XML. El objetivo de esta DTD es ser suficientemente general para permitir modelar el máximo rango posible de tipos de elementos, cuya semántica se caracteriza en los atributos. Esta DTD ha legado importantes aportaciones a XML, como los *XPointer*, lo cual explica a su vez porqué la actividad en torno a esta DTD ha declinado a medida que XML gana aceptación.

2.3 Documentos en el entorno jurídico

2.3.1 Documentos abstractos, copias y versiones

Los documentos jurídicos existen como entidad abstracta, independientemente del modo en que se almacenen en un soporte informático [3]. Por ejemplo, una *ley* es una entidad con existencia propia, a la cual se refiere cualquier persona, sin prestar atención al modo en que se haya podido digitalizar dicho documento.

Existe otro aspecto relacionado con estos documentos que también es relevante para el trabajo que se realiza en esta tesis: los documentos oficiales (leyes, decretos, ...) suelen sufrir frecuentes modificaciones, que dan lugar a nuevas *versiones* del mismo documento. Estas versiones de un mismo documento, que difieren parcialmente en su contenido (y, posiblemente, en su estructura lógica) pertenecen, no obstante, a la misma clase de documento.

2.3.2 Documentos estructurados

Los documentos jurídicos están fuertemente estructurados: las leyes, o la jurisprudencia se encuentran divididas en elementos muy bien definidos. Esta estructura semántica (existe en la entidad abstracta, independientemente del formato, estructura física o lógica de sus copias digitales) es utilizada en las citas que estos documentos hacen a otros similares y por los especialistas del campo. Estos especialistas se basan en dicha estructura para discernir el texto que realmente les interesa de un documento (algunas

leyes ocupan varias páginas una vez impresas). Es más, se sienten cómodos con estas estructuras y tienden a conservarlas en los documentos que crean.

Así pues, parece que éste es un entorno donde disponer de documentos digitales cuya estructura lógica se corresponda con la estructura semántica de los documentos abstractos puede ser de gran utilidad para facilitar su manipulación [14, 56]. De este modo se podría proporcionar a los especialistas herramientas automáticas que facilitasen su tarea. Entre otros, se podría facilitar la composición de documentos [14] y la creación de hipertexto por el cual se pueda navegar [3].

2.3.3 Estándares para documentos estructurados en el entorno jurídico

Los estándares para documentos estructurados (SGML, XML) han sido utilizados con los documentos jurídicos. Como ya se comentó cuando se presentaron estos estándares, aquí también se ha observado una evolución en el modo de representar la estructura de los documentos. De las propuestas donde se almacenaba separada del contenido [38, 3], se pasó a otras donde la estructura forma parte del texto [56, 12, 65, 55]. Varias DTD han sido definidas expresamente para este tipo de documentos [65, 55, 14]. Algunas de estas DTD están diseñadas en base a criterios de formato [86, 51], cuyos problemas ya se han comentado. Otra opción son las DTD que intentan ser lo más generales posibles [101], de modo que se modele el rango más amplio posible de documentos jurídicos. Este es el objetivo del proyecto *Legis* [65], en cuya DTD se intentaba abarcar los textos jurídicos de varios países europeos. Esta generalidad se consigue a costa de pérdida de especificidad, ya que es en los niveles más bajos del árbol estructural (árbol asociado a la estructura lógica general) donde las distintas normativas europeas difieren. Otra propuesta de estas características es el proyecto *Eulegis* [55], que se encuentra en fase de desarrollo.

Una opción distinta es la adaptación de DTDs generales, incorporando nuevos elementos que asimilen las peculiaridades de la información legislativa [56]. En [56], D. Finke propone *extender* la DTD TEI para su utilización en el entorno jurídico. Las extensiones propuestas consisten en la inclusión de nuevos elementos y atributos. Los nuevos elementos deberían representar las divisiones semánticas. La alternativa a la inclusión de elementos sería modelar el *tipo* de un elemento en atributos `type`. Esta solución, además de modelar la misma información de forma más artificiosa, tiene la desventaja de ser poco apreciada por los expertos jurídicos. Se debe recordar que una de las cualidades de los documentos es que sean "legibles"; para un usuario no informático es mucho más sencillo reconocer las divisiones que busca en los nombres de las etiquetas que en sus atributos que, inevitablemente, les resultan menos comprensibles.

Ejemplo 9. Este ejemplo ilustra lo que acaba de comentarse. En ambos casos la etiqueta corresponde a un elemento semántico de tipo *articulo*. En el primer caso se ha utilizado la DTD TEI. El nombre del elemento (`div1`) ha sido escogido arbitrariamente entre las distintas opciones que ofrece la TEI. Su semántica se ha modelado en el atributo `type`, el cual no es en modo alguno más relevante que cualquier otro atributo, a pesar de ser el atributo que contiene la más importante información semántica. En el segundo caso, se ha definido una DTD específica para los textos normativos. La consecuencia natural es que el *nombre* del elemento, que siempre predomina sobre cualquiera de sus

atributos, es semánticamente expresivo.

```
<div1 id="14/198722" type="articulo">   (1)
<articulo id="14/198722">               (2)
```

□

2.4 Una propuesta para capturar la estructura lógico-semántica

El algoritmo presentado en esta sección genera un documento digital cuya estructura lógica se corresponde con la estructura semántica del documento abstracto del cual es copia. Efectúa la *traducción* de documentos planteada en el capítulo 4. Su entrada es una copia del documento, con una estructura lógica que no se corresponde con la estructura semántica de la entidad abstracta. Reconoce la estructura semántica a partir del contenido (texto) del documento. La copia que genera como resultado está normalizada y, por consiguiente, puede ser utilizada en cualquiera de las tareas del sistema, por cualquiera de sus componentes. De hecho, disponer de dicha copia es crucial para la explotación de relaciones que se comentará con detalle en el capítulo 3. Como se ha indicado en el apartado 2.2.2, se aborda el problema del reconocimiento de la estructura lógica de documentos individuales, con la particularidad de que el proceso se apoya en el *vocabulario* utilizado en el contenido del documento.

El reconocimiento de la estructura semántica se realiza a partir del vocabulario presente en el texto del documento, al igual que un lector cualquiera percibe la estructura semántica del mismo. Los documentos semánticamente bien estructurados incluyen en su texto una serie de términos (o expresiones) que sirven al lector para reconocer el comienzo de cada división. Se obtienen así elementos (divisiones) *semánticos*, que reciben esta denominación debido a que ningún otro tipo de criterio (como su aspecto) participa en modo alguno en su definición. Esta estructura de alto nivel se completa con los elementos situados en los niveles inferiores del árbol asociado a la estructura lógica, divisiones no semánticas, que no pueden ser reconocidas a partir del vocabulario y que se trasladan tal cual se encuentran en la copia original; este es el caso, por ejemplo, de los párrafos. En una representación arborescente del documento resultante, los niveles inferiores (hojas o ascendientes directos) corresponden a elementos no semánticos preservados tal cual se encuentran en la copia de entrada, mientras que los niveles superiores (nodos internos en el camino desde la raíz a un elemento no semántico) corresponden a elementos semánticos generados a partir del contenido del documento; existen así dos franjas en dicho árbol: la franja formada por los niveles superiores que representa la parte semántica de la estructura lógica del documento de salida, y la franja inferior que comprende los elementos de presentación y contenido del documento.

Las posibilidades de aplicación de este algoritmo abarcan aquellos casos en que las copias digitales de los documentos han sido etiquetadas usando criterios que no respetan la semántica del documento. Esta es una situación muy habitual en las bibliotecas que ofrecen acceso Internet a sus documentos; la opción más habitual para modelar estos documentos es HTML. Esta DTD define sus etiquetas, que el usuario *no* puede expandir,

con lo cual los creadores se ven obligados a utilizar estructuras lógicas basadas en el formato de los documentos. Esta elección tiene el inconveniente adicional de que la estructura semántica queda oculta, de modo que no puede ser reconocida por ningún tipo de herramienta automática que se base en la estructura (etiquetado) de los documentos; por ejemplo, es imposible realizar consultas basadas en la estructura que tengan algún "significado" para el usuario (no es informativo para cualquier usuario buscar en la división *h1*, *h2*, ... mientras que sí lo es buscar en una *sección*, *introducción*, etc.).

El reconocimiento de la estructura semántica se apoya en el conocimiento de la DTD de salida (DTD que define la clase a la cual pertenecerá la copia resultante). Asimismo, aquellos aspectos del documento de entrada, comunes al documento abstracto, y que no se pueden reconocer exclusivamente a partir del contenido se preservan en el documento de salida; este es, por ejemplo, el caso de las divisiones en párrafos. La copia generada es un documento XML, con el mismo contenido que la copia de entrada, pero distinta estructura lógica. Si se visualiza el documento de salida como un árbol que refleja las divisiones y su jerarquía de inclusión, se puede afirmar que los nodos internos superiores se generan en base a la semántica, mientras que las hojas (y, posiblemente, ascendientes directos) se generan teniendo en cuenta aspectos de presentación. La implementación del algoritmo se apoya sobre un analizador XML, con lo cual algunas de las restricciones sobre los documentos de entrada, que se comentarán al explicar el algoritmo, vienen impuestas por el estándar XML.

La sección 2.4.3 muestra un ejemplo donde se puede observar cómo el contenido de las copias de entrada y salida al algoritmo es idéntico, mientras que sus estructuras lógicas difieren.

Las características de los documentos manipulados y parámetros del algoritmo se detallan a continuación.

2.4.1 Entradas al algoritmo

Las entradas al algoritmo de traducción son tres: un documento de entrada d (copia del documento abstracto), la información sobre la jerarquía de inclusión en la clase a la cual pertenecerá la copia generada, h, y la información sobre la equivalencia entre el vocabulario utilizado en el texto del documento y los elementos permitidos en la clase de salida, v.

El documento de entrada

El documento de entrada, d, se encuentra etiquetado, pero su estructura lógica no refleja la estructura semántica del documento abstracto asociado. Además, cumple otra serie de requisitos cuya razón de ser se explica en la discusión de este capítulo:

- Todo el texto que no es etiquetado es relevante (se encuentra también en el documento abstracto).
- El texto contiene un conjunto de expresiones o términos (en v) que sirven para reconocer el principio de un elemento durante una lectura secuencial del documento. La presencia de una de estas expresiones marca el comienzo de un elemento semántico.

- El principio de uno de estos elementos (semánticos) marca el final de todos los elementos semánticos situados en la jerarquía en un nivel inferior o igual al del elemento que comienza. Por ejemplo, el comienzo de un *capítulo* en un libro implica que todos los capítulos anteriores están terminados.
- El etiquetado del documento de entrada puede ser de cualquier tipo. La única condición es que esté correctamente equilibrado (todas las etiquetas de apertura están convenientemente cerradas, y viceversa). Esta condición sirve para garantizar que el etiquetado de formato se conserva correctamente en la copia de salida (no hay otro modo de reconocer los límites de estos elementos).
- El anidamiento entre elementos de la entrada se encuentra limitado a los elementos no semánticos; no hay elementos semánticos anidados dentro de elementos no semánticos.

El vocabulario

El vocabulario, v, es la equivalencia entre las expresiones textuales utilizadas en la copia de entrada para marcar el comienzo de un elemento semántico y los tipos de elementos permitidos en la DTD de salida. Se define para cada clase de documentos.

La jerarquía de inclusión en la salida

La jerarquía, h, es parte de la información presente en la DTD de salida; es tal que un elemento no puede contener elementos de nivel superior o igual al suyo en la jerarquía. Esta jerarquía se utiliza en la creación de elementos semánticos.

2.4.2 El documento de salida

El documento de salida, *str*, es una copia del documento abstracto que cumple:

- El contenido es exactamente el mismo que en la copia de entrada (d) al proceso de traducción.
- Es un documento XML bien formado.
- Su estructura lógica se ajusta a las reglas expresadas en la jerarquía utilizada como guía durante la traducción.
- Su estructura lógica cumple las reglas expresadas en h y v.
- Los nodos internos (en el árbol) son elementos semánticos, mientras que las hojas se corresponden con divisiones de presentación.

2.4.3 Un ejemplo

Documento de entrada

En este ejemplo el documento de entrada (en la figura 2.7) es una versión breve y simplificada de un texto normativo español.

```
<doc>
<p>Ley 1.</p>
<h4><a>CAPÍTULO I.</a> DEL REFERENDUM.</h4>
<p><a>Artículo Primero.</a></p>
<p> Texto del artículo primero.</p>
<p><a>Artículo Segundo.</a></p>
<p>Texto del artículo segundo.</p>
<p><b>Artículo Tercero.</b></p>
<p>Texto del artículo tercero.</p>
</doc>
```

Figura 2.7: Documento de entrada al algoritmo de extracción de la estructura semántica.

Una lectura del texto -ignorando sus etiquetas- muestra que el documento contiene un (`Capítulo`), el cual se divide a su vez en tres divisiones de tipo (`Artículo`). Un segundo examen, teniendo en cuenta el etiquetado, completa esta información:

- Hay un párrafo (`Ley 1.`), que no está incluido dentro de ninguna otra división.
- El capítulo contiene una división de tipo *título*: `CAPÍTULO I. DEL REFERENDUM.`
- En el interior del primer `Artículo` se encuentra un párrafo: `Texto del artículo primero.`
- El texto del segundo `Artículo` se expande hasta el comienzo del primer `Artículo`.
- El texto del tercer `Artículo` alcanza el final del documento.

Documento de salida

El documento de salida refleja en su estructura la división semántica mencionada. El texto se puede ver en la figura 2.8 y el árbol asociado se muestra en la figura 2.9. Como se observa, el etiquetado refleja las divisiones semánticas reconocidas en la copia de entrada, cosa que no ocurría en ésta. Hay tres elementos `articulo` en el interior de un elemento de tipo `capitulo`. Todos ellos son nodos internos en el árbol documental. Los títulos (`title`) de los elementos se han reconocido con la ayuda del etiquetado de la copia de entrada. Los elementos de presentación (`p` en la figura) se mantienen exactamente igual que en la entrada, y se corresponden con las hojas del árbol.

La jerarquía

Las reglas jerárquicas son las correspondientes a la clase "texto normativo español"(figura 2.10) y el fragmento correspondiente a la equivalencia entre vocabulario en el texto y elementos en la DTD utilizado en este ejemplo se puede ver en la tabla 2.1. El árbol representa las inclusiones permitidas entre elementos semánticos en los documentos pertenecientes a la clase cuya gramática se describe en el apartado 2.2.3. El árbol está completamente expandido en la rama situada más a la izquierda, para facilitar la visualización. El subárbol compuesto por un elemento y sus descendientes es siempre igual, y debería aparecer cada vez que ese tipo de elemento ocurre en el

```
<doc>
<p>Ley 1.</p>
<capitulo><title>CAPÍTULO I. DEL REFERENDUM.</title>
<articulo><title>Artículo Primero.</title>
<p> Texto del artículo primero.</p>
</articulo>
<articulo><title>Artículo Segundo.</title>
<p>Texto del artículo segundo.</p>
</articulo>
<articulo><title>Artículo Tercero.</title>
<p>Texto del artículo tercero.</p>
</articulo>
</capitulo>
</doc>
```

Figura 2.8: Documento obtenido al aplicar la extracción de estructura lógica al documento de la figura 2.7.

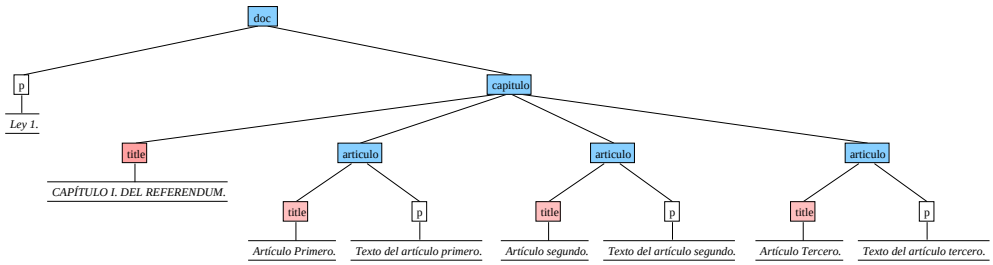


Figura 2.9: Arbol estructural obtenido al aplicar la extracción de estructura lógica al documento de la figura 2.7. Los nodos internos son elementos semánticos. Las hojas coloreadas se obtienen utilizando el contenido junto al etiquetado de la entrada; las hojas no coloreadas corresponden a elementos de presentación.

árbol. Por ejemplo, un elemento de tipo `capitulo` puede tener como descendientes elementos de tipo `seccion` o `articulo`. Sin embargo, ningún elemento de tipo `seccion` o `articulo` puede contener uno de tipo `capitulo`.

Cada tipo de elemento tiene asociado un *nivel* en la jerarquía, que es el nivel más bajo del árbol en el que puede encontrarse dicho tipo durante un recorrido desde la raíz a las hojas (profundidad). Los elementos de tipo `capitulo` son de nivel *3*, los de tipo `seccion` de nivel *4*, y los de tipo `articulo` de nivel *5*; los elementos de nivel *3* pueden contener elementos de nivel *4* o superior, pero la inclusión en sentido inverso está prohibida.

Durante la generación del documento de la figura 2.8, se consulta la jerarquía antes de abrir un elemento semántico, con el fin de cerrar correctamente los elementos de nivel igual o mayor al que se abrirá inmediatamente. En el ejemplo, el comienzo de un `capitulo` supone que todos los elementos de tipo `articulo`, `seccion` y `capitulo` abiertos hasta aquí deben estar cerrados antes de abrir el nuevo `capitulo`.

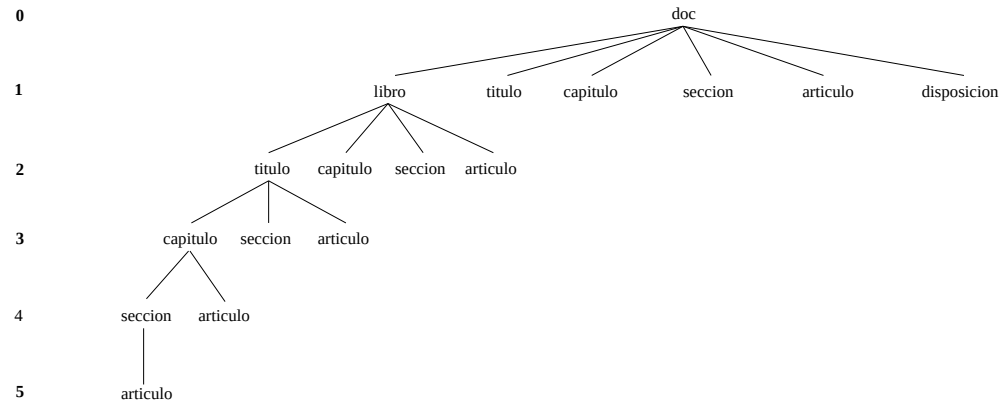


Figura 2.10: Jerarquía de inclusión entre elementos semánticos de la clase "texto normativo español". La representación es parcial (el árbol sólo está completamente expandido en su rama izquierda).

El vocabulario

El fragmento de la equivalencia de vocabularios utilizado en este ejemplo se muestra en la figura 2.1. Los elementos semánticos que se han creado son `capitulo` y `articulo`. Un elemento `capitulo` comienza cuando se encuentra la cadena *Capítulo* en el documento de entrada, mientras que los elementos `articulo` son reconocibles por la aparición de la cadena *Artículo*.

Text vocabulary	*DTD element*
`Capítulo`	capitulo
`Artículo`	articulo

Tabla 2.1: Fragmento extraído de la equivalencia de vocabularios utilizado en la conversión de textos normativos españoles, empleado en el ejemplo de este apartado al trabajar sobre el documento de la figura 2.7.

2.4.4 El algoritmo de reconocimiento de estructura lógica

El algoritmo de conversión simula en su funcionamiento el modo en que una persona reconoce la estructura semántica (divisiones) de un documento durante una lectura secuencial del mismo. El conocimiento implícito que el lector posee acerca la estructura lógica general de la clase de documentos a la cual pertenece aquél que está leyendo le permite reconocer las divisiones de éste último. Parte del vocabulario que aparece en el texto indica el comienzo de una de estas divisiones (y, por tanto, el final de la anterior). Es así como el lector crea una representación mental de la estructura lógica del documento que examina (cuántos capítulos tiene, cuántas secciones hay en cada capítulo, ...).

Hay dos fuentes de información relacionadas con la estructura lógica general de la clase a la cual pertenecerá la copia que se generará en la salida del algoritmo, y que serán utilizadas durante su aplicación:

1. La equivalencia entre el vocabulario utilizado en el contenido (texto) del documento y los tipos de elementos permitidos en la clase de salida (equivalencia entre la ontología propia del campo del cual provienen los documentos y la que el sistema usa internamente para garantizar la interoperabilidad). Se la denomina v.

2. La jerarquía de inclusiones entre elementos semánticos de la clase, h.

Definición 2.1 *El algoritmo de extracción es una función* $\delta : D \times V \times H \longmapsto STR$, *donde*

- D *es un conjunto de documentos XML bien formados*
- V *es un conjunto de equivalencias entre vocabularios*
- H *es un conjunto de jerarquías de inclusión*
- STR *es un conjunto de documentos XML semánticamente etiquetados.*

En cada aplicación del algoritmo, $\delta(d, v, h) = str$, *se tiene:*

- d*: un documento de* D, *en el cual se pueden encontrar términos del vocabulario* v
- v*: la equivalencia entre el vocabulario que aparece en* d *y los tipos de elementos aceptados en la clase a la cual pertenecerá el documento de salida*
- h*: la jerarquía de inclusiones para dicha clase*
- str*: el documento de salida, etiquetado de acuerdo con* d, v *y* h.

Proposición 2.1 *En cada aplicación del algoritmo,* d, v *y* h *se corresponden en el sentido siguiente: el vocabulario que aparece en* d *y* v *coinciden, y los tipos de elementos de* v *y* h *son los mismos.*

Una incongruencia entre d y v daría lugar a una transferencia sin ningún cambio del documento de entrada a la salida. Una incongruencia entre v y h provocaría la finalización del algoritmo en un estado anormal.

Creación del árbol de salida

Tipos de nodos

El documento de salida está bien estructurado y se puede representar su estructura en un árbol, tal como se explicó en la sección 2.2.1.

Se pueden distinguir dos categorías de nodos en este árbol:

- *Nodos (elementos) semánticos.* Nodos creados a partir del contenido. Son nodos internos del árbol.
- *Nodos de formato.* Se crean a partir del etiquetado de la entrada. Son hojas del árbol.
  En caso de que el documento fuente tenga etiquetado redundante, se puede afirmar que estos nodos son bien hojas, o se encuentran en los niveles inferiores del árbol, verificando que en ningún caso puede ocurrir que uno de estos nodos sea

ascendiente de algún nodo semántico. En un recorrido desde la raíz a las hojas, los niveles inferiores corresponden a nodos que provienen de la estructura lógica del documento de entrada y los niveles superiores corresponden a nodos generados a partir de su contenido. De aquí en adelante, se adoptará una postura optimista en este tema, y se hablará de "hojas" para referirse a los nodos de formato[9].

Creación de nodos
La generación de nodos *internos* se hace a partir del vocabulario del documento de entrada. La generación de las *hojas* se realiza según los criterios que se explican:

- *Hojas generadas por combinación de contenido y etiquetado.* En su mayoría se trata de títulos de elementos. Cada porción de texto que se encuentre en una de estas hojas, aparece en el documento de entrada rodeada por un número indeterminado de etiquetas de apertura y cierre. Las etiquetas de apertura le preceden, formando una secuencia consecutiva. Sin embargo, las etiquetas de cierre pueden encontrarse entremezcladas con el texto, de modo que la última de ellas es la que realmente marca el final del elemento.

- *Hojas importadas del documento origen.* En estos nodos el texto no contiene ningún tipo de término o expresión que sirva de guía para reconocer una división. Se asume entonces que el etiquetado que les rodea fue puesto ahí siguiendo criterios puramente de *presentación*, que puede ser interesante conservar. Por ello, se transfieren a la salida tal cual se han encontrado.

Ejemplo 10. En la figura 2.9se puede ver el árbol asociado al documento de salida obtenido en la aplicación del algoritmo al ejemplo del apartado 2.4.3. Los nodos internos son elementos semánticos. Las hojas coloreadas se obtienen a partir de la combinación de contenido y etiquetado y las hojas no coloreadas corresponden a nodos de formato. □

Creación del texto de salida

El documento de salida es un documento estructurado. Su contenido coincide exactamente con el de entrada. En cuanto a la creación de las etiquetas en la salida, el algoritmo **abre** un nuevo elemento en los casos siguientes:

- Cuando encuentra un elemento del vocabulario v en el texto de entrada. El elemento correspondiente se abre en el documento de salida. Se trata de un elemento *semántico*. En el ejemplo del apartado 2.4.3 se crea un nuevo elemento `capitulo` cuando se encuentra la clave *Capítulo*.

- Cuando encuentra un fragmento de texto en la entrada a continuación de una etiqueta de apertura, sin que aparezca ningún elemento del vocabulario v. En este caso se trata de un elemento de *formato*, que será transferido intacto a la salida. Este es, por ejemplo, el caso de los párrafos.

[9] En una situación ideal no hay etiquetado redundante en los documentos de entrada. Esto es, no aparecen fragmentos de contenido desprovistos de vocabulario que determine el comienzo de elementos semánticos, que además estén rodeados por varios niveles de etiquetas. Por ejemplo, no habría párrafos tales como `<p><p><p> This is a paragraph.</p></p></p>`.

input	*vocabulary mapping*	*output*
<any-tag> keyword1 text ...	keyword1 T1	<T1> keyword1 text ...

input	*output*
<any-tag> text	<any-tag> text ...

Los elementos del documento de salida se **cierran**:

- Inmediatamente antes de abrir un elemento semántico. Por ejemplo, el comienzo de un nuevo capítulo implica el final del anterior. La etiqueta de cierre precederá a la de apertura del elemento que se va a crear.

input	*vocabulary mapping*	*output*
<any-tag> keyword1 text ...	keyword1 T1	**</T1>**<T1> keyword1 text ...

- Cuando se encuentra la etiqueta de un elemento de *formato* en la entrada.

input	*output*
...</any-tag>	...</any-tag>

Funcionamiento del algoritmo

El algoritmo (en la figura 2.11) está diseñado para trabajar utilizando otra aplicación (un parser XML en la implementación), que le proporciona sus entradas, las cuales ha extraído a su vez del documento XML original. Dado que no se controla cómo construye esta aplicación las entradas que proporcionará al algoritmo conversor (no hay garantías de que los elementos del vocabulario v no se reciban fragmentados en varias entradas), éste generaliza su modo de actuación para contemplar la posibilidad de recibir un elemento de v fragmentado en varias entradas sucesivas.

Las entradas procedentes de la aplicación de base son eventos de alguno de los tipos siguientes: STARTDOC (principio del documento), ENDDOC (fin del documento), STARTTAG (comienzo de elemento), ENDTAG (fin de elemento), TEXT (texto). Los tres últimos vienen acompañados por la cadena de caracteres encontrada por el analizador XML.

El algoritmo asociado al autómata traductor se muestra en la figura 2.11, y se explica a continuación. Una versión más detallada se encuentra en las páginas 37 a 38 de la memoria. La reacción según la entrada recibida es:

- STARTDOC: la copia de salida se inicializa. Puede limitarse a la creación de un nuevo documento o incluir la inserción de algún tipo de cabecera.

- ENDDOC: se da por terminada la copia de salida. Si había elementos abiertos en la salida, se cierran antes de considerar acabada la copia de salida. Se garantiza

así que el algoritmo termina correctamente y que el documento que se acaba de crear es correcto[10].

- STARTTAG: se preserva la etiqueta de entrada hasta que se reciba texto que permita decidir qué hacer con ella (ignorarla o traspasarla a la copia de salida). De hecho, esta etiqueta puede que sea ignorada más adelante (el elemento que rodea contiene vocabulario que sirve para decidir la creación de elementos semánticos), o que, en otro caso, se copie en la salida. Dado que no es posible predecir cuál de las dos situaciones se dará, se conserva momentáneamente.

- ENDTAG: la decisión depende de lo que se hiciese con la apertura equivalente. Si la etiqueta de apertura fue ignorada, ésta también lo será; si la apertura fué escrita en la salida, también se escribirá el cierre.

- TEXT: se utiliza para tomar una decisión acerca del etiquetado en la salida. El texto recién llegado se concatena con textos recibidos previamente, pendientes de analizar. La cadena resultante sirve para tomar una decisión. Si se trata del *comienzo de una clave* (elemento de v) en la entrada, se guarda hasta que se reciba nuevo texto. En caso de constituir una *clave completa*, se abre el elemento semántico correspondiente en la salida (cerrando si es necesario, los elementos de nivel igual o inferior en la jerarquía h). La última posibilidad es que se trate de *texto* perteneciente al contenido de un elemento, en cuyo caso se vierte al documento de salida tal cual. Puede deberse a una de dos situaciones: es el comienzo de un elemento de presentación, o es texto rodeado de más texto. En el primer caso, se abre el elemento correspondiente en la salida y se vuelca el texto. En el segundo, simplemente se escribe el texto.

La cadena que se examina en cada iteración es, bien la que se acaba de recibir, bien el resultado de concatenar esta última con los principios de clave guardados en iteraciones anteriores.

Proposición 2.2 *El algoritmo siempre termina.*

Prueba: El recorrido secuencial del documento de entrada garantiza que en algún momento se alcanza su final.

Proposición 2.3 *El documento de salida, h, está bien formado y su etiquetado es semánticamente correcto.*

Prueba: Está bien formado por construcción: los elementos semánticos estarán abiertos y cerrados correctamente, ya que el principio de uno de ellos va siempre precedido por el cierre de todos los de nivel igual o superior. Se garantiza la corrección del anidamiento, ya que se guarda la traza de los elementos abiertos en una estructura LIFO. Al final de la ejecución, todos los elementos abiertos sin cerrar, se cierran.

Lema 2.1 *Al acabar el documento, todos los elementos están cerrados.*

[10] XML bien formado y semánticamente "válido".

```
switch input do
   STARTDOC:  opens the output document
   ENDDOC:    ends the output document
   STARTAG:   keeps the tag until text is found
   ENDTAG:    if the tag closes an element with no content
                then ignores it
              or else
                if it closes a formatting element
                  then closes the element in the output
                  or else ignores the tag
                fi
   TEXT:      s = string to analyse[a]
              switch s do
                start of keyword:   keeps it till the next input comes[b];
                complete keyword: opens a semantic element in the output;
                no-keyword:       if it is at the start of a formatting element
                                  then
                                     opens the element in the output
                                  or else /* inside text of an element */
                                     writes s to the output
                                  fi
end
```

[a] s can be the string received with the input, or be the result of concatenating the input string with a start of keyword kept from the previous iteration.

[b] If the next input is also text, it will be concatenated with this string to obtain the string s to be analysed in that iteration.

Figura 2.11: Algoritmo de extracción de la estructura lógica-semántica.

Prueba: Los elementos que aún no se han cerrado al encontrar el final del documento de entrada permanecen en la pila de elementos abiertos. Dicha pila debe estar vacía al finalizar la ejecución del algoritmo.

Proposición 2.4 *Los elementos no semánticos se encuentran siempre en los niveles inferiores del árbol de salida.*

Prueba: Para los elementos que se importan de la entrada, la verificación es inmediata: estos elementos se cierran en la salida a imagen de la entrada. Dado que el documento de entrada está bien formado, hay garantía de que estos elementos están cerrados correctamente. En cuanto a los elementos formados por combinación de contenido y etiquetado de la entrada, cada uno de estos elementos comienza con una secuencia de etiquetas de apertura en el documento de entrada. El hecho de que éste sea un documento bien formado garantiza que se encontrará el cierre de todos los elementos. Así pues, el elemento se cerrará en la salida en el momento en que se encuentre su cierre en la entrada.

Versión detallada del algoritmo

Una versión detallada del algoritmo se encuentra en las páginas 37 a 38.

El bucle principal del algoritmo se ejecuta hasta encontrar el final del documento de entrada. En este momento, el documento de salida se termina correctamente (cerrando todos los elementos que aún no lo habían sido), tras lo cual el algoritmo finaliza.

Un pila ORIGEN sirve para guardar el rastro de las etiquetas de apertura encontradas en la entrada. Sirve de ayuda para decidir qué hacer cuando se encuentran las etiquetas de cierre equivalentes. Si la etiqueta de apertura fue ignorada, la de cierre lo será también. Si la primera se copió en la salida (correspondía a un elemento de formato), la segunda también lo será.

Una pila ABIERTOS mantiene la memoria sobre los elementos abiertos en la salida. Está pila es inspeccionada cada vez que se debe abrir un elemento semántico, para verificar si previamente se debe cerrar algún otro elemento. Cada vez que se abre un elemento semántico en la salida, todos las etiquetas de elementos de nivel igual o superior se sacan de la pila y se vuelcan a la salida. El anidamiento correcto de etiquetado en la salida está garantizado por el modo en que se han introducido las etiquetas en la pila.

Evolución del algoritmo sobre un ejemplo

El documento de la figura 2.12 es un extracto muy simple obtenido de uno de los documentos del prototipo que se describe en el capítulo 5. Este documento se ha utilizado como entrada al algoritmo (ha sido simplificado para su presentación en esta memoria). Se muestra la evolución en la creación del documento de entrada y las pilas. En cada estado se incluye el documento de salida, tal como se encuentra en este instante, el contenido de la pila ABIERTOS y el del pila ORIGEN.

Las transiciones entre estados están etiquetadas con el evento de entrada y la cadena que se recibe del *parser*. Algunas transiciones adjuntan una pequeña nota a la derecha (en negrita e itálica) que indica cuál es el criterio que ha servido para completar la decisión que determina la transición. Esta evolución se incluye en el apéndice A.

```
<doc>
<p>Ley 1.</p>
<h4><a>CAPÍTULO I.</a> DEL REFERENDUM.</h4>
<p><a>Artículo Primero.</a></p>
<p> Texto del artículo primero.</p>
<p><a>Artículo Segundo.</a></p>
<p>Texto del artículo segundo, previo a una disposición</p>
<p><b>Disposición final.</b></p>
<p>Texto de esta disposición.</p>
</doc>
```

Figura 2.12: Evolución del algoritmo sobre un ejemplo. Documento de entrada.

Algoritmo 1 Algoritmo de extracción de la estructura lógico-semántica. Versión detallada.

ENTRADAS: SourceDoc: D, DTDmapping: V, DTDhierarchy: H
SALIDAS: TargetDoc: STR

```
while there are inputs from SourceDoc do
switch input do
 BEGINDOC:
   open(target)
   write-header(target)
   searching-keyword ← false

 ENDDOC:
   while not empty(ABIERTOS) do
     ele ← pop(ABIERTOS)
     close(ele)
   end while

 STARTTAG:
   push(ORIGEN, 'candidata')

 ENDTAG:
   if top(ORIGEN) = 'candidata' then {empty element}
     ele ← pop(ORIGEN) {ignore empty elements}
   else
     if top(ORIGEN) ≠ 'ignorada' then {it closes a formatting element}
       close(pop(ABIERTOS)) {close the element in the output}
     else {top(ORIGEN) = 'ignorada'}
       ele ← pop(ORIGEN) {ignore the closing tag}
       if top(ORIGEN) ≠ 'ignorada' then {it marks the end of a semi-semantic (title) element}
         write(pop(ABIERTOS))
       end if
     end if
   end if

 TEXT:
   if searching-keyword then
     s ← concat(keycandidate,text)
   else
     s ← text
   end if
   if startkeyword(s) then
     searching-keyword ← true
     split(s,s1,keycandidate) {keycandidate = start of some keyword}
     if s1 is not null then
       write(s1) {it is certain that the text fragment preceeding the possible start of the semantic element
       belongs to a previous element}
     end if
   end if
   if completekeyword(s) then
     split(s,s1,keycandidate)
     if s1 then
       write(s1) {fragment text before the candidate start of semantic element belongs to some other
       element with certainty}
```

```
    end if{replace the top sequence of 'candidata' in ORIGEN by 'ignorada'}
    while top(ORIGEN)='candidata' do
      ele ← pop(ORIGEN)
      push(AUXILIAR, 'ignorada')
    end while
    while not(empty(AUXILIAR) do
      push(ORIGEN, pop(AUXILIAR))
    end while{candidate tags have been ignored}
    outputtag ← eqtag(keycandidate,DTDmapping) {obtain the equivalent tag from the vocabulary mapping}
    if in(outputtag,ABIERTOS) then
      while top(ABIERTOS) ≤ outputtag do {close hierarchically lower or equal elements}
        ele ← pop(ABIERTOS)
        close(ele)
      end while
    end if
    open(outputtag)
    push(ABIERTOS, outputtag)
    write(keycandidate)
    searching-keyword ← false
    keycandidate ← null
  end if
  if notkeyword(s) then
    if top(ORIGEN)='candidata' then {start of a formatting element}
      while top(ORIGEN) = 'candidata' do
        push(AUXILIAR, pop(ORIGEN))
      end while
      while not(empty(AUXILIAR)) do
        ele ← pop(AUXILIAR)
        push(ABIERTOS, ele.tag)
        open(ele.tag)
      end while{candidate tags are open in the target document in an ordered manner: text can be written}
      write(s)
      searching-keyword ← false
      keycandidate ← null
    else {in the middle of the text of some element}
      write(s)
    end if
  end if

 end switch
end while
```

2.5 Aplicación de la propuesta a los documentos del entorno jurídico

Los documentos del prototipo de prueba de esta tesis son textos normativos españoles, jurisprudencia y otros. Cada uno de ellos constituye una clase, cuyas gramáticas se describen en el capítulo 5. El algoritmo de extracción de la estructura lógico-semántica se utiliza en estos documentos para obtener copias normalizadas a partir de otras procedentes de servidores públicos. Las copias así generadas son documentos

XML, cuya estructura lógica (y DTD de la clase correspondiente) concuerda con las divisiones semánticas del documento abstracto. El ejemplo del apartado 2.4.3 muestra la aplicación del algoritmo a un texto normativo. La gramática asociada a la clase "texto normativo español" es la que se puede ver en la figura 2.4, y su jerarquía se encuentra en la figura 2.10.

2.6 Discusión

La mayor parte de los esfuerzos dedicados a la estructura lógica de los documentos están orientados a obtener la gramática (o DTD en el caso de SGML o XML) asociada a una clase de documentos. En esta tesis dicha gramática se considera conocida y los esfuerzos se concentran en obtener la estructura lógico-semántica de una instancia de la clase, utilizando como guía en dicho proceso el conocimiento sobre la clase. Otros dos trabajos que se centran en copias individuales son los de Smith y López [102] y Lim y Ng. [83]. Sin embargo, en [102] utilizan como guía los elementos del documento de entrada para buscar en su interior los conceptos que les interesan, mientras que en [83] buscan la jerarquía en el documento de entrada, lo cual significa que la gramática que utilizan es la asociada a dicha entrada. La propuesta de esta tesis se basa en la gramática de salida (la DTD de entrada es indiferente) y el reconocimiento de los elementos no se guía por la existencia de conceptos, sino por la presencia de vocabulario clave.

El algoritmo que se ha presentado en este capítulo se ocupa de descubrir la estructura semántica de un documento. Respecto a su estructura lógica, que no coincide con la que se busca, se sabe que el anidamiento es correcto. También se utiliza el vocabulario del contenido. Sin embargo, no se utiliza ningún tipo de conocimiento sobre los tipos de elementos que pueden aparecer en la estructura lógica (o la DTD) del documento de entrada. Por ello, siempre que es posible trabajar en base al contenido se utiliza éste con preferencia al etiquetado. Difiere en este punto con la propuesta presentada en [83], que descubre la estructura lógica del documento analizado, sabiendo cuál es su DTD. Por otro lado, el algoritmo aquí presentado se basa en la existencia de vocabulario preciso y conocido para detectar los límites de elementos semánticos. Con ello, el modo de trabajar es diferente del de [102]: en primer lugar, aquí no se utilizan las divisiones del documento de entrada, y en segundo lugar, al no buscar *conceptos* no es necesario explorar por regiones.

El algoritmo que se ha propuesto en la sección 2.4 no pretende obtener la gramática de ninguna clase de documentos (ni de entrada, ni de salida) puesto que se dispone de información sobre la correspondiente a la clase de salida. Tampoco es el objetivo hacer transformaciones de DTD, ya que la DTD de entrada es desconocida. Una de las ventajas de este algoritmo es que el conocimiento de la estructura lógica de entrada es mínimo, lo cual amplía sus posibilidades de aplicación.

Pero la más importante ventaja de este algoritmo está en su propia finalidad: disponer de una copia de un documento cuya estructura lógica se corresponda con la estructura semántica de la entidad abstracta. Aparte de la información propiamente semántica, esto tiene múltiples ventajas, como son disociar el contenido de la presentación (pudiendo así modificar de modo más fácil y general las reglas de presentación), o la

que se discutirá también en el capítulo 3 de obtener una equivalencia directa entre las referencias utilizadas en los textos y la estructura lógica de la copia digital, que de otro modo sería, cuando menos complicado, sino imposible. Esta última ventaja es determinante a la hora de plantear una explotación avanzada de las relaciones entre documentos.

No obstante, no es posible obtener un algoritmo de traducción general, que funcione con cualquier gramática en la entrada y permisibilidad absoluta en su sintaxis. Es necesario restringir las características de la copia de entrada, lo cual se ha hecho teniendo en cuenta las características de los documentos de entrada utilizados en el prototipo. Así, por ejemplo, se han tomado algunas decisiones como las que se explican:

- El anidamiento entre elementos semánticos se restringe para evitar confusiones entre el comienzo de un elemento y las referencias a otros documentos, dado que el vocabulario utilizado en ambas situaciones es el mismo.

- El etiquetado de niveles inferiores en la entrada (párrafos, títulos, etc.) se preserva, ya que no hay modo de reconocerlos en base al contenido. Por ello no se plantea en ningún momento la posibilidad de eliminar el etiquetado del documento de entrada. Además, dichas divisiones son frecuentemente utilizadas en las citas de los documentos del prototipo, lo cual hace imprescindible conservarlas.

Las posibilidades de ampliación del algoritmo se encuentran en su aplicación a documentos con mayor flexibilidad en las estructuras lingüísticas asociadas a su vocabulario y en los documentos no etiquetados. Mayor flexibilidad en las expresiones asociadas al vocabulario supondría un análisis más elaborado del contenido del documento. Esto se localizaría en una adaptación de los vocabularios v para que los métodos que implementan dichas equivalencias tengan en cuenta las nuevas variaciones.

En cuanto a los documentos que no presentan ningún tipo de etiquetado, son en realidad una simplificación de la situación que se ha considerado en este trabajo: bastaría con reconocer los elementos semánticos en base al vocabulario v, sin preocuparse en absoluto por el etiquetado de la entrada. A este caso son aplicables las mismas consideraciones que se han hecho cuando se ha justificado por qué no se ha planteado la eliminación previa del etiquetado en las copias de entrada como la primera fase del algoritmo.

3

Las relaciones entre documentos

Índice General

Las relaciones entre los documentos, la forma de modelarlas y las posibilidades de explotarlas centran la atención de este capítulo. La propuesta que se hará en este capítulo se ocupa de mostrar cómo modelar las referencias entre documentos y sus modificaciones como enlaces, así como del modo de explotar estas relaciones. Dos tipos de explotación son las que se contemplan: la interrogación de estas relaciones y la generación de nuevos documentos (versiones históricas) apoyada en dicha interrogación.

Las relaciones entre documentos pueden tener orígenes y significados diversos, como las relaciones semánticas y los enlaces explícitos en el interior de los documentos a otros. Las relaciones consideradas en esta tesis derivan de las referencias entre documentos. Sean de uno u otro tipo, las relaciones se pueden representar siempre en un grafo de enlaces, donde los recursos participantes en las relaciones son los vértices y los arcos se corresponden con los enlaces que representan las relaciones. Si además las relaciones son heterogéneas, se puede asociar a los enlaces un *tipo*, que representa la naturaleza de la relación [108]. El uso más extendido hasta ahora de este grafo es la creación de hipertexto, por el cual los usuarios pueden navegar [39]. Sin embargo, este grafo se puede aprovechar con otros propósitos, como la consulta de relaciones [49], que es el modo en que se hace en esta tesis. Adicionalmente, en esta tesis se aprovecha el hecho de disponer de documentos estructurados. Cuando se trabaja con este tipo de documentos, los vértices de los enlaces pueden ser fragmentos en vez de documentos completos. Este aspecto, hasta ahora poco aprovechado, puede ser reflejado en el grafo, cuya granularidad será suficiente para permitir manipulaciones avanzadas de documentos, que de otro modo (con documentos completos en los vértices) serían imposibles.

Una de estas manipulaciones está relacionada con el segundo aspecto abordado en esta tesis en lo que a explotación de relaciones se refiere. La existencia de *versiones* de los documentos da lugar a varios problemas de relaciones entre documentos y versiones. Las versiones de un documento son variaciones del mismo documento abstracto. Las variaciones pueden afectar a su contenido y, opcionalmente, a la estructura lógica (este concepto se presentó en el capítulo 2). Existe una relación obvia entre las versiones de un mismo documento [93, 121], de la cual surge la problemática de cómo expresar este tipo de relación y su mantenimiento [34, 38, 101]. Sin embargo, en esta tesis el problema se aborda de modo diferente: en vez de mantener las versiones, las relaciones entre ellas, u ocuparse de detectar esta relación mediante la comparación de documentos [36], lo que se hace es generar dinámicamente las versiones. Las versiones son en muchos casos el resultado de modificaciones hechas a versiones anteriores, razón por la cual se las denomina *versiones históricas*. Estas modificaciones, en definitiva, son un tipo más de relación. Y en esta afirmación se apoya nuestra propuesta de generación dinámica de versiones. La generación de versiones se basa en el siguiente planteamiento: si es posible consultar las relaciones y manipular el grafo de enlaces asociado, ha de ser igualmente factible generar las versiones con un recorrido de dicho grafo. El algoritmo que realiza el recorrido durante el cual se generan las versiones se presenta en la sección 3.7.

El modo escogido para implementar las relaciones es una base de enlaces (uno de los componentes arquitectónicos presentados en el capítulo 4). Los estándares asociados a XML, XLink [119] y XPointer [116], aportan la flexibilidad necesaria para modelar el grafo de enlaces: multidireccionalidad, grado, etc.; por otro lado, parecen la elección más conveniente cuando los documentos se han modelado con XML, ya que se continúa garantizando así la interoperabilidad inherente a los datos XML.

La creación de las colecciones de enlaces utilizadas se realiza durante un proceso de detección de citas (referencias) en los documentos analizados, ya que las relaciones que centran el interés de este trabajo son detectables a partir de las referencias a otros documentos encontradas en el texto de los mismos.

3.1 Tipos de relaciones entre documentos en las bibliotecas digitales

Los documentos pueden estar relacionados por motivos diversos: pueden compartir autor, tratar del mismo tema, un documento puede contener referencias a otro(s) documento(s), contener enlaces hipertexto, etc. Estas relaciones son una fuente de información que en algunos casos puede ser tan importante como los propios documentos [52].

Existen varias clasificaciones para las relaciones y enlaces entre documentos, procedentes en su mayor parte del entorno de hipertexto [47, 92, 4, 108]. En esta tesis se parte de una clasificación general, dentro de la cual se encuadrarán las relaciones tratadas en la tesis. Así pues, dos o más documentos pueden estar relacionados porque:

- Existen relaciones *semánticas* entre ellos. Ejemplos son los casos en que los documentos comparten autor, temática, están incluidos en los mismos catálogos, etc. Este es un tipo de relación que puede ser difícil de detectar incluso para un humano, lo cual deja claro que aún se está lejos de conseguir una detección automática para estas situaciones. La mayor parte de los trabajos que han conseguido algún resultado se basan en algún tipo de clasificación [67, 68, 108, 71, 46, 23, 21].

- Un documento *referencia* (cita) otro u otros documentos. En este caso la relación es evidente para una persona, si bien la detección automática puede ser tan complicada como complejo sea el lenguaje utilizado en el documento que contiene las referencias. Ejemplos son las citas bibliográficas en los artículos científicos o técnicos [67, 68, 31, 20] y las citas entre documentos jurídicos [121]. Las referencias entre artículos científicos son aprovechadas para mejorar los periódicos y revistas electrónicos, que pueden así compartir recursos (por ejemplo, figuras). En el caso de los documentos jurídicos las referencias son muy frecuentes y son, además, la información crucial que permite acceder a documentos necesarios para hacer una interpretación correcta de aquellos que contienen las referencias [121]; por ejemplo, es imposible interpretar una sentencia judicial si no se dispone del texto de la ley (o leyes) en que se apoya dicha sentencia.

- Hay enlaces *explícitos* en un documento hacia otro documento. Dichos enlaces han sido incluidos por el autor en el momento de su creación, y la causa de su existencia puede ser de cualquier tipo. Sin embargo, en muchos casos la naturaleza del enlace (el motivo que dio lugar a su existencia) no se encuentra reflejada en él y únicamente su creador es consciente de ella. El ejemplo común son los enlaces HTML. Este tipo de enlace puede representar relaciones semánticas, pero su diferencia con el primer grupo es que en este último caso, la semántica de la relación se ha perdido durante la creación del enlace.

Una última clase de relación se da entre los elementos de los documentos estructurados, y es la que liga dos elementos por una relación de inclusión (jerarquía en el árbol estructural); son los enlaces *estructurales* [108]. Se pueden incluir aquí las relaciones que ligan porciones de contenido cuya agregación da como resultado un documento compuesto [109, 68]. Las piezas que forman el documento resultante pueden ser documentos completos o elementos procedentes de documentos estructurados; en este último caso las relaciones son las procedentes de la jerarquía de inclusión en el árbol documental (ver sección 2.2 en el capítulo 2). Estas relaciones se han almacenado como un tipo más en algunos casos [3], si bien con la llegada de estándares para los documentos estructurados como XML se encuentran implícitos en el documento.

El modo más popular de utilizar y explotar las relaciones entre documentos en la actualidad es la creación de hipertexto navegacional [39, 26].

3.2 Enlaces

Las relaciones entre documentos se pueden modelar como *enlaces*. Los enlaces hipertexto (los más populares en la actualidad) presentan las siguientes características [108]:

- Un enlace tiene un *origen* y un *destino*.
- Un enlace se utiliza para activar una acción de navegación en el origen que lleva al destino.
- Un enlace representa algún tipo de relación entre el origen y el destino.

3.2.1 Grafos de enlaces

Las relaciones entre documentos se pueden modelar como enlaces, dando como resultado un *grafo de enlaces*, cuyo arcos pueden ser dirigidos y estar *etiquetados*[1]. Los vértices del grafo son los documentos relacionados y sus ejes o arcos los enlaces [39].

Las referencias y los enlaces explícitos son *dirigidos*: tienen un *origen* (el documento que referencia o contiene el enlace) y un *destino* (el documento referenciado o al cual apunta el enlace). Los enlaces derivados de relaciones semánticas pueden estar dirigidos o no, según la naturaleza de la relación. Los vértices del grafo se encuentran conectados por arcos (o ejes) desde el origen del enlace al destino. El tipo del enlace es la *etiqueta* asociada al arco correspondiente. Los grafos hipertexto son los más populares [39], de los cuales hay un ejemplo en la figura 3.1.

Una selección de porciones del grafo en base al tipo de enlace produce *grafos parciales* que contienen los mismos nodos y parte de los enlaces del grafo original. El grado de cada vértice de un enlace es el grado del nodo asociado en el grafo: el número de arcos (enlaces) que tienen el nodo en alguno de sus extremos.

Ejemplo 11. Los enlaces entre versiones (*revision links*) [93, 47] se pueden representar con arcos entre el documento original y sus revisiones (versiones), o desde las versiones

[1] El *tipo* de un enlace se corresponde con la etiqueta que indica la naturaleza de la relación que liga los nodos conectados.

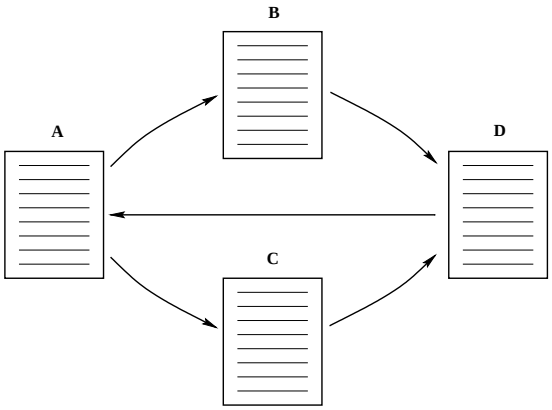


Figura 3.1: Ejemplo de grafo utilizado para navegar por hipertexto. El documento A contiene enlaces cuyos destinos son B y C. A su vez estos dos documentos tienen sendos enlaces hacia D, cuyo enlace conduce nuevamente hacia A. Un recorrido cuyo punto de partida sea A puede pasar por B (o C), alcanzar posteriormente D y finalizar retornando a A.

(revisiones) hacia él; los arcos dirigidos entre los dos documentos sitos en los vértices (el original y la versión) muestran simultáneamente la relación y la secuencia temporal entre las versiones (la ordenación temporal se encuentra implícita en la dirección). □

Documentos estructurados: granularidad en el grafo

Trabajar con documentos estructurados posibilita discernir los fragmentos implicados en una relación con mayor precisión. En estos casos los nodos del grafo no representan documentos completos, sino fragmentos de documentos [53, 29], estructurados en una jerarquía de inclusiones representada por un árbol (ver capítulo 2). La ventaja en este caso es que los fragmentos implicados en las relaciones se pueden localizar en base a su posición relativa en el árbol estructural. El número de nodos del grafo resultante es superior que cuando la unidad mínima de información es el documento, y la jerarquía hace posible seleccionar subconjuntos de nodos (porciones de documento), utilizando lenguajes adecuados [40, 115]. Los árboles jerárquicos asociados a los documentos forman parte del grafo de enlaces, al cual aportan las relaciones estructurales. Sobre los enlaces se pueden realizar consultas [49], y pueden ser aprovechados además para componer nuevos documentos [109, 103]. También las relaciones jerárquicas de los documentos estructurados pueden ser objeto de consultas [1, 2, 49, 50].

Ejemplo 12. Las anotaciones son un buen ejemplo de enlaces que afectan a fragmentos de documentos. Por ejemplo, las relacionadas con piezas de teatro suelen afectar a escenas o actos concretos en vez de al documento completo (en este último caso serían consideradas "comentarios" en vez de anotaciones). □

La definición de grafo de enlaces que sigue, a la cual se adaptan los algoritmos presentados en adelante, tiene en cuenta las consideraciones que se han comentado sobre la granularidad que se encuentra cuando se trabaja con documentos estructurados.

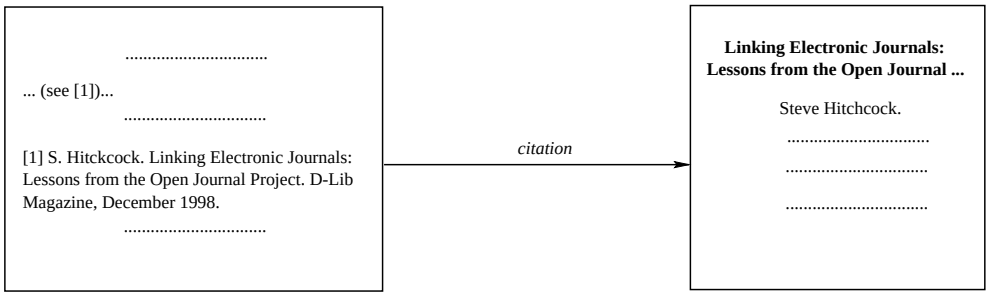


Figura 3.2: *Citation linking*: el documento de la izquierda *referencia* el de la derecha de la figura. La dirección del arco se corresponde con la de la relación.

Definición 3.1 *En el contexto de los documentos estructurados, y restringido a las relaciones que dan lugar a enlaces dirigidos, se puede definir un* grafo de enlaces *como un grafo de enlaces dirigidos y etiquetados* $G = (N, E)$ *compuesto por dos conjuntos finitos: un conjunto de nodos* N *y un conjunto de arcos* E*. Los nodos de* N *son fragmentos de documentos. Cada arco* $u = ((i, j), t) \in E$ *es una tupla que cumple:*

- (i, j) *es un par ordenado de nodos donde* i *es el* origen *del enlace y* j *es el* destino *del enlace,*
- t*, la etiqueta del arco, es el* tipo *del enlace.*

Cada arco u *representa un enlace con tipo.*

Ejemplo 13. El campo *citation linking* está ligado mayoritariamente a los trabajos relacionados con las revistas *on-line*. En este entorno, las citas bibliográficas son la fuente más importante de relaciones. La figura 3.2 muestra un grafo con un único enlace, donde cada nodo representa un documento. El *arco* que los une tiene la siguiente semántica: los documentos están enlazados si uno de ellos *cita* al segundo (de ahí la etiqueta *citation*). El arco *dirigido* significa que el documento al cual apunta (el *destino* del enlace) es el documento citado, mientras que el otro (*origen* del enlace) es el que cita. □

3.3 Enlaces y estándares de documentos estructurados

XML tiene estándares asociados que permiten modelar las relaciones entre documentos, las cuales serán a su vez datos XML. La capacidad de estos estándares supera a la de los tradicionales enlaces hipertexto (básicamente, HTML) en cuanto a flexibilidad y potencia semántica. Dichos estándares son XLink, que permite enlazar recursos, y XPointer, que permite direccionar fragmentos dentro de esos recursos[2].

XLink (*XML Linking Language*) [119] permite modelar relaciones entre dos o más recursos (o porciones). En este estándar, un *enlace* representa esta relación, y se modela

[2]En el momento de esta redacción (enero 2001) aún no son recomendaciones *W3C* estables.

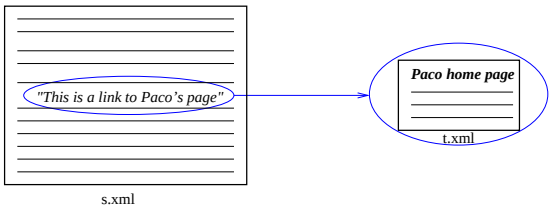


Figura 3.3: Enlace simple.

con un elemento *XLink* . Esta definición no coincide exactamente con la noción de *arco* en los grafos de enlaces que se introdujo en la sección 3.2. De hecho, un "enlace XML" (*xlink*) es la unión de uno o más arcos que tienen "algo" en común (la semántica de la relación).

XLink permite insertar los enlaces unidireccionales tradicionales en los documentos, pero se puede ir más allá. Esto quiere decir que, al utilizar *XLink* , es posible:

- representar relaciones entre dos o más recursos (enlaces de grado n)
- asociar metadatos al enlace
- crear bases de enlaces que residen separadamente de los recursos relacionados.

De este modo se puede modelar la información del grafo de enlaces: los vértices son los recursos (documentos, imágenes, etc.), los arcos son los elementos `arco` de los xlink, las etiquetas se pueden modelar en los atributos `role` y otros metadatos sobre los arcos y vértices pueden representarse mediante atributos (de los recursos o arcos).

3.3.1 XLink

La versión de *XLink* que se describe en esta sección es la que se ha utilizado en la implementación del prototipo de esta tesis: el *Working Draft* de 21 de febrero de 2000 [119].

Hay dos categorías principales de enlaces *xlink*. Los xlink **simples** (*simple*) que se pueden utilizar para los enlaces más comunes: los que tienen sólo dos vértices, embebidos el interior de los documentos (*inline*). El vértice origen del enlace es el documento en el cual reside, mientras que el vértice destino es un recurso remoto. Son similares a los ya bien conocidos enlaces HTML (elementos `a`).

Ejemplo 14. El enlace simple que se muestra a continuación corresponde al enlace de la figura 3.3. El contenido del origen del enlace es el contenido del elemento xlink, y el destino se indica en el valor del atributo `xlink:href`.

```
<simplelink xlink:type="simple" xlink:href="http://www.bla.bla/paco.html">
            This is a link to Paco's page
</simplelink>
```

□

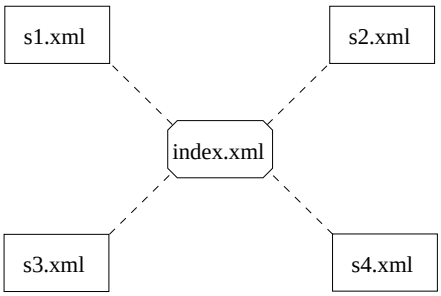


Figura 3.4: Enlace extendido.

Los enlaces **extendidos** (*extended*) poseen toda la funcionalidad que *XLink* es capaz de ofrecer: enlaces *out-of-line*, multidireccionales, y enlaces en los que pueden participar un número n de recursos superior a 2. Puede ser *inline* o *out-of-line*, y son los únicos donde se pueden asociar más de dos recursos en un mismo enlace o ligar recursos que no se puede modificar (por ejemplo, documentos ubicados en colecciones en las cuales no se tiene permiso de escritura). Los enlaces extendidos son los más interesantes, ya que con ellos se pueden representar grafos tan complejos como se desee (algo que, por ejemplo, es imposible hacer con los enlaces simples y HTML).

Los atributos de los elementos XLink pueden ser algunos de los definidos en el estándar, o creados a medida por los autores de los enlaces. *XLink* proporciona atributos para indicar el *tipo* (`type`) de un recurso, el papel con el que participa en la relación (`role`) y para direccionar los recursos (`href`)[3]. En cuanto a los enlaces extendidos (que pueden asociar N recursos y un número arbitrario de arcos, como en la figura 3.4), el estándar define un tipo específico de elemento para los arcos. Así, los elementos que forman parte de un enlace extendido pueden ser nodos del grafo, o bien arcos que unen estos nodos. Estos arcos se distinguen de los restantes elementos porque presentan el valor *arc* en el atributo *type*.

3.3.2 Fragmentos de documentos: XPointer, XPath.

Así como *XLink* permite direccionar documentos, XPointer [116] direcciona fragmentos internos (un punto, un conjunto de nodos, un intervalo) de documentos XML. La parte de la dirección de un recurso que corresponde al fragmento interno debe construirse con XPointer. Se incluye dentro de un atributo `xlink:href`, como argumento de la función `xpointer()`, y siempre detrás del identificador del recurso al cual pertenece (URI); el carácter '#' marca la separación entre el URI y el XPointer.

XPointer se asienta sobre el lenguaje XPath (*XML Path Language*). La mayor parte de los xpointer son *location paths*, construidos con la concatenación de varios *location steps*. Cada uno de ellos direcciona un punto concreto del documento, en general relativo a algún otro, como por ejemplo el principio del documento u otro *location step*. A este punto de referencia se le llama *context node*. En general, un *location step* consta de tres

[3]Se pueden encontrar más atributos en la especificación XLink.

partes: *axis*, *node test* y un predicado opcional.

axis::node-test[predicate]

El *axis* indica en qué dirección se debe buscar el *context node*. El *node-test* indica qué ejes seleccionar dentro del *axis*. El predicado es una expresión booleana que se comprueba sobre cada uno de estos nodos. Únicamente los nodos que se encuentran en los tres subconjuntos serán seleccionados finalmente.

Ejemplo 15. La localización de los elementos internos de los documentos se puede hacer con XPath, como en el ejemplo de la figura 3.5, donde se selecciona el primer elemento de tipo `articulo` dentro del primer elemento `disposicion` del documento `l13-1986.xml`. En este ejemplo hay dos *location steps*:

1. `descendant::disposicion[1]`

   El `axis` superior selecciona todos los descendientes del *context node*. El *node-set* indica que únicamente los descendientes de tipo `disposicion` deben ser considerados. El predicado selecciona el primer elemento de tipo `disposicion` entre todos los descendientes.

2. `articulo[1]`

   En este *location step* no hay *axis*; por defecto, se consideran todos los hijos del *context node* en curso (el primer elemento `disposicion` de todos sus descendientes). Entre ellos, se seleccionan los elementos de tipo `articulo`. Una vez más, se selecciona el primero.

□

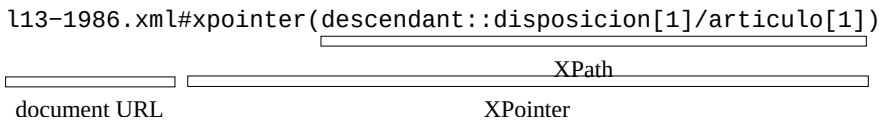


Figura 3.5: Valor de un atributo *xlink:href*. La dirección interna es un XPointer que contiene un XPath. Este direcciona el primer elemento de tipo *articulo* dentro del primer elemento de tipo *disposicion* del documento *l13-1986.xml*.

3.4 Versiones de documentos

Dado un documento abstracto, `D`, pueden existir varias versiones del documento, correspondientes a modificaciones hechas a `D`. Todas las versiones tienen un grado de similaridad alto, si bien nunca son totalmente iguales. El problema al que dan lugar es el siguiente: "dado un documento abstracto `D` y el conjunto de versiones de `D` (que también son entidades abstractas), ¿es posible acceder a cualquier versión de `D`, permitiendo a su vez al usuario abstraerse (o no, según le convenga) de la existencia de varias versiones distintas?"

Según DeRose, las versiones de documentos pueden deberse al menos a dos causas [47]:

- *Traducciones* de un documento a varias lenguas, que comparten la misma estructura lógica y difieren totalmente en el contenido (aunque no en su semántica).

- Versiones *históricas* de un documento, debidas a modificaciones parciales de su contenido (y su estructura). La semántica puede verse afectada.

Las versiones históricas plantean varias cuestiones: almacenar o no todas las versiones, cómo modelar las relaciones entre ellas, cómo detectar su existencia, ... La respuesta a estas cuestiones depende en gran medida del fin del sistema en el cual se presenta este problema. La solución más evidente es el almacenamiento simultáneo de todas ellas [121, 38]. Globalmente se puede hablar de tres aproximaciones distintas para el tema de las versiones:

1. Enlazar las versiones relacionadas. Este tipo de enlace recibe el nombre de *revision links* desde que Parunak los denominó de este modo en 1990 [93]. En este caso se mantienen simultáneamente dos bases de datos: la que contiene documentos y la de enlaces. Esta es la aproximación seguida por Wilson [121] y Choquette y cols. [38]. El mayor inconveniente de esta solución es la dificultad para mantener la base de enlaces actualizada [34, 38, 101].

2. Considerar diferentes instantáneas de la base de datos y compararlas para detectar los cambios debido al hecho de que un objeto ha sido reemplazado por una nueva versión [34]. Esta solución se ha utilizado en bases de objetos, razón por la cual se puede considerar cuando los documentos se encapsulan como objetos. En esta solución la base de enlaces desaparece y los cambios se representan indirectamente como la diferencia entre dos estados de la base de datos [36]. Es aplicable a versiones históricas, pero no a las traducciones.

3. La tercera propuesta proviene del área de los datos semiestructurados. Chawathe y cols. [36, 35] modelan los cambios que han sufrido datos estructurados jerárquicamente (que es el caso de los documentos estructurados) como cambios en los nodos del árbol documental. Representan los cambios como anotaciones (atributos) a los nodos afectados, facilitando así las consultas acerca de la "historia" de un documento. La detección de los cambios se realiza aquí comparando los árboles de los documentos. A diferencia de las dos propuestas anteriores, ésta es la primera solución donde se tiene en cuenta la estructura de los documentos, de modo que las anotaciones se asocian a elementos del documento en vez de a documentos completos.

   Esta idea de anotar los nodos con atributos que contienen información sobre su evolución histórica se puede encontrar en algunos servidores públicos [79], donde los elementos están calificados con atributos que comentan cuáles son las modificaciones que han sufrido posteriormente a la creación del documento donde se encuentran.

3.5 Relaciones y versiones históricas en el entorno jurídico

Los documentos jurídicos cumplen unas condiciones que los hace especialmente convenientes para mostrar la importancia que el aspecto de las relaciones entre documentos puede llegar a tener. De hecho, están muy relacionados y sufren frecuentes modificaciones, lo cual da lugar a nuevas versiones. Este último es un problema especialmente relevante en este entorno; el acceso adecuado a las distintas versiones facilita el trabajo de los especialistas, mientras que no disponer de ellas puede ser el obstáculo definitivo para la realización de dicho trabajo.

3.5.1 Relaciones

En los documentos jurídicos, además de las relaciones semánticas (documentos de la misma categoría, procedentes de las mismas Cortes, etc.) existen numerosas referencias entre ellos. Cualquier documento puede referirse a varios textos normativos anteriores, jurisprudencia, etc.; del mismo modo, cualquier documento puede estar referenciado desde varios documentos. Por ejemplo, algunas leyes son en realidad una colección de modificaciones a normas anteriores, de modo que cada una de estas modificaciones supone una referencia a la norma modificada.

Estas relaciones pueden ser una fuente de información preciosa para un jurista [121] en situaciones como la preparación de un caso. La interpretación de una sentencia requiere disponer, además de la propia sentencia, del texto de las normas en las cuales se basa la sentencia (a las cuales hace referencia). Merece la pena resaltar que la explotación de estas relaciones en ambos sentidos es otro requisito: tan importante es conocer la jurisprudencia relacionada con una cierta norma, como las normas relacionadas con una cierta sentencia. Es decir, debe ser posible recorrer el grafo de relaciones en cualquier dirección y sentido.

Representación de relaciones

La forma en que se han representado las relaciones en este entorno ha evolucionado de modo paralelo a los estándares relacionados con los documentos estructurados. Las soluciones previas a la aparición de XML optaban por almacenar el contenido de los documentos, su estructura y las relaciones en bases separadas. La estructura se modelaba como *enlaces estructurales* [3, 38]. Las soluciones que se basan en SGML y XML no necesitan representar de forma explícita los enlaces estructurales, implícitos en el estructura del documento. En su mayoría, estas soluciones modelan las relaciones embebidas en el interior de alguno de los documentos implicados en la relación [65, 56]. Esta elección frecuentemente viene determinada por la utilización de HTML, que obliga a hacerlo de este modo [79].

En cualquier caso, el hipertexto ha sido generalmente aceptado como la solución idónea para este tipo de información [3, 62], de modo que los hiperenlaces sirven para que el usuario navegue y reconstruya él mismo durante el proceso de navegación los documentos que le interesan.

3.5.2 Versiones

Otra de las particularidades de los documentos jurídicos es que sufren frecuentes modificaciones parciales a su contenido (por ejemplo, así ocurre a menudo con los textos normativos), lo cual da lugar a versiones históricas de un mismo documento, cada una de ellas válida durante el período comprendido entre la publicación oficial de la modificación y el momento en que se publica la siguiente. Claramente, el texto con el que se debe trabajar en cada momento es aquél que es jurídicamente válido. También, la revisión de procesos ocurridos en períodos concretos debe hacerse teniendo en cuenta el texto vigente en el momento histórico en que dicho proceso tuvo lugar [121].

En lo que a las versiones se refiere, se han planteado tres tipos de propuestas en las bibliotecas legislativas implementadas hasta ahora:

1. Mantener simultáneamente las versiones de un documento [38, 101], que se conectan mediante enlaces. Las versiones se relacionan mediante *revision links*. La mayor dificultad que se ha encontrado es mantener actualizados dichos enlaces, así como los que afectan a los documentos (que, por consiguiente, son susceptibles de afectar igualmente a todas y cada una de sus versiones).

2. Modelar las modificaciones como atributos, cuya resolución por parte *del usuario* llevaría a la obtención de la versión deseada [56]. El objetivo al que se da prioridad en este caso es facilitar las consultas acerca de la "historia" de un documento (si ha sufrido o no modificaciones desde su promulgación, y cuáles han sido). Esta solución, no obstante, es compatible con la anterior.

3. Almacenar las reglas de generación de cada versión [13]. Esta aproximación coincide en su objetivo con la presentada en esta tesis. Sin embargo, difieren en el modo de conseguirlo: en [13] no se utilizan los enlaces para este proceso. Esta particularización de las reglas de generación para cada versión tiene la ventaja de su precisión y eficiencia (que, como se comenta en la discusión del capítulo 4, es una de sus prioridades), pero la desventaja de que, al especificar para cada versión sus reglas de generación, dicha creación no se puede llevar a cabo para aquellos documentos cuyas reglas aún no se han explicitado, incluso aunque todos los documentos necesarios para la composición estén disponibles. Un método más general, como el que se propondrá en este capítulo, es, lógicamente, menos eficiente, pero tiene la ventaja de que permite la generación de *cualquier* versión para la cual los documentos base de los cuales extraer los fragmentos utilizados en la composición estén en la biblioteca.

3.6 Modelado de referencias y modificaciones usando enlaces etiquetados

3.6.1 Las relaciones modeladas

Hay dos clases de relaciones que, junto con los enlaces estructurales, conforman el grafo de relaciones considerado en esta tesis: aquellas que no modifican ninguno de los documentos participantes en la relación, y las que modifican alguno de ellos. De este modo, es posible distinguir:

- *Referencias* entre documentos, causadas por la existencia de referencias (o citas) en algún documento a otro u otros. Las relaciones de este tipo no modifican ninguno de los participantes en la relación.
- *Modificaciones*, en las cuales un documento contiene la información sobre cómo modificar otro. La aplicación (resolución) de la modificación da lugar a una nueva versión del documento modificado. Las modificaciones pueden consistir en la inserción de nuevo texto, la sustitución de texto por otro, o la eliminación de parte del contenido. Todas ellas se pueden reducir a *sustituciones*: una inserción es la sustitución de un nodo nulo por el texto que se debe insertar y una eliminación es la sustitución de un nodo no nulo por el nodo nulo. En consecuencia, de aquí en adelante todas ellas (inserciones, sustituciones y eliminaciones) recibirán la denominación común de *modificaciones*.

Las relaciones de referencia se detectan en el texto de un documento `A`, el cual cita o referencia un documento (o fragmento) `B`. `A` y `B` son documentos relacionados. La relación puede consistir únicamente en la relación, pero también puede haber una modificación asociada: en estos casos, se hace una referencia al (fragmento de) documento que debe modificarse, para indicar después cómo hacerlo. Se da la situación de que existen dos enlaces heterogéneos (una referencia y una modificación) que comparten el destino del enlace (el documento o fragmento citado y modificado) pero que difieren en su origen (distintos fragmentos del mismo documento).

La referencia a un cierto elemento de un documento supone que dicho elemento y todos los nodos incluidos en él (descendientes) están afectados por la relación. Por ejemplo, la mención de la sección primera de este capítulo engloba todos sus subelementos: párrafos, figuras, ... La posibilidad de referenciar fragmentos de documentos es una necesidad si se desea en el grafo de enlaces que se obtiene a partir de estas relaciones la misma precisión que en los textos a partir de los cuales surge.

3.6.2 El grafo resultante

Los enlaces

Un *enlace* se representa mediante un arco dirigido, que parte de un vértice **origen** y alcanza un vértice **destino**. Un enlace *con tipo* es un arco etiquetado. El nodo origen es el fragmento de documento que contiene la referencia o modificación y el destino es el documento, o fragmento, referenciado o modificado. La naturaleza de la relación se representa en la etiqueta del arco. La figura 3.6 muestra un ejemplo.

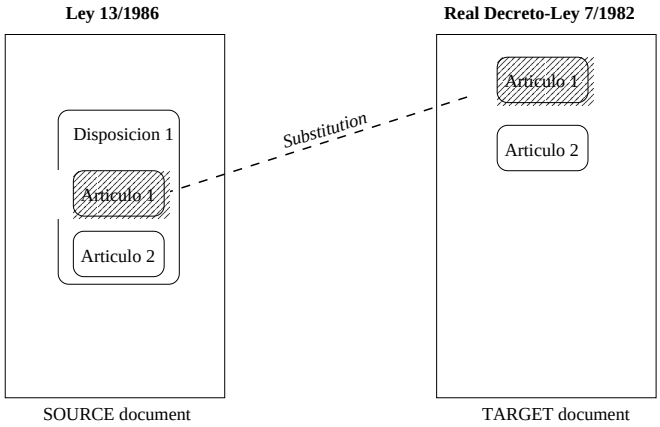


Figura 3.6: Un ejemplo de enlace con tipo. El documento origen del enlace (*Ley 13/1986*) contiene un elemento (*disposición 1/artículo 1*) que reemplaza el elemento *artículo* del documento *Real Decreto-Ley 7/1982*. Los vértices del enlaces son fragmentos de documentos (elementos de tipo *artículo*).

En el grafo resultante los enlaces relacionan fragmentos de documentos y están etiquetados según la naturaleza de la relación (cita, modificación, estructural). Los vértices del grafo (fragmentos de los documentos) pueden participar en más de un enlace, como en el caso de las modificaciones, y estos enlaces pueden ser heterogéneos. Un ejemplo puede verse en la figura 3.8.

De este modo, un *enlace con tipo* se puede definir como una tupla $< v_o, v_t, t >$ donde:

- v_o es el *node-set* en el origen del enlace
- v_t es el *node-set* en el destino del enlace
- t es el tipo del enlace

Vértices en los enlaces

Las relaciones consideradas (referencias y modificaciones) causan una granularidad mayor que la correspondiente a los documentos como unidad mínima de información. Este aspecto debe ser tenido en cuenta obligatoriamente cuando se consideran las modificaciones, relaciones que, mayoritariamente, afectan a *porciones* de documentos en vez de documentos completos (ver ejemplo de la figura 3.6).

En el caso de los documentos estructurados la estructura lógica refleja las divisiones del documento en los fragmentos respectivos; una referencia se encuentra siempre en el contenido de algún elemento localizable por su posición relativa en el árbol estructural, lo cual también se cumple con el fragmento referenciado (o modificado).

Sin embargo, también se debe tener en cuenta que la mayoría de estas relaciones involucran a varios nodos; es decir, el vértice del enlace consiste en un conjunto de nodos (*node-set*). Por ejemplo, la referencia a una definición involucra a un elemento de tipo `definition` así como a todos los párrafos que forman parte de ella. Como resultado de todo esto, se va a tener en cuenta una visión mixta de documento como un árbol de nodos, relacionados entre sí por una jerarquía de inclusión (reflejada en la estructura lógica del documento), y un conjunto de nodos, que se pueden agrupar en subconjuntos

(*node-sets*). Así, los vértices del grafo serán *node-sets*, discriminables por su posición en el árbol estructural. En general, cada *node-set* es un subárbol que se puede representar por su nodo raíz.

El grafo

La combinación de los árboles asociados a los documentos con los enlaces que afectan a sus fragmentos, teniendo en cuenta que los *node-sets* se pueden discriminar sin ambigüedad por la posición de su nodo raíz, da como resultado un grafo *etiquetado* G. Sus vértices están formados por los nodos de los documentos y sus arcos están etiquetados en función de la naturaleza de la relación que representan:

- Los arcos procedentes de los árboles documentales se corresponden con las relaciones de inclusión de la estructura lógica del documento. (La etiqueta de estos arcos *estructurales* no se muestra en los ejemplos que siguen, si bien se distinguen por ser de mayor grosor que los demás.)
- Los arcos asociados a las relaciones de *referencia* tienen asociada la etiqueta *citation*.
- Los arcos que representan relaciones de modificación tienen la etiqueta *modification*.

Cardinalidad en el grafo

Los vértices pueden ser simultáneamente origen y destino de varios enlaces. Es habitual que un documento participe en varios enlaces heterogéneos (porque referencia otros documentos, él es referenciado, está afectado por varias modificaciones, etc.) y que estos enlaces los relaciones con otros documentos. Esto sigue siendo cierto cuando se habla de fragmentos de documentos: cada fragmento puede ser referenciado desde varios puntos distintos y puede ser objeto de múltiples modificaciones. La figura 3.7 muestra un documento (o fragmento) `A` que participa en tres enlaces heterogéneos, de los cuales es el destino: `B` y `C` deben reemplazar a `A` en futuras versiones, mientras que `D` se limita a referenciarlo. En la figura 3.17 se puede ver otro ejemplo de multiplicidad; esta figura se corresponde con una porción del grafo de la figura 3.8.

Ejemplo 16. Los artículos del texto normativo español más importante ('*La Constitución*') aparecen referenciados en cientos de documentos. Son, por tanto, potenciales nodos destino de tantos enlaces *citation* como referencias a ellos existen. □

Ejemplo 17. La figura 3.8 muestra un grafo donde se dan los tres tipos de relaciones que se han comentado. Hay cuatro documentos con raíz en D, P, M y N respectivamente. Los árboles correspondientes representan las relaciones estructurales (sin etiqueta y en grueso). El subárbol de D con raíz en d_2 está referenciado (enlace c_1) en M y modificado por el subárbol cuya raíz es m_2 (enlace de modificación m_1). También está referenciado (enlace c_3) y modificado (enlace m_3) en P. El elemento d_3 de D está referenciado (enlace c_2) en el contenido de p_2. El subelemento d_{21} de d_2 in D es a su vez objeto de una modificación (m_2) y referencia (c_2) que le afectan exclusivamente a

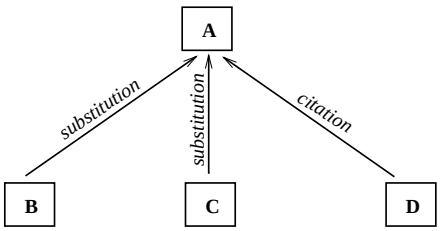


Figura 3.7: Un grafo con tres enlaces heterogéneos. Uno de los vértices, `A`, participa en tres relaciones diferentes.

él. d_{21} está asimismo afectado por una modificación que le concierne sólo a él (m_2) y por las modificaciones expresadas en los enlaces m_1 y m_3. □

Grafos parciales

A partir de este grafo se pueden obtener tres grafos parciales (ver figura 3.9), usando como criterio de selección el tipo del enlace:

- El bosque formado por los árboles asociados a los documentos, que reflejan su *estructura* lógica. Los caminos del árbol permiten acceder a los *node-sets*.
- El grafo de *referencias*, que incluye los enlaces estructurales y de relación. Por este grafo se puede "navegar" al igual que en un hipertexto.
- El grafo de *modificaciones*, donde participan los árboles anteriores y los enlaces de tipo modificación.

A partir del grafo de la figura 3.8 se obtienen los tres grafos (estructural, de modificaciones y de referencias) de la figura 3.9.

Las modificaciones a los *node-sets* afectan a su elemento raíz y a todos sus descendientes. Esta es la propiedad en la que se apoya el proceso de generación de versiones que se explica en el apartado 3.7.3. El proceso de generación de cada versión comienza con la extracción de un subgrafo a partir del grafo de modificaciones.

3.7 Generación de versiones usando enlaces

El algoritmo que se presenta en esta sección genera una versión de un documento a partir de la versión original almacenada en la biblioteca y la información que se encuentra en el grafo de modificaciones. Dada una solicitud para generar la versión de un cierto documento, `D`, se extrae un subgrafo formado por el grafo estructural de la versión disponible de `D` y los enlaces de tipo modificación que alcanzan sus nodos, que además correspondan a una fecha en el rango temporal comprendido entre la versión disponible y la fecha de la versión solicitada. A este grafo se le denomina el *grafo de versionado* (*versioning graph*).

Hay tres problemas relacionados con las versiones de documentos que se pueden considerar: la detección de los cambios que dan lugar a las versiones, la representación de

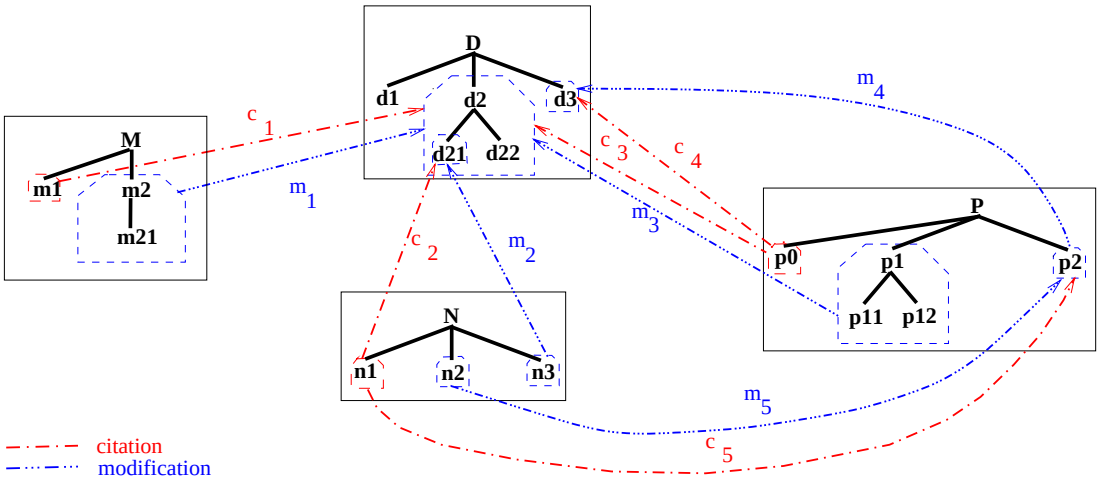
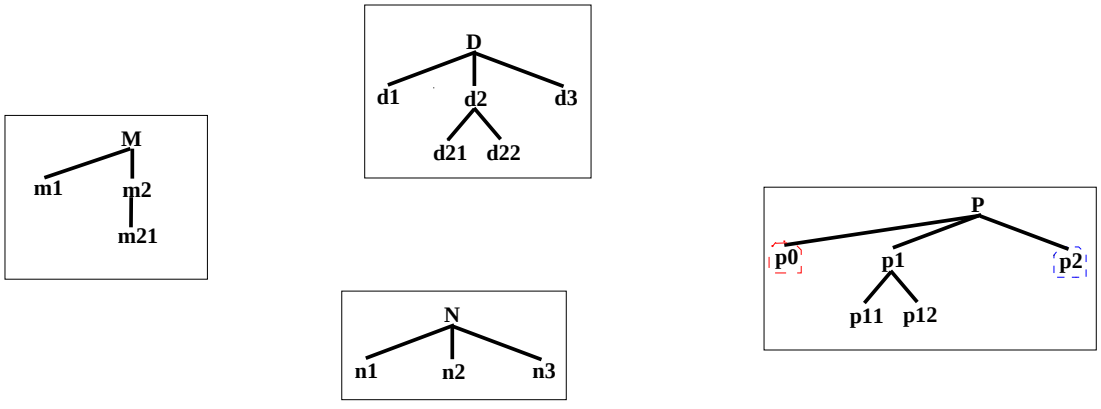
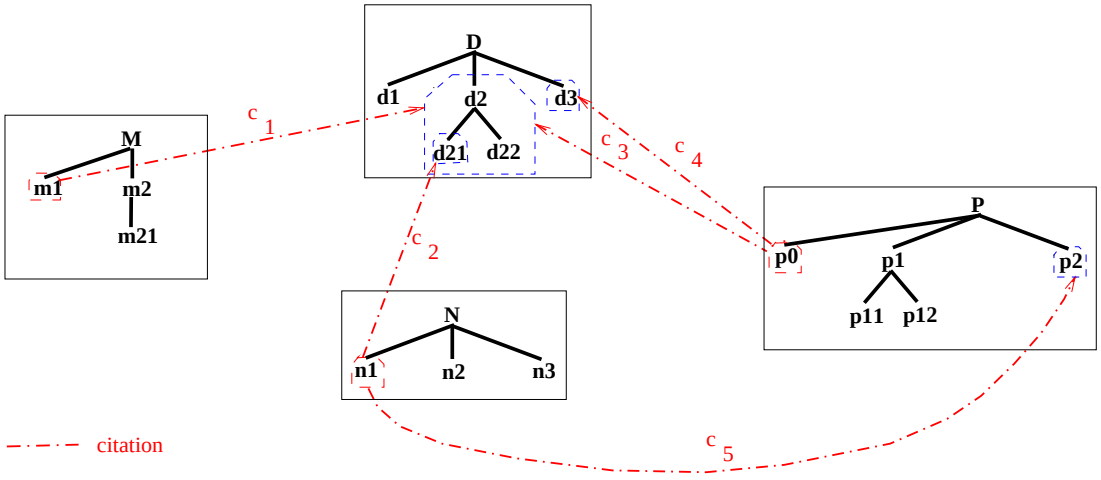


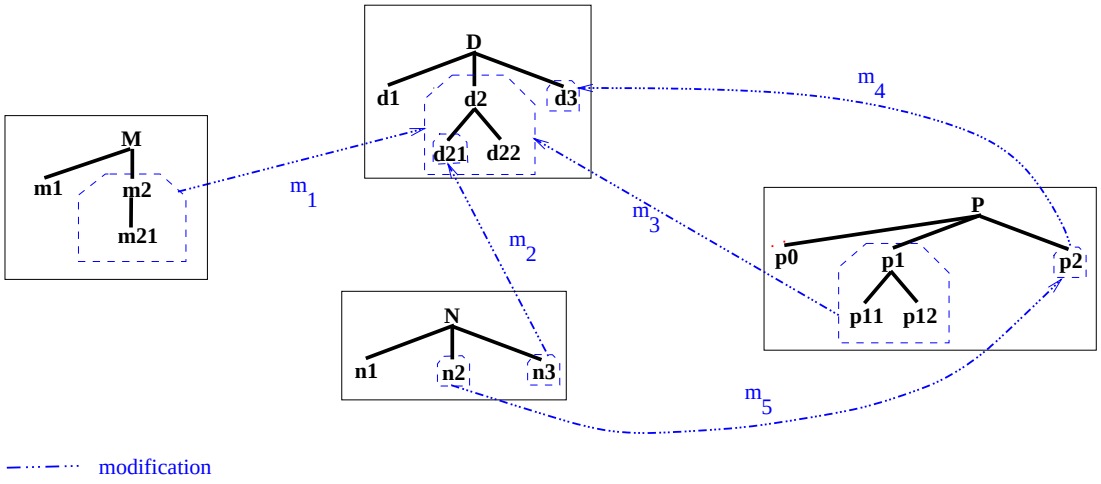
Figura 3.8: Grafo con enlaces heterogéneos: estructurales, de referencia y de modificación.



(a) Grafo estructural.



(b) Grafo de referencias.



(c) Grafo de modificaciones.

Figura 3.9: Grafos parciales obtenidos a partir de la figura 3.8.

estos cambios y las consultas que les atañen. La detección de variaciones se puede hacer por comparación de documentos [36] o a partir de las referencias que se encuentran en los textos [121, 68]. En esta tesis se trabaja con esta última opción: la existencia de versiones se puede detectar en las referencias incluidas en los textos de los documentos. En cuanto a la representación de los cambios, existen dos opciones además de la utilizada en esta tesis: almacenar todas las versiones y representar los cambios a través de los enlaces que ligan versiones [38] o representarlos como anotaciones [36, 35]. Los problemas que se pueden encontrar cuando se almacenan todas las versiones ya se han comentado en la sección 3.4. Por otro lado, este enfoque presupone la disponibilidad de una copia digital de cada versión. Esto no es algo que se pueda garantizar; cuando las versiones se deben a modificaciones que se expresan referenciando la parte afectada e indicando cómo aplicarlas no es habitual que se genere una nueva versión completa del documento resultante de dicha aplicación. En este caso se deja al usuario interesado la composición del nuevo documento. En lo que respecta a la representación de los cambios como anotaciones, se debe decir que esta solución facilita las consultas sobre la evolución histórica de los documentos, pero no es así con la generación automática de versiones. La información acerca de las relaciones de modificación no aparece como enlaces que formen parte de un grafo, sino como atributos de los nodos, a partir de los cuales se debería reconstruir el grafo. Así pues, en esta tesis la elección tomada consiste en modelar las relaciones de modificación como cualquier otro tipo de relación: con enlaces etiquetados; la generación de versiones se ve así facilitada, ya que será posible hacerlo en recorridos del grafo de relaciones.

La versión generada en cada aplicación del algoritmo debe mostrar su contenido "válido" en un cierto momento: el resultado de aplicar todas las modificaciones ocurridas desde que se creó la primera versión y el momento indicado. El supuesto de partida es que la versión inicial del documento está disponible, así como los documentos que contienen las modificaciones que le afectan. Se propone también almacenar la propuesta en una base de enlaces que contenga todos los enlaces del grafo de relaciones. La obtención de cada versión se realiza a petición del usuario, aplicando a la versión original los cambios expresados en los enlaces de modificación.

El algoritmo propuesto (apartado 3.7.3) consiste en un recorrido recursivo del árbol estructural de la versión original del documento, donde cada vez que se alcanza un nodo se analiza teniendo en cuenta que las modificaciones y citas en los documentos estructurados que afectan a un determinado nodo afectan en realidad al subárbol del cual el nodo es raíz (por ejemplo, la sustitución de un capítulo por otro supone reemplazar el capítulo y todos sus descendientes: artículos, párrafos, figuras, etc.). Una característica que ha determinado la elección de este método es que normalmente las modificaciones vienen expresadas dentro de algún documento, pero no constituyen documentos completos en si mismas. Es decir, los vértices de los enlaces son fragmentos de documentos. Si se trabaja con documentos estructurados, es posible utilizar un grafo de modificaciones cuyos vértices sean *node-sets*, tal como se explicó en la sección 3.6.2. Esto permite atacar la generación de versiones como un problema de grafos. Las modificaciones que serán finalmente aplicadas en la generación de cada versión son filtradas en el grafo utilizando los parámetros de la solicitud (en el caso de versiones históricas, la fecha).

Desde el punto de vista arquitectónico, el algoritmo explicado aquí es un método del servicio de manipulación de documentos y el escenario en que se puede encontrar es

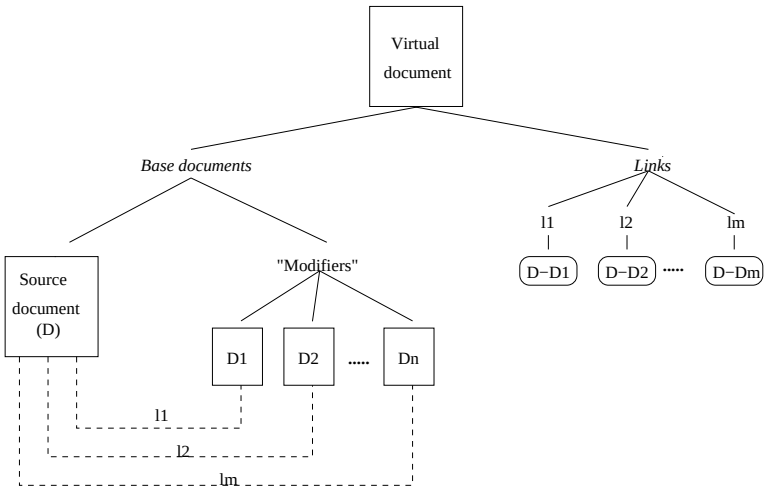


Figura 3.10: Documentos utilizados en la generación de versiones. Los documentos de entrada al algoritmo son el que será modificado y los que contienen las modificaciones.

la generación de versiones, ambos presentados en el capítulo 4. La figura 3.10 muestra los documentos que participan en la generación de versiones. Ante una petición del usuario, los documentos y enlaces se combinan para crear un documento virtual: la versión solicitada. Se obtiene aplicando las modificaciones a alguno de los documentos disponibles en las colecciones de la biblioteca (*source*). Los nuevos textos se extraen de otros documentos también en la biblioteca (*modifiers*).

3.7.1 Los árboles de las versiones

Un documento estructurado se representa en un grafo como un árbol, cuyos nodos internos se corresponden con los elementos del documento. El documento de salida (versión) resultante del proceso de actualización (generación de versiones) pertenece a la misma *clase* de documentos que la versión original (su estructura lógica se adapta a la misma gramática), pero su estructura lógica puede ser distinta.

El árbol (estructura lógica) es el resultado de reemplazar, eliminar o insertar nodos en el árbol original. Los nodos de éste último que no están afectados por ninguna modificación (no son destino de ningún enlace de este tipo) aparecerán en el árbol de salida exactamente igual que estaban en el árbol inicial.

Ejemplo 18. Un ejemplo extraído del entorno jurídico puede servir para ilustrar lo que se ha comentado. Los documentos de entrada en este caso son los textos normativos que se muestran en la figura 3.11:

- Un documento inicial, `lo2-1980.xml`, cuyo primer elemento de tipo *articulo* debe ser sustituido. El texto se puede ver en la figura 3.11(a).

- Un documento modificador, `l13-1986.xml`, que contiene el elemento que ha de reemplazar al primer *articulo* de `lo2-1980.xml`. Será el primer elemento de tipo *articulo* que se encuentre dentro del primero de tipo *disposicion*. El texto de este

documento se puede ver en la figura 3.11(b).

El documento resultante del proceso de modificación, que se puede ver en la figura 3.11(c)- es la nueva versión del documento de entrada, en la cual se ha reemplazado el *node-set* correspondiente al elemento *articulo* por el primer elemento *articulo* dentro del primero elemento *disposicion* del documento `113-1986.xml` (el resto se conserva igual que estaba). La figura 3.12 muestra el efecto de este proceso sobre el árbol documental. El enlace que representa la modificación se puede ver en la figura 3.13. □

3.7.2 El subgrafo utilizado para generar versiones

Cada versión de un documento es el resultado de resolver un subgrafo obtenido a partir del grafo de modificaciones. No existen ciclos en el grafo de modificaciones: ningún documento puede modificar documentos que se crean con posterioridad a él. En consecuencia, hay garantía de que no hay ciclos en el grafo que se utilizará para generar versiones.

El grafo que se utiliza para cada generación de versión incluye:

1. El árbol, T, asociado a la versión (inicial) disponible en las bases de la biblioteca. Esta versión es el documento inicial del proceso de generación de versiones.

2. El conjunto de enlaces de modificación, M, que alcanzan alguno de los nodos de T.

3. Los enlaces de modificación que alcanzan alguno de los vértices origen de algún enlace en M.

Los enlaces de modificación tienen además un atributo de *prioridad*: la fecha del enlace (fecha en que la modificación fue emitida). Este atributo se utilizará para resolver los conflictos que se encuentren cuando un nodo sea objeto de más de una modificación.

3.7.3 El proceso de generación de versiones de documentos

El proceso de generación de versiones de documentos opera resolviendo enlaces durante el recorrido del grafo asociado a cada versión. El algoritmo 2 genera la versión de un documento D, aplicándole las modificaciones que verifican el criterio de prioridad expresado en el argumento de entrada, la fecha (*date*). Únicamente las modificaciones cuya fecha sea previa a la indicada se aplicarán a D.

El algoritmo de generación de versiones manipula el árbol del documento inicial de modo recursivo, comenzando en su nodo raíz y tratando todos sus descendientes de modo similar, hasta llegar a las hojas (contenido del documento). Cada llamada recursiva desciende en el árbol, lo cual significa que el nivel de detalle considerado es mayor.

En cada nodo que se alcanza se puede dar una de las situaciones siguientes:

- El nodo no está afectado por una modificación (no hay enlaces de tipo modificación que lleguen a él). El recorrido debe seguir en sus descendientes, ya que éstos sí pueden ser el destino de alguna modificación.

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<doc>
<articulo id="a1"><title>Artículo Primero. </title>
<p>El referendum en sus distintas modalidades, se celebrará de
acuerdo con las condiciones y procedimientos regulados en la
presente Ley Orgánica.</p>
</articulo>
<articulo id="a2"><title>Artículo Segundo. </title>
<p>Uno. La autorización para la convocatoria de consultas populares
por vía de referendum en cualquiera de sus modalidades, es competencia
exclusiva del Estado.</p>
</articulo>
</doc>
```

(a) Documento original en el ejemplo: *lo2-1980.xml*

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<doc>
<p>Ley 13/1986, de 14 de Abril de 1986, de Fomento y Coordinación
General de la Investigación Científica y Técnica</p>
<p>Don Juan Carlos I,Rey de España.</p>
<disposicion id="da"><title>DISPOSICIONES ADICIONALES. </title>
<p><a>Undécima.</a>
 1. Quedan modificados los artículos 1.°, 4.° y 8.° de la Ley
Orgánica 2/1980, de 30 de abril , que quedarán redactados en la
forma siguiente :</p>
<articulo id="da111"><title>Artículo 1.</title>
<p>Con la denominación de Instituto de Astrofísica de Canarias se crea
un Consorcio Público de Gestión , cuya finalidad es la investigación
astrofísica .</p>
<p>El Instituto de Astrofísica de Canarias estará integrado por la
Administración del Estado, la Comunidad Autónoma de Canarias la
Universidad de La Laguna y el Consejo Superior de Investigaciones
Científicas .</p>
</articulo>
<articulo id="da112"><title>Artículo 4.</title>
<p>El Consejo Rector estará integrado por el Ministro de Educación y
Ciencia , que actuará como Presidente; un Vocal en representación de la
Administración del Estado, que será nombrado a propuesta del
Ministerio de la Presidencia , y tres Vocales más en representación de
cada una de las restantes Administraciones públicas y Organismos que
se relacionan en el artículo 1.° Formará parte del Consejo Rector,
asimismo, el Director del Instituto , que será miembro nato.</p>
</articulo>
</disposicion>
</doc>
```

(b) Documento modificador: *l13-1986.xml*

```
<?xml version="1.0" encoding="ISO−8859−1"?>
<doc>
<articulo id="da111"><title>Artículo 1.</title>
<p>Con la denominación de Instituto de Astrofísica de Canarias se crea
un Consorcio Público de Gestión, cuya finalidad es la investigación
astrofísica .</p>
<p>El Instituto de Astrofísica de Canarias estará integrado por la
Administración del Estado, la Comunidad Autónoma de Canarias la
Universidad de La Laguna y el Consejo Superior de Investigaciones
Científicas .</p>
</articulo>
<articulo id="a2"><title>Artículo Segundo. </title>
<p>Uno. La autorización para la convocatoria de consultas populares
por vía de referendum en cualquiera de sus modalidades, es competencia
exclusiva del Estado.</p>
</articulo>
</doc>
```

(c) Documento modificado: nueva versión de *lo2-1980.xml*

Figura 3.11: Generación de versiones. Documentos de entrada y salida.

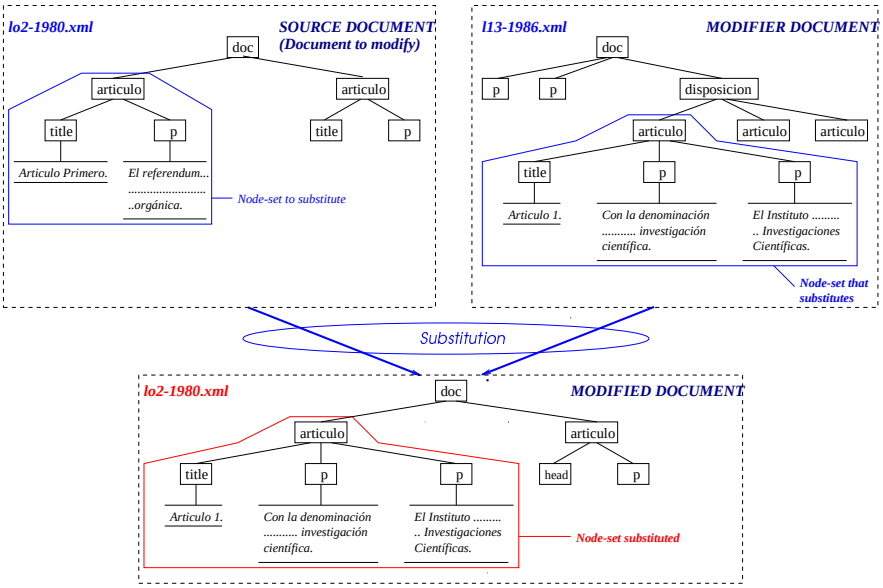


Figura 3.12: Sustitución de elementos usando enlaces. El primer elemento de tipo *articulo* del documento origen (*lo2-1980.xml*) se sustituye por el primer elemento *articulo* incluido en el primer elemento de tipo *disposicion* dentro del documento modificador. Como resultado se obtiene una nueva versión de *lo2-1980.xml*.

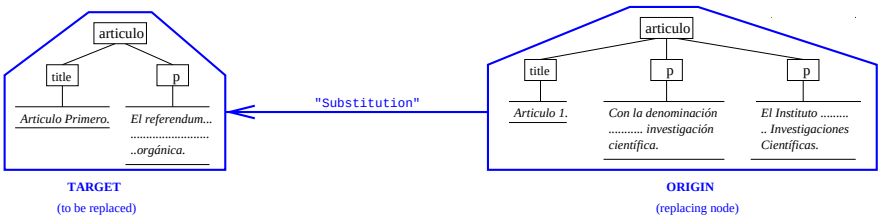


Figura 3.13: Enlace de modificación (*substitution*). El origen (ORIGIN) deberá reemplazar al destino (TARGET) cuando se genere una nueva versión del documento inicial. El vértice ORIGIN es un subárbol formado por el elemento *articulo* y todos sus descendientes. El vértice TARGET es el subárbol del documento modificador que tiene como raíz el primer elemento *articulo* que forma parte del primer elemento *disposicion*.

- El nodo está afectado por una modificación. Es el destino (raíz del *node-set*) de algún enlace modificador. En este caso debe aplicarse la modificación sustituyendo este subárbol por el origen del enlace.

Los casos base que garantizan la finalización de la recursión son, por tanto:

- El nodo que se está evaluando es el nodo vacío. Esto quiere decir que se ha descendido hasta el nivel más bajo en la jerarquía del documento (incluso el contenido ha sido visitado), sin que haya habido modificaciones que aplicar.

- El nodo está afectado por una modificación. La aplicación de esta modificación supone la sustitución de todo el subárbol (contenido incluido) con raíz en el nodo, con lo cual no es necesario seguir buscando modificaciones que afecten a sus descendientes.

Las llamadas recursivas tienen lugar cuando el nodo que se está evaluando no es ni el nodo nulo ni destino de ningún enlace de modificación. Estos nodos se copiarán en la nueva versión tal cual están, pero se debe continuar el proceso en sus descendientes, que cabe la posibilidad de que sí estén afectados por alguna modificación.

Un ejemplo puede verse en la figura 3.12, donde la única modificación considerada afecta al primer elemento de tipo `articulo` del documento original, lo cual significa que su aplicación supondrá la sustitución del elemento y todos sus descendientes. El resto de elementos del documento no se modifican, con lo cual permanecen intactos después de ser visitados durante el recorrido recursivo.

Ejemplo 19. El documento D de la figura 3.14 está compuesto por tres elementos (d_1, d_2,d_3); el elemento d_2 está formado a su vez por dos elementos (d_{21},d_{22}). En D se expresan dos modificaciones, m_1 y m_2, cuya aplicación generará una nueva versión de D. Esta se obtiene como sigue:

- El elemento p_1 (incluidos sus descendientes p_{11} y p_{12}) reemplaza al elemento d_2 de D. Esta es la aplicación de la modificación m_1.

- El elemento p_2 sustituye al elemento d_3 de D como resultado de aplicar la modificación m_2.

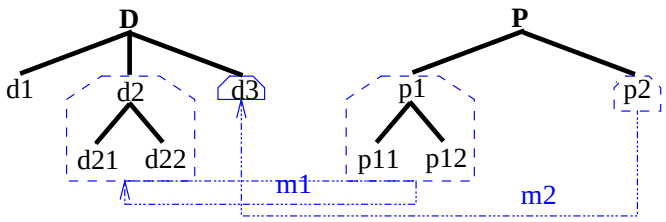


Figura 3.14: Grafo usado para la generación de una versión.

El algoritmo de generación de versiones comienza visitando el nodo D. Una modificación a este nodo supondría la sustitución del documento *completo* por otro. Dado que en este ejemplo no está afectado por ninguna modificación, el algoritmo continuará, explorando sus descendientes, d_1, d_2 y d_3, que pueden ser a su vez objeto de alguna modificación. El nodo d_1 permanece inalterado, ya que no es alcanzado por ninguna modificación. A d_2 le reemplaza el subárbol con raíz en p_1, mientras que d_3 es sustituido por p_2, con lo cual finaliza el recorrido recursivo. □

3.7.4 Versionando nodos

Dado un nodo afectado por modificaciones, éstas pueden ser varias y pueden darse situaciones de *solapamiento* entre modificaciones o de transitividad. Así pues, a la hora de aplicar una modificación a un nodo, se distinguen tres situaciones posibles:

- *Caso simple.* El nodo está afectado por un único enlace. La resolución se limita a sustituirlo por el origen del enlace.
- *Modificaciones transitivas.* Existe una secuencia de modificaciones históricas, tal que el origen de la modificación que afecta al nodo en curso, está afectado a su vez por otra modificación (es destino de un enlace modificador). Esta situación puede repetirse de modo transitivo con los sucesivos orígenes de los enlaces. En este caso la resolución de la modificación supone la aplicación de la secuencia de modificaciones históricas representada por la cadena de enlaces modificadores (ver figura 3.15).

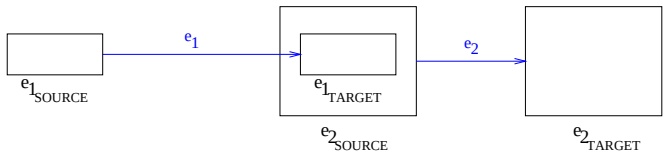


Figura 3.15: Modificaciones transitivas. $e_{1_T} \subset e_{2_S}$.

Pueden distinguirse a su vez dos casos incluidos en esta categoría. Dados dos enlaces e_1 y e_2 que participan en una transitividad, puede ocurrir una de las dos situaciones que siguen:

– $e1_T \subseteq e2_S$ *(e_1 modifica parcial, o totalmente, e_2). Además la fecha de e_1 es más reciente que la de e_2.* La figura 3.16 muestra una situación donde $e1_T \subset e2_S$.
Se debe aplicar e_2, si bien e_1 debe aplicarse previamente sobre el vértice origen de e_2.

Ejemplo 20. El ejemplo de la figura 3.16 muestra una porción del grafo utilizado para generar la versión del documento D. La sustitución del subárbol con raíz en d_3 supone reemplazarlo por el subárbol cuya raíz es p_2; dado que éste debe ser sustituido a su vez por n_2, finalmente d_3 será reemplazado por n_2. □

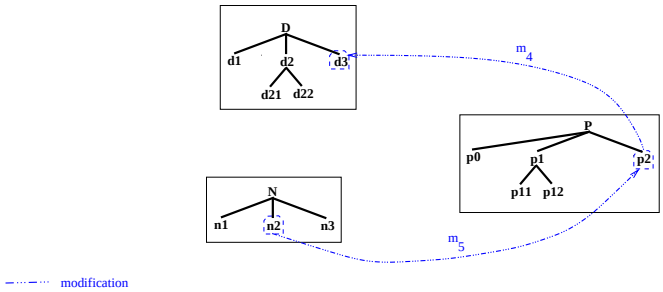


Figura 3.16: Modificaciones transitivas durante la generación de versiones.

– $e1_T \subset e2_S$ *(misma figura que en el caso anterior), pero e_2 es más reciente que e_1.*
Esta es una situación imposible: un documento no puede modificar otro posterior a él.

- *Modificaciones solapadas.* Esta es una situación de conflicto debida a que existe más de una modificación que afectan a un nodo. Pueden distinguirse a su vez:

  – *Solapamientos totales.* Dos o más modificaciones coinciden *exactamente* en sus destinos (mismo subárbol en el destino). Este es el caso que se puede ver en la figura 3.17. Se utiliza el criterio de prioridad: el enlace que se resuelve es el más reciente.

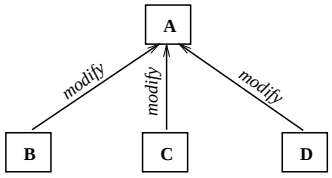


Figura 3.17: Solapamientos totales: los destinos coinciden.

  – *Solapamientos parciales.* En este caso los destinos de las modificaciones son tales que uno de ellos incluye al otro (el segundo es subárbol del primero). Afectan al mismo documento, pero distintos fragmentos (esta situación se

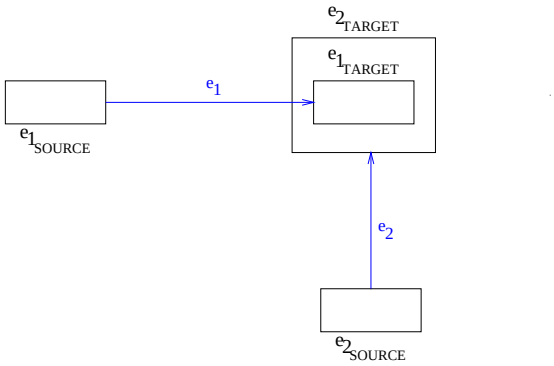


Figura 3.18: Solapamiento en las modificaciones. $e_{1_T} \subset e_{2_T}$.

muestra en la figura 3.18). La decisión sobre cuál se debe aplicar depende de las fechas de los enlaces. En caso de que el enlace que afecta al mayor fragmento sea el más reciente, éste se aplica, anulando la otra modificación. Si fuese al revés (el enlace que afecta al fragmento más pequeño es más reciente), podría deberse a un caso de incoherencia en las modificaciones, lo cual no puede ser detectado de modo automático por una aplicación, con lo cual dicha modificación ($e_{1_T} \subseteq e_{2_T}$ y e_1 es más reciente que e_2) no se resuelve y se incluye en una lista de conflictos irresolubles, que se devolverá en la salida.

Ejemplo 21. Un ejemplo de una situación semejante es una modificación a una obra de teatro donde, a una modificación a una cierta escena X, le sigue una modificación a todo el acto que incluye esta escena. La última modificación supone *desestimar* la primera (la que afectaba únicamente a la escena). □

Ejemplo 22. La figura 3.19 muestra otro grafo extraído del grafo en la figura 3.9(c). En este caso la modificación a d_{21} se ignora y se utiliza únicamente la que afecta a d_2. □

El algoritmo

El algoritmo 2 versiona un documento `D`, teniendo en cuenta la fecha recibida en los argumentos de llamada como fecha de la versión de salida. Le aplica todas las modificaciones cuya fecha está en el intervalo temporal comprendido entre la fecha de creación de D y la recibida en los argumentos de invocación. Este algoritmo hace uso del algoritmo 3 para resolver cada nodo, aplicando los criterios hasta aquí explicados. En las líneas 2 a 5 se seleccionan todos los enlaces con destino en el nodo en curso y se resuelven los conflictos de solapamiento: O_L es el *node-set* que reemplazará a este nodo. Si no hay solapamientos parciales (inclusiones), el nodo será sustituido directamente por O_L (o el resultado de resolver las modificaciones transitivas que afectan a O_L).

Si hay conflictos debidos a solapamientos parciales entre el nodo y sus descendientes,

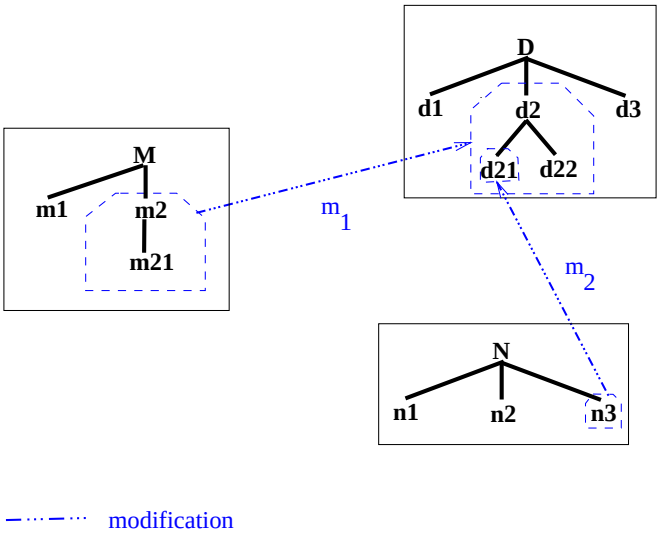


Figura 3.19: Solapamiento en las modificaciones.

se contemplan las dos situaciones posibles descritas en la página 66: las modificaciones a sus descendientes que son más recientes que las que afectan al nodo en curso no se pueden resolver, mientras que las que son anteriores quedan canceladas. La inducción de la recursión (nodos que se copian tal cual se encuentran en el documento inicial) se corresponde con las llamadas recursivas comprendidas entre las líneas 18 y 20.

Algoritmo 2 Algoritmo de aplicación de modificaciones a documentos.

Entradas: D: document, date: criteria
Salidas: A new version of D. A list of conflictive node couples.
1: Let d be the root node of D
2: $Modify(d, date)$

Documentos entrada y salida en la generación de versiones

Los documentos de entrada en la generación de versiones provienen de las bases de datos de la biblioteca, y pertenecen a alguna de las clases soportadas en ella. Los enlaces proceden de las bases de enlaces. De este modo, los datos que participan en el proceso de generación de una versión son:

- Un documento que será actualizado (versión inicial disponible en las bases de documentos).
- Un conjunto de documentos (modificadores) que contienen modificaciones que afectan al primero.
- El conjunto de enlaces que expresan estas modificaciones: conectan el documento inicial con los modificadores.

Algoritmo 3 Algoritmo de modificación en nodos.

Function $Modify(n : node, date : criteria)$

```
 1: if n is not null then
 2:   if n is the target vertex of some modification link then {n is the root of some node set in a link's
      target}
 3:     Let M_L be the list of modification arcs with n in the target's root
 4:     Let L be the link in M_L with more recent date attribute
 5:     Let O_L be the origin node of link L
 6:     Let M_D be the list of descendants of n that are the target in some modification link
 7:     if M_D is empty then {There is no inclusion overlapping}
 8:       n ← Modify(O_L, date) {Treat transitive modifications before applying O_L}
 9:     else {There is inclusion overlapping between n and its descendants}
10:       if there is some node c in M_D with a modification date more recent than L's date then
11:         Mark n with an Abnormal actualizations conflict flag and do not version n
12:         Add the pair (n, c) to the list of conflicts
13:       else {All modifications to children of n are previous to L and, thereby, cancelled by L}
14:         n ← Modify(O_L, date)
15:       end if
16:     end if
17:   else {n is kept untouched. Its descendants may, however, be affected by some modification}
18:     for all c ← child(n) do
19:       Modify(c, date)
20:     end for
21:   end if
22: end if
```

El documento inicial y los modificadores pueden pertenecer a clases diferentes (asociados a distintas DTD). Sin embargo, la clase de los documentos no tiene ninguna influencia en el proceso de generación de versiones.

3.8 Implementación del grafo de relaciones en una base de enlaces

Campos y su semántica

Los enlaces se pueden modelar como registros con varios campos donde se guarda su información. En la definición de estos campos influyen las condiciones que se impone a la base de enlaces que se obtenga, para reflejar correctamente la información presente en el grafo de relaciones:

- Se debe poder direccionar fragmentos internos de los documentos.
- Ha de ser posible realizar consultas acerca de las relaciones.
- Las consultas pueden realizarse multidireccionalmente: se puede preguntar por los documentos citados desde uno dado, o por los documentos que lo citan. Por ejemplo, se puede preguntar en dos sentidos opuestos acerca de las modificaciones: *¿Qué documentos modifican este documento?* o *¿Qué documentos son modificados desde este documento?*.

- El grado del grafo ha de verse reflejada convenientemente.
- Las relaciones han de ser accesibles independientemente de los documentos implicados, único modo de explotar convenientemente la multidireccionalidad y grado del grafo.

Los campos que se obtienen se pueden presentar en función de qué porción del enlace modelan: origen, destino o arco. Los campos asociados al *origen* del enlace son:

- **Identificador del documento**: identificador *lógico* (ver capítulos 2 y 4).
- **Dirección física**. Ubicación de la copia digital del documento disponible en la biblioteca.
- **Localizador interno** dentro del documento, que discrimina sin ambigüedad el fragmento que contiene la referencia o modificación.
- **Tipo del enlace**: uno entre (*citation, modification*).

  El tipo del enlace se modela en el nodo origen: dadas las relaciones consideradas en esta tesis (referencias y modificaciones que se encuentran en documentos), su tipo viene determinado por el contenido del vértice origen del enlace. Dada esta consideración, es lo más natural expresarlo en el nodo origen.
- **Clase del documento**. Clase (ver capítulo 2) a la cual pertenece el documento.
- **Fecha**. Fecha de creación del documento. Tal como se explicó en la sección 3.7, en el caso de múltiples modificaciones que afectan a un mismo nodo, este atributo es el criterio utilizado para resolver estos conflictos.

Los atributos del *destino* del enlace son:

- **Identificador del documento** (mismo significado que para el nodo origen)
- **Dirección física** (mismo significado que para el nodo origen)
- **Localizador interno** (ídem)

Representación con XLink

Todos los requisitos citados son obtenibles con XLink y XPointer, que proporcionan las herramientas adecuadas para su implementación. Por ello la base de enlaces es una base de "xlinks".

3.9 Aplicación de la propuesta a los documentos del entorno jurídico

Los documentos jurídicos contienen frecuentes citas y modificaciones que afectan a fragmentos de los documentos. Por ejemplo, los textos normativos y la jurisprudencia

son referenciados a menudo. Las modificaciones a un documento se encuentran en otros documentos legislativos posteriores donde se cita y modifica el fragmento afectado.

En el ejemplo que sigue, la ley española *Ley 13/1986* cita el cuarto *artículo* de la *Ley 11/1977*:

```
A los efectos de su gestión económico-financiera los Organismos a que se
refiere el artículo 13 de la presente Ley se entenderán incluidos en el
apartado b) del párrafo primero del artículo 4 de la Ley 11/1977,
General Presupuestaria, de 4 de enero.
```

Este mismo documento (*Ley 13/1986*) incluye una modificación a la *Ley Orgánica 2/1980*:

```
[...] Quedan modificados los artículos 1°,  2°, 4° y
 8° de la Ley Orgánica 2/1980, de [...] que quedarán
redactados en la forma siguiente:

Artículo 1.

Con la denominación de Instituto de Astrofísica de Canarias se crea un
Consorcio Público de Gestión, cuya [...] y el Consejo Superior
de Investigaciones Científicas.

Artículo 4.

El Consejo Rector [...]
```

Así pues, es posible obtener una nueva versión del documento *Ley Orgánica 2/1980* aplicando las modificaciones expresadas en la *Ley 13/1986*. Para ello se puede aplicar el algoritmo 2. La nueva versión del documento *Ley Orgánica 2/1980* que se obtiene es la que aparece en la figura 3.11(c).

También se puede consultar estas relaciones utilizando el grafo de enlaces que se presentó en la sección 3.6.

3.10 Discusión

Una de las formas más populares de modelar las relaciones es el hipertexto. Los enlaces representan las relaciones y se puede navegar por el grafo que se obtiene. Cada paso de la navegación supone atravesar un enlace partiendo de su origen hasta alcanzar su destino; el número de nodos que comparte el origen es el aspecto que introduce complejidad en dicho recorrido. Los nodos del grafo son documentos, y la consecución de mayor granularidad en el grafo requiere forzosamente la evolución a métodos y estándares capaces de trabajar con documentos estructurados, como es el caso de XML.

Sin embargo, los grafos considerados en esta tesis no se limitan al hipertexto navegacional que se ha utilizado tradicionalmente para modelar las relaciones. En este caso se aporta una solución que permita obtener más ventajas que la navegación: la consulta y la generación de nuevos documentos (versiones). En ambos casos tiene gran importancia la consideración de una unidad mínima de información más pequeña que

el documento (que es la unidad más habitual en los hipertextos). El subgrafo de referencias que se ha presentado en este capítulo puede ser utilizado con los mismos fines que los grafos hipertexto: cada enlace de tipo *citation* permite avanzar un paso en la navegación entre documentos. Sin embargo, el grafo de modificaciones se utiliza para *automatizar* tareas cuya realización queda en otros casos bajo la responsabilidad del usuario; para llevarlas a cabo éste se ve obligado a navegar por el grafo hipertexto, corriendo el siempre inevitable riesgo de "perderse" si el grafo es complejo [92]. Así, en esta tesis se propone la utilización del grafo de modificaciones en procesos de composición dinámica de documentos, cuyo recorrido del grafo es automático; la generación de versiones es un buen ejemplo. Esta aplicación, unida a la posibilidad de realizar consultas sobre las relaciones aprovechando el grafo de relaciones, demuestra cómo los grafos basados en la semántica de las relaciones que representan permiten ampliar las posibilidades de explotación respecto a aquellos que utilizan únicamente las relaciones que el usuario explicita mediante enlaces. Esto se traduce en ventajas tanto en lo que a posibilidades de ampliación de funcionalidades de la biblioteca se refiere, como a un mejor aprovechamiento de la información implícita en los datos disponibles.

Las modificaciones no son exclusivas de los documentos, pero en su caso tienen el interés añadido de que se encuentran expresadas dentro de otros documentos (que no se deben fragmentar) y que, en la mayoría de los casos, no afectan a la totalidad del documento. Los elementos involucrados en una relación no son objetos de primer nivel, ya que en su mayoría no constituyen una unidad semántica con entidad suficiente para ser considerados documentos. Esta es una diferencia importante respecto a otros entornos donde también se ha estudiado el problema de las versiones, como las actualizaciones de software, donde siempre se trabaja a nivel de fichero [41]. La segunda particularidad relevante de los documentos es que es posible extraer las relaciones del texto. Además, si se manipulan documentos estructurados (como ocurre en esta tesis), no es necesario que los elementos implicados en una relación dispongan de ningún tipo de identificador; son localizables por su posición en el árbol documental. En esta tesis se ha aprovechado esta propiedad, de modo que los *node-sets* implicados en las relaciones se identifican por la posición de su nodo raíz. La ventaja no se limita a la independencia de identificadores, cuyo criterio de creación es arbitrario (el que los usuarios arbitrariamente elijan) y por tanto difícilmente controlable; dicha independencia supone en realidad la posibilidad de aplicar las propuestas presentadas en esta tesis a cualquier documento estructurado, sea cual sea su estructura o clase.

Las relaciones de referencia y modificación consideradas en esta tesis provienen del interior de los textos de documentos. Dan lugar, por tanto, a un grafo de relaciones con un alto nivel de granularidad. Esta granularidad es precisamente la que permite plantear tratamientos avanzados sobre los documentos, como la composición de versiones. Por otro lado, en esta tesis la integridad de los documentos que contienen las modificaciones no se altera: nunca se fragmentan en unidades más pequeñas para facilitar los tratamientos ni se modifican para representar las relaciones. Los fragmentos se direccionan en base a la estructura arborescente y la información sobre las modificaciones se almacena en bases especialmente dedicadas. Así, la inclusión (o modificación) de relaciones no afecta en absoluto a los documentos ligados por ellas. Otra ventaja es que se facilitan las consultas sobre relaciones y los recorridos que requieren la reconstrucción del grafo de relaciones.

El segundo problema que se ha presentado en este capítulo es el de las versiones de documentos. Se ha hecho así ya que es un problema que suele ir ligado a la existencia de relaciones. En cuanto a las versiones de un documento se ha optado por generarlas automáticamente, evitando de este modo los problemas de mantenimiento de enlaces a que da lugar la coexistencia de varias versiones de un mismo documento [38, 101]. La generación de versiones propuesta en este capítulo va un paso más allá de la composición de documentos basada en la expresión explícita de los enlaces "estructurales" que ligan el documento marco y aquellos que actúan como sus componentes [109, 8]. En nuestro caso, las reglas de composición se *deducen* a partir de enlaces semánticos "*no* estructurales" (enlaces de modificación). La complejidad del recorrido en el cual ocurre esta composición (generación de una versión) viene dada por la multiplicidad de enlaces que comparten un nodo destino, a diferencia del caso del hipertexto, donde -como se ha mencionado previamente- el motivo de complejidad se encuentra en la multiplicidad en el origen de los enlaces.

Una propuesta para generar automáticamente las versiones de los documentos es la presentada en [13], si bien no se apoya en la explotación de los enlaces. La utilización de enlaces propuesta en esta tesis tiene frente a la especificación de las reglas de obtención particulares de cada versión la ventaja de su generalidad. Otra ventaja adicional del método basado en enlaces es que éstos pueden ser útiles para otras funciones que no sean generación de versiones, en las que sea necesario realizar interrogaciones sobre dichas relaciones.

La aplicación del algoritmo generador de versiones se ve limitada en el caso de modificaciones históricas incoherentes. La compleción que permita resolver este problema pasaría por la interacción con el usuario, que indicaría a la aplicación qué hacer con estos conflictos.

Por último, el tratamiento automático de las relaciones sería total con la integración de un método de detección de relaciones (citas) en los documentos de partida. La detección de referencias es objeto de varios estudios [31], si bien únicamente en entornos cuyo lenguaje es rígido en su sintaxis y vocabulario se han obtenido resultados satisfactorios [121].

4

Arquitectura para la explotación de relaciones

Índice General

Este capítulo aporta una propuesta arquitectónica para integrar en las bibliotecas digitales funcionalidades orientadas a la explotación de relaciones. Las nuevas funcionalidades se incorporan con aquellas que son "clásicas" en las bibliotecas digitales (consulta, recuperación de documentos y navegación), probando de este modo que las funcionalidades de explotación de enlaces se pueden integrar en una biblioteca digital sin afectar la operación de las ya existentes.

Las bibliotecas digitales ofrecen a sus usuarios funcionalidades que permiten explotar la información que estos sistemas albergan. Las tareas necesarias para responder correctamente a las solicitudes de los usuarios son responsabilidad del sistema. La arquitectura de la biblioteca debe facilitar la distribución de tareas entre servicios que cooperan (o componentes, si se trata de una fase de implementación) de modo que cumpla además una serie de propiedades especificadas en las líneas de actuación de la ingeniería del software (arquitectura abierta, federación, integrabilidad e interoperabilidad). En este tipo de sistemas estas propiedades son especialmente importantes, debido en muchos casos al modo en que se construyen las bibliotecas digitales: integrando datos y sistemas ya existentes, que además son heterogéneos, para construir así una única biblioteca en la cual los usuarios no sean conscientes de dicha variedad.

Si se consideran las funcionalidades mínimas en una biblioteca (recuperación, consulta y navegación), es posible obtener un modelo de referencia para las bibliotecas digitales, apreciable en cualquiera de ellas. Tres modelos reciben una atención especial en esta tesis: el modelo de referencia para los servicios básicos que se acaba de comentar, un modelo para los enlaces entre referencias (*reference linking*) y una arquitectura para la manipulación de documentos.

La integración de colecciones y la cooperación de servicios implica la utilización de un protocolo que controle la interacción entre los servicios participantes. Los más clásicos son los protocolos de Recuperación de Información, basados en un modelo cliente/servidor, como Z39.50. Estos protocolos están pensados para situaciones de distribución de las colecciones y su objetivo principal es facilitar las *consultas* entre colecciones remotas. Otros protocolos más recientes fueron diseñados con el objetivo de hacer cooperar múltiples servicios. Además de la consulta y recuperación de registros se consideran otras funcionalidades, lo cual da como resultado protocolos *orientados a los servicios*. Algunos ejemplos son *Dienst* y los protocolos definidos en el Proyecto de Bibliotecas Digitales de Stanford (*Stanford Digital Library Project*), ambos definidos expresamente para ser utilizados en entornos de bibliotecas digitales.

La arquitectura que se propone en este capítulo, que se ocupa de la integración de la explotación de relaciones, está definida utilizando el modelo de servicios, de modo que se detallan los servicios y el protocolo de interacción entre ellos (a través de sus interfaces). A partir de esta visión funcional y abstracta de la biblioteca se evoluciona en la fase de desarrollo a un conjunto de componentes que implementan los servicios conseguidos en la etapa anterior a través de la invocación mutua.

El prototipo presentado en el capítulo 5 es una implementación de los servicios para tratamiento de enlaces, basada en la arquitectura de componentes que se muestra en el apartado 4.2.6.

4.1 Protocolos y arquitecturas en las bibliotecas digitales

Las bibliotecas digitales pueden ofrecer funcionalidades diversas [88, 90, 89, 10, 11], implementadas habitualmente por un conjunto de servicios autónomos, que cooperan. En cualquier caso, la arquitectura del sistema se ve influenciada por las funcionalidades ofertadas. Dado que entre los objetivos de esta tesis se encuentra la manipulación de documentos y de relaciones derivadas de las referencias entre documentos, se presta en este capítulo especial atención a las referencias siguientes: un modelo de referencia para servicios básicos (presentado en la subsección 4.1.1), una arquitectura de referencia para los enlaces entre referencias (comentada en la subsección 4.1.2), y una propuesta de arquitectura orientada a la manipulación de documentos (ver subsección 4.1.3).

La interacción entre los servicios está regulada por protocolos especialmente diseñados para ser utilizados en las bibliotecas digitales. Dos ejemplos entre los más completos son los descritos en esta sección. El primero es *Dienst* [43], el protocolo definido en Cornell para ser utilizado en las bibliotecas NCSTRL y la biblioteca de Informes Técnicos de ERCIM. El segundo conjunto de protocolos forman parte del proyecto de Bibliotecas Digitales de Stanford [104]. Estos protocolos están descritos en términos de sus servicios e interfaces, en el caso de Dienst, y en el caso del proyecto de Stanford, mediante sus componentes y respectivas interfaces. La existencia de estos protocolos orientados a los servicios no significa, sin embargo, que no se utilicen otros protocolos de más bajo nivel en las bibliotecas digitales. Los componentes que implementan los servicios contemplados en ellos interaccionan mediante protocolos como HTTP [44], protocolos que forman parte de CORBA [63, 122] y protocolos de Recuperación de Información (como Z39.50) [104, 9]. Z39.50 es un protocolo de Recuperación de Información que considera los servicios de búsqueda y recuperación, y que ha inspirado algunas decisiones (en lo que a semántica se refiere, principalmente) de protocolos de bibliotecas digitales posteriores.

4.1.1 Modelo de referencia básico para bibliotecas digitales

Las funcionalidades básicas en las bibliotecas digitales son la búsqueda y recuperación de documentos, junto a la navegación en las colecciones de la biblioteca. Los usuarios pueden explotarlas a través de las interfaces de los servicios que facilitan su interacción con el sistema de la biblioteca. Estas funcionalidades siempre deberían estar presentes, incluso aunque la biblioteca esté distribuida, haya heterogeneidad, o cualquier otra característica adicional de la biblioteca o cuáles son los servicios ofertados. El conjunto de servicios responsables de proporcionar estas funcionalidades son:

- *Servicios de acceso a colecciones*, que proporcionan acceso a las colecciones de la biblioteca(recuperación de documentos).
- *Servicios de interfaz de usuarios*, que proporcionan a los usuarios interfaces al resto de servicios (y datos) de la biblioteca.
- *Servicios de búsqueda*, que permiten buscar en los documentos de la biblioteca y obtener el conjunto de identificadores de documentos que responden a la consulta.
- *Servicios de nombres*, que asocian nombres significativos a los objetos digitales, y permiten a los usuarios acceder a trabajos intelectuales (en vez de objetos di-

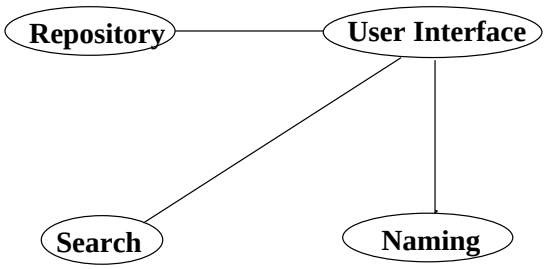


Figura 4.1: Servicios básicos en las bibliotecas digitales.

gitales) [10]. Este servicio permite abstraerse de la implementación de los documentos (abstractos) en objetos digitales.

Estas funcionalidades básicas dan como resultado el modelo de referencia de la figura 4.1. En la biblioteca más básica el servicio de interfaz de usuario interacciona con los restantes. Transmite las solicitudes de recuperación de documentos al servicio de colección, pasa las consultas de usuario a los servicios de búsqueda y utiliza el servicio de nombres para obtener la equivalencia entre los identificadores de las obras intelectuales y direcciones de objetos digitales.

Este modelo se expande en cada biblioteca en función de sus necesidades particulares. La heterogeneidad da lugar la presencia de *Translation services* [98, 100, 91]. La distribución también influencia la arquitectura de las bibliotecas digitales; nuevos servicios (que se ocupan de la distribución de las consultas y la redirección de los datos y mensajes) se integran en el sistema [45, 122, 22, 61].

Algunos casos donde se pueden encontrar estas funcionalidades básicas son la *Networked Computer Science Technical Reports Library (NCSTRL)* [44], la *Ercim Technical Reference Digital Library* [22], o *Networked Digital Library of Theses and Dissertations (NDLTD)* [95].

En cada biblioteca, un conjunto de componentes implementa estos servicios. En este conjunto están incluidos los componentes software (o agentes), así como los repositorios de datos. Los datos que califican a otros datos y los datos sobre el sistema (metadatos) también se incluyen en el diseño de la arquitectura. Los servicios de acceso a colecciones (*Collection services*) son ofertados por agentes de interfaz de colecciones (*Collection Interface agents*) [24], mientras que los servicios de interfaz de usuario están implementados en la mayoría de los casos por agentes de interfaz de usuario (*User Interface agent*) [24, 11]. Es posible encontrar variaciones sobre este modelo, y el resto de la introduccion de este capítulo se dedica a los protocolos utilizados en la base de estas arquitecturas, así como en dos modelos (subsecciones 4.1.2 y 4.1.3) que han sido utilizados como referencia durante la modelización de la arquitectura de esta tesis.

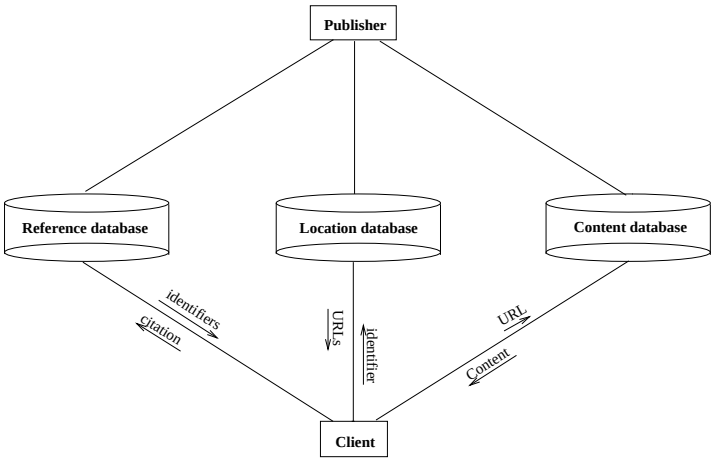


Figura 4.2: Un modelo de componentes para las referencias entre documentos en revistas.

4.1.2 Arquitectura para los enlaces entre referencias

La mayoria de los sistemas dedicados a manipular las relaciones debidas a referencias en los articulos de revistas se adaptan a una arquitectura de referencia [31]. Este modelo se ocupa principalmente de la resolucion de citas en los identificadores de documentos equivalentes, y de la resolución de los identificadores de los objetos digitales en las colecciones. La arquitectura de componentes (en la figura 4.2) tiene sus elementos principales en tres tipos de bases de datos, utilizadas para la resolución de referencias. El proveedor de la informacion (*Publisher*) proporciona metadatos sobre cada trabajo, que se almacenan en la *Location database*; el proveedor tambien se ocupa de actualizar las colecciones. La *Location database* alberga los datos necesarios para obtener la equivalencia entre los identificadores de trabajos y los identificadores fisicos (URLs), y se utiliza durante la resolucion de identificadores. La base de datos de Referencias (*Reference database*) contiene metadatos que corresponden, como minimo a una referencia convencional, y son utilizados durante la resolucion de identificadores. La base de contenidos(*Content database*) contiene los documentos, que se recuperan definitivamente despues de la resolucion de los identificadores. El cliente (*Client*) es un componente para designar genericamente cualquier aplicacion capaz de proporcionar documentos a los usuarios o a otros clientes software.

Los servicios proporcionados son aquellos que permiten recuperar documentos. Con respecto a los servicios de la figura 4.1, se incluyen los servicios de colecciones (*Repository*) (para acceder a los registros) y los servicios de nombres (*Naming*) (para resolver los identificadores). Los servicios de nombres son imprescindibles en el entorno de las referencias entre articulos, ya que es el modo de permitir abstraerse de cuales son los objetos digitales utilizados, a la vez que obtener la equivalencia entre las referencias y las direcciones de objetos digitales.

El flujo de datos entre los componentes también se muestra en la figura 4.2. La *referencia* es enviada por el cliente a la base de referencias (*Reference database*), la cual responde con una lista de *identificadores de trabajos* que se corresponden con la

referencia. El cliente envía los identificadores que pueden ser interesanates a la base de ubicaciones (*Location database*), la cual responde con una lista de URLs (direcciones físicas) que se corresponden con el identificador del documento. Finalmente, bien se puede acceder directamente al contenido del documento, bien a través de un identificador físico (la URL).

Los servicios de nombres siguen en la mayoría de los casos las convenciones expresadas en el estándar *Digital Object Identifier (DOI)* [94], en lo que a sus metadatos se refiere. Esta especificación define un método estándar para caracterizar un trabajo intelectual. El identificador propuesto es una implementación de un *Uniform Resource Name*, y establece una serie de campos que informan a la aplicación sobre las copias del documento, su ubicación en las colecciones de la biblioteca, el proveedor y alguna información adicional que también puede ser útil.

4.1.3 Arquitectura para la manipulación de documentos

Arnold-Moore y cols. [15] proponen una arquitectura cuyo objetivo principal es la manipulación de documentos. Los requisitos funcionales considerados a la hora de diseñar la arquitectura son:

- *Definición de datos*: las restricciones que conciernen la semántica y la estructura de los documentos deben ser accesibles desde los documentos o, en su caso, a partir de una definición de clase.

- *Recuperación de datos*: deben ser factibles los accesos y consultas a los documentos, que pueden suponer acceder a documentos completos, a elementos individuales, a los atributos de éstos y sus valores, al contenido y a los metadatos de los documentos.

- *Manipulación de datos*: ha de ser posible recuperar y reutilizar elementos arbitrarios, así como la generación de nuevos datos.

- *Gestión de documentos*: el último requisito es la gestión de versiones y composición de documentos. En el caso de las versiones esta gestión consiste en la posibilidad de acceder a todas las versiones de un documento, distinguiendo convenientemente entre todas ellas. Este requisito afecta a los identificadores, que pueden ser diferentes en versiones distintas, pero suficientemente similares para deducir a partir de ellos que dos objetos cualesquiera son versiones del mismo documento. La composición de documentos consiste en la obtención de nuevos documentos, reutilizando fragmentos de otros documentos. Los documentos compuestos pueden "incluir" datos compartidos por múltiples documentos, de modo que los cambios en los datos compartidos se vean reflejados automáticamente en el documento matriz.

Los requisitos funcionales dan como resultado la arquitectura de componentes de la figura 4.3. Un gestor de flujo de trabajo (*workflow manager*) administra y dirige el flujo de tareas. Una herramienta de consulta (*query engine*) resuelve las consultas acerca del contenido, la estructura y los metadatos. Un gestor de documentos (*document manager*) se encarga de la seguridad, control de acceso y la creación y manipulación de

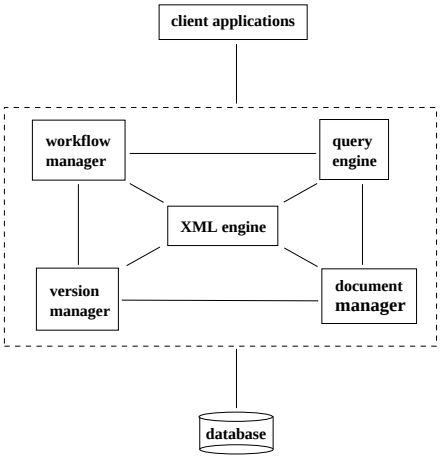


Figura 4.3: Arquitectura para un sistema dedicado a la manipulación de documentos. Existe un componente dedicado exclusivamente a la manipulación de versiones.

los documentos compuestos. Una herramienta XML se ocupa del análisis (*parsing*) y validación de los documentos, la manipulación de los árboles asociados y la comparación de documentos en base a su estructura; este componente es utilizado por los restantes componentes del sistema. El conjunto de componentes forma un bloque con el cual interaccionan las aplicaciones cliente.

La correspondencia que se puede establecer entre esta arquitectura y el modelo de referencia para los servicios básicos de la subsección 4.1.1 es la siguiente: el servicio de búsqueda (*Search service*) está implementado por la herramienta de consulta (*Query engine*), mientras que los servicios de nombres (*Naming service*) y de colecciones (*Repository service*) están implementados simultáneamente por el gestor de versiones (*Version manager*) y el gestor de documentos (*Document manager*).

4.1.4 Protocolos para consultas y recuperación de datos

Los protocolos orientados a las consultas son protocolos de la capa de aplicación, que regulan la interacción entre un cliente y un servidor. Su finalidad es facilitar las tareas de recuperación de información. Por ello, han sido utilizados en las bibliotecas digitales con el fin de conseguir así la deseada interoperabilidad en los accesos a fuentes de información heterogéneas. Su contribución más relevante es la interoperabilidad semántica, que se consigue utilizando los atributos que proporcionan, para describir los datos, y en las consultas. Estos protocolos regulan únicamente la interacción entre los agentes participantes arquitectura de las bibliotecas digitales. El más representativo es el protocolo ANSI/NISO *Z39.50* [124]. Su versión más reciente data de 1995. Fué diseñado para facilitar el acceso a catálogos bibliográficos, y posteriormente ampliado para otros tipos de bases de datos, como las colecciones de documentos. Se basa en un modelo *cliente/servidor*, donde el cliente (*origin*) solicita los servicios de un servidor (*target*). Z39.50 no especifica nada sobre los metadatos, los datos mismos, o la organización de cualquiera de ellos. En consecuencia, una de las funciones tanto de la aplicación cliente como de la aplicación servidor es la traducción de la representación local utilizada para los datos a los criterios Z39.50.

El cliente utiliza el protocolo para consultar y recuperar registros del servidor. La interacción está modelada como un conjunto de mensajes intercambiados entre ambos, donde se invoca alguna de las facilidades consideradas en el protocolo. De entre estas facilidades, algunas de las más representativas son:

- *Initialization Facility*, que sirve para fijar los parámetros que regulan la comunicación.
- *Termination Facility*, utilizada para finalizar la comunicación.
- *Search Facility*, para consultar en las bases de datos.
- *Retrieval Facility*, que permite al cliente recuperar registros del servidor.
- *Explain Facility*, disponible para consultar la base de metadatos del servidor.

4.1.5 Protocolos multi-servicio

Protocolos más recientes que los de Recuperación de Información tienen en cuenta las funcionalidades avanzadas que se vienen incorporando en los últimos años a las bibliotecas.

Dienst

Dienst [43] es el protocolo y arquitectura utilizados en la biblioteca *Networked Computer Science Technical Reports Library* [82], y en el sistema *Cornell Reference Architecture for Distributed Digital Libraries (CRADDL)* [77]. Dienst define un conjunto de servicios y la interacción entre ellos. El conjunto de servicios se puede ver en la figura 4.4:

- *Servicios de interfaz de usuario (User interface)*, que se encargan de proporcionar una interfaz amigable al usuario, para que pueda así aprovechar las funcionalidades del sistema. Los servicios de esta categoría interaccionan con los demás servicios y se responsabilizan de la correcta evolución de las operaciones.
- *Servicios de repositorio (Repository services)*, que almacenan los documentos y permiten acceder a ellos, según se especifica en el modelo de documentos de Dienst.
- *Servicios de índice (Index services)*, encargados de las búsquedas, que aceptan consultas y devuelven la lista de identificadores de los documentos que responden a estas consultas. Utilizan los servicios de repositorio (*repository services*), para extraer la información utilizada en la indexación.
- *Servicios de acceso a colección (Collection services)*, utilizados para definir y acceder a las colecciones de la biblioteca.
- *Servicios "meta" (Meta services)*, que permiten acceder a la información que describe las colecciones y servicios de la biblioteca.
- Servicios de nombres (*Naming services*), que permiten resolver los URN[1].

[1]Los URN (Uniform Resource Name) son un tipo de identificador *persistente*, independiente de la

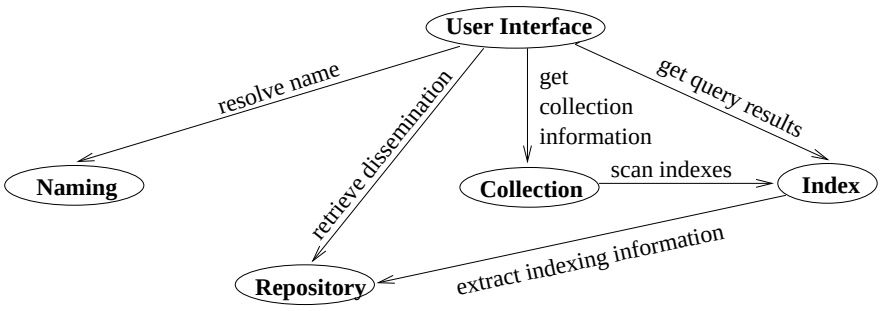


Figura 4.4: El modelo de servicios de NCSTRL.

Stanford InfoBus

En el proyecto de Bibliotecas Digitales de Stanford se incluyen un conjunto de protocolos que permiten la cooperación de servicios heterogéneos. Este proyecto no se centra en un tipo particular de biblioteca, sino que trabaja sobre un modelo de servicios general. Se ha definido un conjunto de servicios que pueden aparecer en una biblioteca, y los protocolos se han diseñado teniendo en cuenta estos servicios. El conjunto de servicios definido consiste en:

- Servicios que permiten descubrir recursos, como *GLOSS* [64].
- Servicios de interfaz de usuario. Por ejemplo, *DLITE* [105] y *SenseMaker* [17].
- Servicios para procesar información.
- Servicios de búsqueda.
- Servicios de traducción.

El conjunto de protocolos que regulan la interacción entre estos servicios recibe el nombre genérico de *Stanford InfoBus*, y se distribuyen en cinco capas [98]:

- Protocolos para la manipulación de items y documentos (*SDLIP*). El protocolo SDLIP permite acceder a las fuentes de información, buscar y recuperar registros.
- Protocolos de búsqueda (*STARTS*). En este caso, el objetivo es facilitar la elección de las mejores fuentes para realizar una consulta, así como la combinación de los resultados de consultas provenientes de estas fuentes.
- Protocolos para manipular metadatos (*SMA*).
- Otros protocolos: gestión del pago por la utilización de recursos (UPAI) y de los derechos de acceso (FIRM).

ubicación física, que permiten localizar los objetos dentro de un espacio de nombres. Difieren de la popular clase URL (*Uniform Resource Locator*), que son identificadores *no persistentes*.

4.2 Una propuesta para servicios de enlaces

La arquitectura y protocolo que se propone en esta tesis tiene como objetivo integrar servicios donde se exploten enlaces en las bibliotecas digitales y mostrar cómo se integran éstos con el resto de funcionalidades del sistema. La detección de enlaces, las consultas sobre las relaciones y la generación de versiones de documentos son las tres novedades más importantes introducidas en esta tesis en lo que a servicios de bibliotecas digitales se refiere. En todas ellas es necesaria una atención especial a los aspectos relacionados con los enlaces. Estos servicios, que no son clásicos en las bibliotecas digitales, pueden, no obstante, integrarse con otros servicios de éstas. Se utiliza aquí el modelo de servicios, dado que su flexibibilidad está sobradamente probada (ver en el apartado 4.1.5 cómo en este modelo se facilita la integración de nuevas funcionalidades con otras existentes: el modelo básico es reconocible en todas las propuestas, mientras que los servicios básicos varían). El modelo de servicios básicos (en la subsección 4.1.1) se ha utilizado como la base sobre la que se construye la propuesta de esta tesis: los servicios que tratan con enlaces se integran con los servicios del modelo básico. Por otro lado, los servicios para enlazar referencias (subsección 4.1.2) han sido incluidos bajo la denominación de "Detección de enlaces". Esta denominación es más precisa y además es necesario distinguir entre esta funcionalidad y otras que también estén relacionadas con la gestión de enlaces. En lo que a versiones se refiere, el aspecto que se considera en nuestra propuesta es su generación, en contraste con el aspecto consistente en su gestión. La comparación de la propuesta de esta tesis con la comentada en el apartado 4.1.3 será objeto de discusión en la sección 4.3 de este capítulo.

La descripción de los servicios de esta propuesta se realiza en una serie de escenarios presentados en la sección 4.2.2, y su integración con otros servicios también se incluye en su presentación. El resultado es la división de tareas entre un conjunto de servicios accesibles mediante sus interfaces, que interaccionan para conseguir las funcionalidades donde se explotan las relaciones.

Por otro lado, la arquitectura y protocolo correspondiente se han descrito en función de un conjunto de vistas que muestran la funcionalidad del sistema, el flujo de datos y la arquitectura de estos últimos. Los requisitos funcionales que guían el diseño son a su vez la base para la definición de un conjunto de escenarios. Las cualidades del sistema, que también influyen en el diseño, serán contempladas al final de la propuesta (apartado 4.2.9).

4.2.1 Presentación

La explotación de relaciones entre documentos se materializa en los siguientes objetivos:

- Realizar consultas sobre las relaciones.
- Generar automáticamente versiones de documentos.
- Detectar las relaciones entre los documentos.

Junto a los servicios que responden a estos objetivos se encuentran otros servicios para acceder a los documentos, consultarlos, insertar nuevos documentos y administrar

la biblioteca, permitiendo la incorporación de nuevas colecciones, datos o servicios.

La visión de los usuarios

En una biblioteca se pueden encontrar usuarios "normales" (denominados *users*), que acceden al sistema para consultar información, administradores (*administrators*), que se encargan del correcto funcionamiento del sistema, y autores (*authors*), que pueden insertar nuevos documentos en las colecciones.

Requisitos de los usuarios comunes

Este tipo de usuario obtiene información de la biblioteca, pero nunca modifica sus bases de datos ni su funcionamiento. Lo que un usuario así puede hacer es:

- Recuperar documentos del sistema, especificando el identificador del documento deseado.
- Obtener una lista de documentos que referencian uno dado (consultar las relaciones).
- Obtener "versiones" de documentos, especificando los parámetros que permiten seleccionar la versión deseada.
- Buscar por términos clave en las colecciones de documentos. Lo que obtendrá en este caso es la lista de identificadores de documentos que responden a la consulta.

Otras funcionalidades que podrían completar esta lista se comentan en la sección 4.3, entre las posibilidades de trabajo futuro.

Requisitos del administrador

El administrador es el único usuario que puede alterar la funcionalidad del sistema o los parámetros que afectan a la globalidad de la biblioteca. Es decir, puede:

- Insertar nuevos servicios y componentes en el sistema.
- Aumentar el rango de clases de documentos soportadas en la biblioteca.

Requisitos de los autores

Los autores son usuarios que no pueden modificar la operación del sistema, pero pueden modificar sus bases de datos. La funcionalidad que estos usuarios pueden aprovechar en el sistema es:

- Insertar nuevos documentos en las colecciones.
- Detectar las relaciones entre documentos, y añadir esta información a la biblioteca.
- Insertar nuevas colecciones.

Datos

Existen dos grandes categorías de datos en las bibliotecas que se consideran en esta tesis: documentos y enlaces. Los enlaces servirán aquí para representar las relaciones entre los documentos.

Los datos de la biblioteca son de suma importancia y se pueden clasificar en función de la estabilidad de los datos, según lo cual se puede hablar de

- Datos *permanentes*, que son los documentos almacenados en las bases de datos del sistema, los metadatos y los enlaces asociados a los datos.
- Datos *temporales*, que son aquellos que el sistema genera y no se almacenan de modo permanente. Aparte de las respuestas a consultas, que fluyen por cualquier sistema, se incluyen aquí las versiones de los documentos generadas en respuesta a una petición del usuario y, que en principio, no se almacenan en la biblioteca[2].

Los tipos de documentos que se generan dinámicamente en el sistema son:

- Versiones de documentos estables.
- Referencias a documentos.
- Respuestas a las consultas.

La lista de datos permanentes en el sistema incluye, además de los documentos, las siguentes categorías:

Enlaces

Los enlaces que representan las relaciones entre los documentos son datos estables, pero generados en el sistema. Esto los diferencia de los documentos, que provienen de fuentes externas.

Metadatos

Los metadatos describen el sistema, y los datos que en él se encuentran. Son básicamente de tres tipos:

- Metadatos de documentos.
- Metatados que describen las colecciones: tipos de documentos en cada colección, etc.
- Metadatos del sistema. Describen cómo se obtiene el sistema mediante una composición de servicios.

Funcionalidades

Las funcionalidades contempladas y que dan lugar a los escenarios que se describen en este capítulo son:

1. Recuperación de documentos.

[2]Esto no se contradice con la posibilidad de que los usuarios creen sus propias bases de datos locales para almacenar los documentos que deseen. Las bases de datos del sistema no se verían por ello afectados por los cambios producidos en las bases de los usuarios.

2. Búsqueda de términos clave en los documentos.
3. Consultas sobre las relaciones entre documentos.
4. Detección de relaciones y generación de los correspondientes enlaces.
5. Generación de versiones de documentos.

4.2.2 Presentación de los escenarios y servicios

Los escenarios presentados en este apartado conducen de modo natural a la definición de una serie de servicios necesarios para conseguir las funcionalidades requeridas. En todos ellos se ha planteado una situación en la cual un usuario interacciona con el sistema, si bien no siempre es necesaria su presencia para desencadenar la acción descrita.

Búsqueda de términos clave

Este escenario presenta una situación en la cual un usuario introduce en el sistema una consulta con la intención de que éste le proporcione la lista de identificadores de los documentos que responden a dicha consulta. La evolución preliminar de actividades se puede ver en la figura 4.5. La consulta del usuario es redirigida a todos los repositorios de la biblioteca (en caso de que el lenguaje de consulta que utiliza el usuario no coincida con el de cada repositorio, será necesario un paso previo de traducción de la consulta[3]). El resultado de cada repositorio es una lista de identificadores de documentos, que son devueltos al usuario.

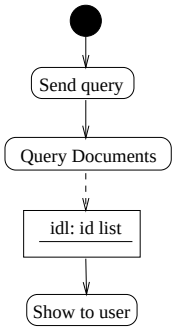


Figura 4.5: Búsqueda por términos clave. Esbozo inicial del escenario.

Consultas sobre relaciones

En este caso el usuario realiza de nuevo una consulta, pero le interesa obtener aquellos documentos que se encuentran relacionados con uno dado. No introduce términos de

[3]La heterogeneidad en los lenguajes de consulta es una posibilidad considerada en el modelo de servicios, pero que no recibe atención especial en esta tesis. Se incluye en el modelo para mostrar cómo esta característica no afectaría al resto del diseño.

búsqueda, sino que indica un documento de partida. Las consultas, pues, se realizarán sobre la base de enlaces en vez de las de documentos. La figura 4.6 muestra la evolución en este escenario. El resultado de la consulta en este caso no es una colección de identificadores de documentos, sino una serie de enlaces. Los identificadores de los documentos (o cualquier otro tipo de información que pueda solicitar el usuario) se extraen de los enlaces en el resultado, antes de ser enviados al usuario.

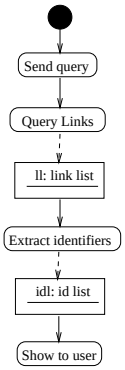


Figura 4.6: Consultas sobre relaciones. Esbozo inicial del escenario.

Inserción de nuevos documentos

La inserción de nuevos documentos siempre viene desencadenada por la intervención de un autor, que indica en qué colección debe realizarse la inserción. El flujo de tareas se puede ver en la figura 4.7. Si el documento supera el proceso de validación incial, se "traduce" (normaliza) a un formato común a la totalidad del sistema, que garantiza la operabilidad en el seno de la biblioteca. La copia resultante de esta normalización será la que se almacene en el repositorio correspondiente.

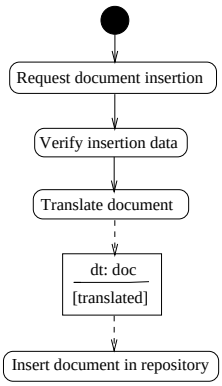


Figura 4.7: Inserción de nuevos documentos. Esbozo inicial del escenario.

Detección de referencias

En este caso se trata de analizar un documento de partida `D`, para encontrar referencias en este documento a otros, que pueden ser asimismo indicadores de relaciones de modificación. Si bien el documento se encuentra en las bases de la biblioteca, sus enlaces asociados aún no han sido generados. El análisis del documento genera enlaces cuyo origen se encuentra en el documento, que se insertarán en la base de enlaces de la biblioteca. Su evolución se muestra en la figura 4.8.

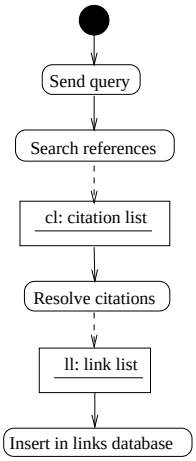


Figura 4.8: Detección de referencias. Esbozo inicial del escenario.

Generación de versiones

Este escenario muestra el proceso seguido para generar una versión histórica de un documento (figura 4.9), cuya copia (de la versión) no se encuentra almacenada en las bases de documentos de la biblioteca. Por ejemplo, el usuario puede querer el documento tal como se encontraba antes de sufrir ninguna modificación, o en su versión actual. Estas preferencias, que seleccionan la versión por criterio de fecha, las indica el usuario en los parámetros de entrada. La versión original del documento debe ser actualizada en base a la información proporcionada por los enlaces que indican cuáles son los documentos que contienen las actualizaciones de contenido que afectan a dicha versión. La solicitud del usuario es procesada por el sistema, que recupera la copia disponible en las bases de datos de la biblioteca, correspondiente a la versión original (a la cual no se le ha aplicado ninguna modificación). A partir de las preferencias del usuario el sistema genera una consulta utilizada para buscar los enlaces que modifican el documento, y además cumplen dichas preferencias. Una vez disponible la colección de enlaces resultantes, se resuelven y se aplican las modificaciones en ellos expresadas a la copia original del documento. El resultado de este proceso de modificación es una nueva versión, que es la solicitada por el usuario.

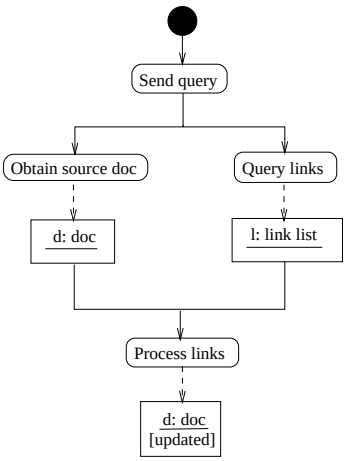


Figura 4.9: Generación de versiones de documentos. Esbozo incial del escenario.

4.2.3 Servicios

En base a los escenarios presentados en el apartado 4.2.2 se concluye la necesidad de servicios capaces de proporcionar:

- *una interfaz de usuario*
- *acceso a las colecciones de documentos*
- *facilidades de consulta*
- *manipulación de documentos*
- *capacidades de actualización que incluyan la inclusión de nuevos documentos y enlaces*
- *traducción (de consultas y documentos)*
- *un modo de designar entidades abstractas (documentos), independiente de su ubicación y del modo en que la correspondiente copia digital se encuentre almacenada*
- *facilidades de administración del sistema*
- *posibilidad de incorporar nuevas categorías de documentos, datos y servicios en la biblioteca.*

Presentación de los servicios

A partir de estas necesidades, se definen los siguientes servicios:

- *Manipulación de documentos.*

Este servicio facilita la manipulación de documentos en el sistema. Permite crear nuevos documentos por composición de fragmentos, así como modificar los documentos disponibles. Cualquier funcionalidad donde se requiera manipular documentos implica la utilización de este servicio.

- *Búsqueda.*

  Los servicios de búsqueda permiten hacer consultas en el sistema. Son capaces de procesar una consulta y devolver los resultados adecuados. Se pueden distinguir dos subtipos: servicios de búsqueda en documentos y servicios de búsqueda en enlaces.

  - *Búsqueda en documentos.* Este servicio permite obtener la lista de identificadores de documentos que responden a una consulta de términos clave en el interior de los documentos.
  - *Búsqueda en enlaces.* El resultado en la utilización de este servicio es una colección de enlaces. También es posible obtener fragmentos seleccionados de estos enlaces (por ejemplo, el identificador del destino del enlace).

- *Generación de enlaces.*

  Este servicio es capaz de analizar un documento de entrada y devolver una colección de enlaces con origen en dicho documento.

- *Interfaz de usuario.*

  Permite al usuario consultar en el sistema, solicitar a éste la ejecución de tareas y visualizar (y, posiblemente, analizar) los resultados.

- *Traducción.*

  Estos servicios son capaces de traducir entre las ontologías externas al sistema y las del sistema. Transforman consultas y documentos a un modelo canónico. Estos servicios son necesarios para garantizar la interoperabilidad en los datos. La normalización de documentos y consultas se realiza siempre utilizando estos servicios. Se pueden distinguir, por tanto, dos tipos de servicio de traducción:

  - *Traducción de consultas.* Este servicio traduce una consulta del lenguaje utilizado por el usuario al lenguaje de consulta empleado en el sistema. Se utiliza únicamente si existe heterogeneidad en los lenguajes de consulta.
  - *Traducción de documentos.* Este servicio traduce los documentos (digitales) de entrada a copias con el mismo contenido, pero cuyos esquemas y formatos garantizan la interoperabilidad. Es crucial para la posterior manipulación y composición de documentos la garantía de esta interoperabilidad, ya que, de otro modo, sería imposible conseguir sendas funciones.

- *Administración.*

  Estos servicios permiten informarse acerca de las colecciones del sistema y los modos de trabajar en él, así como sobre las funcionalidades disponibles.

- *Ampliación.*

  Estos servicios permiten incorporar al sistema nuevas clases de documentos y nuevos servicios.

- *Repositorio.*

  Estos servicios permiten acceder a las bases del sistema. En su modo más básico, posibilitan la recuperación de registros[4], en base a sus identificadores. Tal como se mencionó en la presentación inicial de esta propuesta, hay dos categorías principales de datos en el sistema y, en consecuencia, sendos servicios para los tipos de repositorios respectivos:

  - *Repositorio de documentos.* Este servicio permite acceder a las bases de documentos.
  - *Repositorio de enlaces.* Este servicio permite acceder a las bases de enlaces.

- *Actualización.*

  Este servicio permite a los autores insertar nuevos documentos en la biblioteca, así como requerir la generación de nuevos datos (enlaces).

- *Servicio de nombres.*

  Este servicio permite a los uusarios utilizar identificadores lógicos correspondientes a los documentos abstractos, en vez de identificadores físicos de los objetos (documentos digitales) almacenados en las bases de datos de la biblioteca. El servicio de nombres es capaz de calcular las equivalencias entre uno y otro tipo.

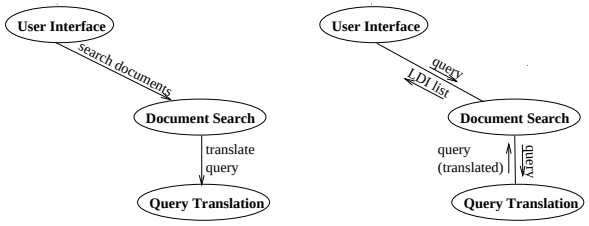
4.2.4 Interacción entre servicios

Se puede explicar los escenarios presentados previamente en base a la interacción de estos servicios, que se invocan a través de sus métodos (*Uses view*), lo cual provoca un flujo de datos entre servicios (*Data flow*). Los gráficos correspondientes a cada servicio se encuentran en las figuras 4.10 a 4.14, cuya conjunción da lugar a los gráficos globales de las figuras 4.15 y 4.16.

Búsqueda por términos clave

El escenario es revisado: un usuario desea localizar aquellos documentos que contienen un cierto término `X`. A través del servicio de interfaz de usuario (*user interface*) puede realizar sus consultas en un lenguaje que le resulta agradable. El servicio de interfaz redirige la consulta a los servicios de búsqueda (*search services*). De la traducción de la consulta al lenguaje utilizado en los repositorios del sistema se encarga el servicio de traducción de consultas (*query translation*) (este paso es necesario únicamente en caso de heterogeneidad en los lenguajes de consulta). Una vez procesadas las consultas, las respuestas son enviadas a la interfaz de usuario, que las integra y las presenta al usuario como un único conjunto coherente. La interacción entre servicios correspondiente a este escenario se puede ver en la figura 4.10.

[4]El término *registro* designa aquí cualquier tipo de dato disponible en el sistema.

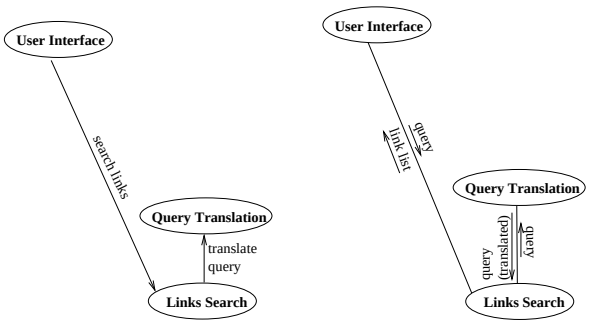


(a) "Uses" view. (b) Data flow.

Figura 4.10: Interacción entre servicios (utilización y flujo de datos) en el escenario de búsqueda por términos clave.

Consultas sobre las relaciones

En este escenario el usuario debe realizar una consulta para obtener información acerca de las relaciones. Esta consulta pasa desde la interfaz de usuario a los servicios de búsqueda en enlaces (*link search services*). Posteriormente, es responsabilidad de los servicios de traducción de consultas la transformación a una consulta equivalente normalizada. El resultado de esta consulta será una colección de enlaces, de los cuales se podrá extraer la información buscada por el usuario.



(a) "Uses" view. (b) Data flow.

Figura 4.11: Interacción entre servicios (utilización y flujo de datos) en el escenario de consultas sobre relaciones.

Inserción de documentos

La inserción de documentos requiere de servicios de actualización (*updating services*) que faciliten esta tarea, validando convenientemente todo el proceso. Cada documento susceptible de ser incorporado a las colecciones del sistema debe ser normalizado, de modo que se ajuste a un modelo canónico que todos los servicios del sistema sean capaces de manipular. De esta tarea se ocupan los servicios de traducción (*translation services*). El resultado de una operación de este tipo, si concluye con éxito, es la inserción del documento en las bases de datos, utilizando para ello los servicios de acceso a repositorios (*repository services*).

Detección de referencias

La detección de referencias (y consiguiente generación de enlaces) es una tarea que asumen los servicios de actualización (*updating services*), dado que suponen modificaciones en las bases de enlaces. Es responsabilidad de estos servicios invocar solicitar la detección de referencias en el documento de entrada (*Link detection services*), validar la corrección de los enlaces generados haciendo uso de los servicios de administración (*Administration services*) y, si todo ha ido correctamente, requerir de los servicios de acceso a repositorios (*repository service*) la inserción de los enlaces en la base correspondiente.

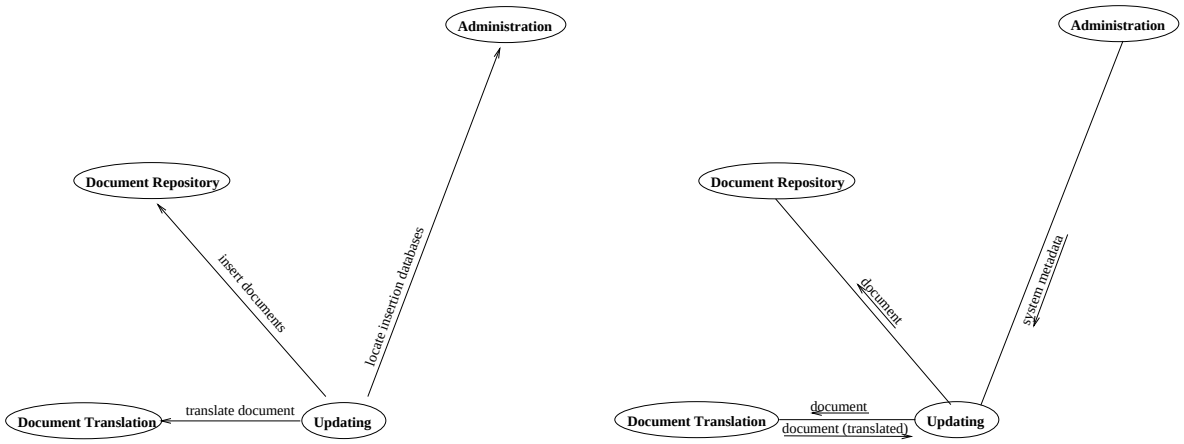
Generación de versiones de documentos

En este escenario, la petición del usuario de una versión dada de un cierto documento `D` es transmitida por el servicio de interfaz de usuario (*user interface service*) al de manipulación de documentos (*document management service*), el cual la convierte en una sucesión de consultas y acciones cuyo resultado es la composición del documento (versión) solicitado. La generación de dicha versión requiere: acceder a la copia de la versión original, disponible en las bases del sistema (*repository services*), consultar la base de enlaces (*search services*) para averiguar cuáles son los documentos que la modifican y componer los fragmentos adecuados para obtener la versión deseada, tarea que asume el servicio de manipulación de documentos (*document management service*). Por último, la versión solicitada es enviada al servicio de interfaz de usuario, que se encarga de presentarla al usuario. La evolución de tareas comentada se puede ver en la figura 4.14.

Una visión global de la interacción de los servicios se puede ver en la figura 4.15. En este caso se trata de la utilización entre servicios (*uses view*). Un servicio *S1* interacciona con otro servicio *S2* si utiliza el servicio *S2* en algún momento de su operación. El flujo de datos entre servicios se ajusta a lo comentado en los escenarios, y la visión global se puede ver en la figura 4.16.

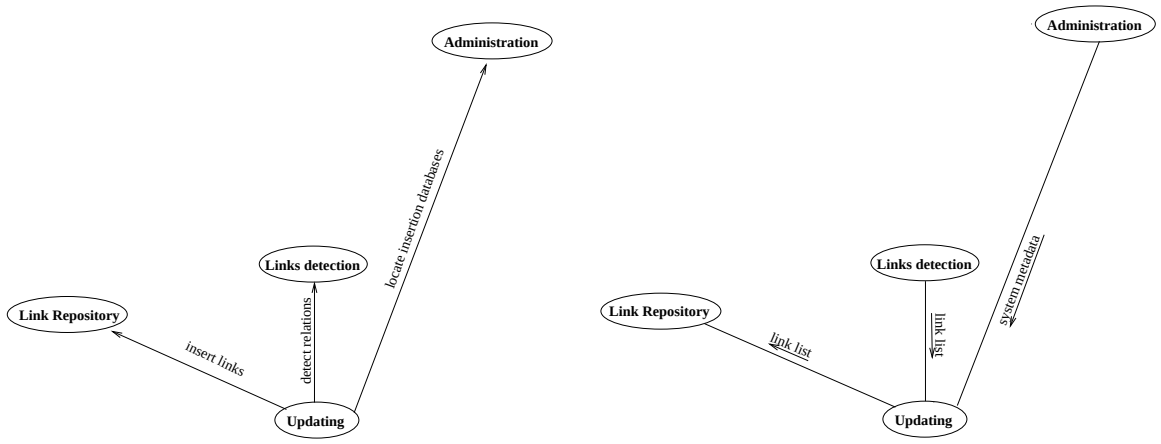
4.2.5 Interfaces de servicios

Los servicios pueden invocar los métodos de otros servicios. La lista que sigue a continuación incluye los métodos relevantes en la resolución de las necesidades provocadas por las funcionalidades consideradas.



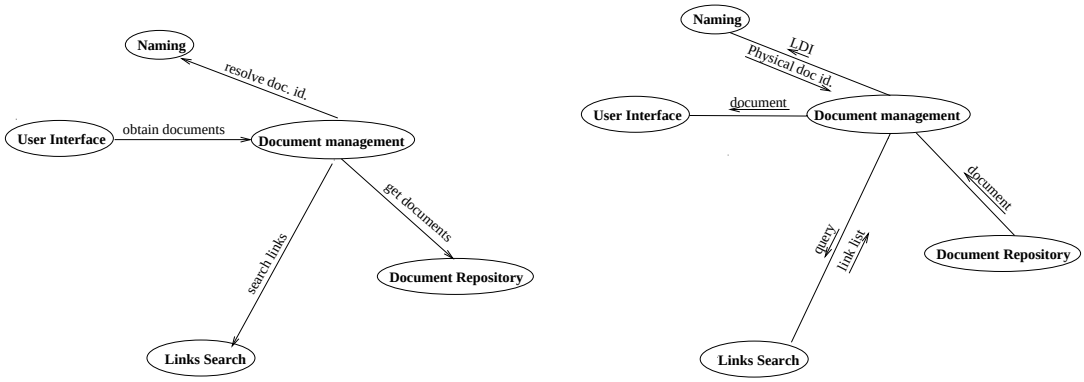
(a) "Uses" view. (b) Data flow.

Figura 4.12: Interacción de servicios (utilización y flujo de datos) en el escenario de inserción de documentos.



(a) "Uses" view. (b) Data flow.

Figura 4.13: Interacción de servicios (utilización y flujo de datos) en el escenario de detección de referencias.



(a) "Uses" view. (b) Data flow.

Figura 4.14: Interacción de servicios (utilización y flujo de datos) en el escenario de generación de versiones de documentos.

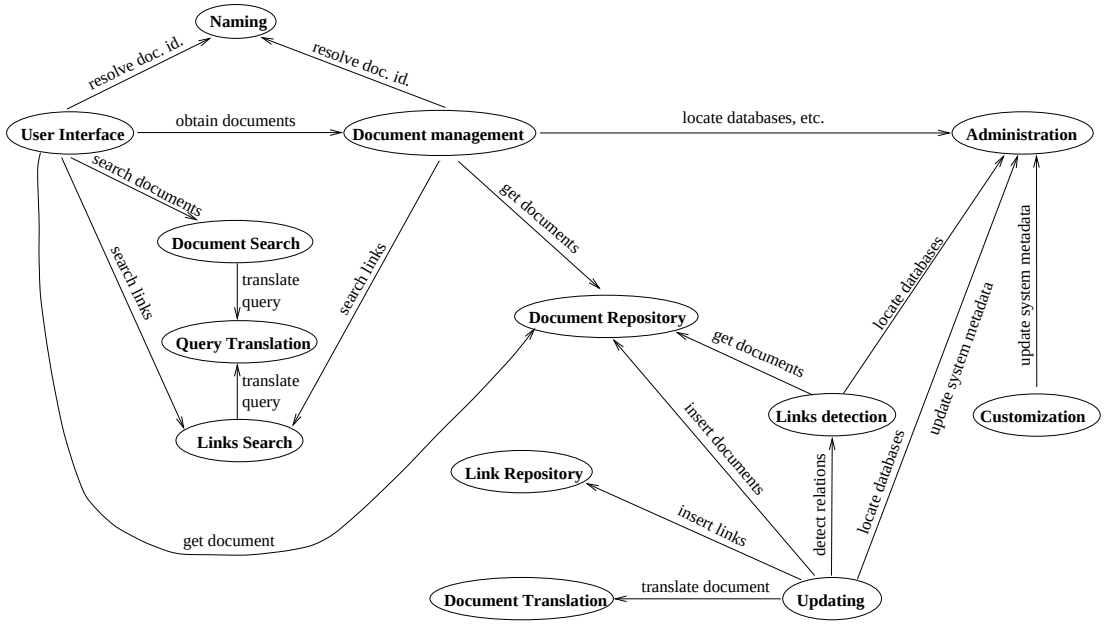


Figura 4.15: Interacción entre servicios. Vista global.

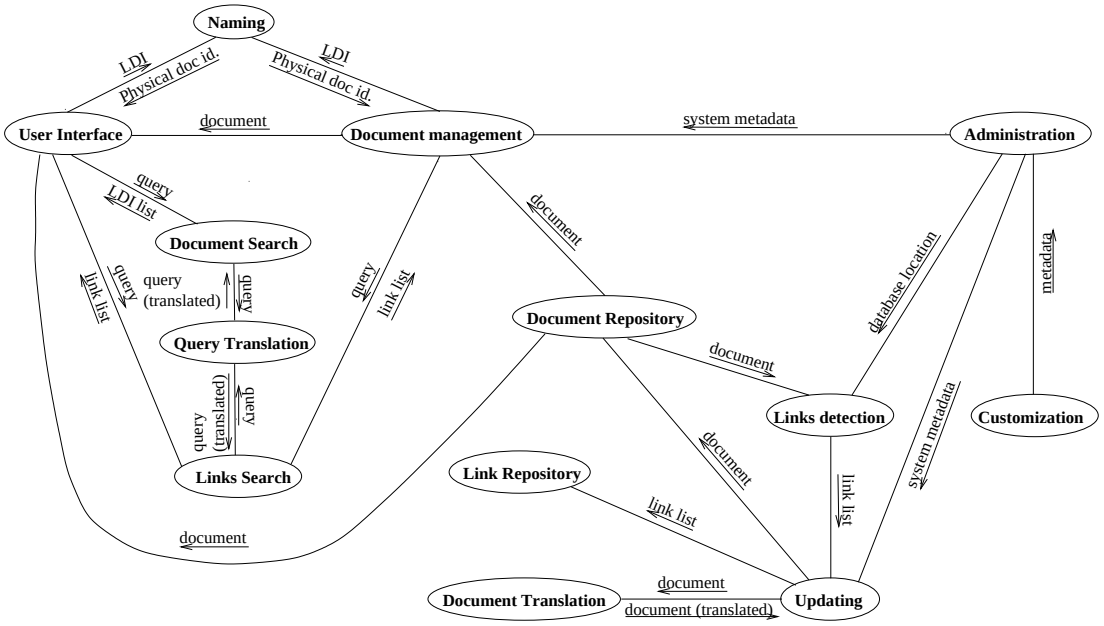


Figura 4.16: Flujo de datos entre servicios. Vista global.

Servicio de interfaz de usuario

- **GetDocument**

  Presenta un documento (contenido) al usuario. Recibe en los parámetros de entrada un identificador de un documento.

- **Search**

  Busca en las colecciones de la biblioteca. Recibe la consulta del usuario como entrada. Permite buscar en los documentos, enlaces o cualquier otro tipo de dato sobre el cual se puedan realizar consultas en la biblioteca.

- **BrowseCollection**

  Permite a los usuarios navegar por las colecciones de la biblioteca.

Servicio de manipulación de documentos

- **GetVersion**

  Devuelve una versión de un documento, en las condiciones (momento temporal, etc.) definidas por el usuario. Recibe como entrada el identificador de un documento y los criterios que permiten seleccionar la versión entre las demás versiones del mismo documento. En caso de ser necesario (la versión no está disponible en las bases de la biblioteca), este método genera la nueva versión por composición de documentos.

- **GetRelatedInformation**

  Toma como entrada el identificador de un documento y los atributos que caracterizan las relaciones sobre las cuales se desea obtener información. Genera un dossier de información relacionada.

Servicios de repositorio

- **Insert**

  Recibe como entrada un documento (o colección de enlaces) y los inserta en el repositorio.

- **GetDoc**

  Devuelve el documento, cuyo identificador recibe en los parámetros de entrada.

Servicio de nombres

- **CreateName**

  Recibe como entrada una instancia (dirección que permite acceder a ella) de un objeto o servicio y devuelve un nombre único. Este nombre y los datos necesarios para su resolución son almacenados por el servicio de nombres.

- **ResolveName**

  Recibe un nombre y devuelve el conjunto de instancias (direcciones de acceso) asociadas a él.

Servicio de búsqueda en documentos

- **SubmitQuery**

  Busca en las base de documentos, utilizando la consulta recibida. El resultado es una lista de identificadores, que corresponden al conjunto de documentos que responden a la consulta.

Servicio de búsqueda en enlaces

- **SubmitQuery**

  Proporciona los enlaces que cumplen los criterios especificados en los argumentos de invocación. Los criterios se detallan como una secuencia de pares (atributo, valor).

Servicio de creación de enlaces

- **GenerateLinks**

  Su entrada es un documento y el identificador correspondiente. Genera una colección de enlaces (enlaces cuyo origen se encuentra en el documento de entrada). El identificador es necesario, ya que será incluido en los enlaces.

Servicios de administración del sistema

- **ListServices**

  Lista los servicios del sistema.

- **ListServiceMethods**

  Lista los métodos asociados a un cierto servicio.

- **ListCollections**

  Lista las colecciones del sistema, junto al modo de acceso a dichas colecciones.

- **ListDocumentTypes**

  Proporciona la lista de clases de documentos soportados en el sistema.

- **DocTypeOntMapping**

  Retorna la descripción acerca de la equivalencia entre el vocabulario utilizado en los textos de los documentos de una cierta clase y el utilizado en el interior del sistema para esa misma clase.

- **SearchToolsMapping**

  Retorna la descripción acerca de la equivalencia entre el lenguaje utilizado por la comunidad de servicios del sistema y los lenguajes particulares de cada repositorio.

Servicios de ampliación

- **InsertService**

  Este método se utiliza para incorporar nuevos servicios al sistema.

- **InsertCollection**

  Este método permite incorporar nuevas colecciones al sistema.

- **InsertDocType**

  Permite incorporar nuevas clases de documentos en el sistema.

Servicio de traducción de consultas

- **Translate**

  Traduce consultas del lenguaje del sistema a los lenguajes particulares de cada repositorio, y viceversa.

Servicio de traducción de documentos

- **Translate**

  Traduce el documento de entrada del formato externo al sistema al modelo común al sistema. Para dicha traducción utiliza la equivalencia de vocabularios asociada a la clase de documentos a la cual pertenece el documento.

4.2.6 Componentes

La fase de desarrollo en la biblioteca da lugar a un conjunto de componentes arquitectónicos que colaboran para implementar los servicios de la biblioteca, los cuales aparecen en las figuras 4.17 y 4.18. Cada componente asume una serie de tareas bien definidas, de modo que se promueva la escalabilidad y modularidad en el sistema.

Algunos componentes están involucrados en tareas relacionadas con la explotación de interrelaciones (implementan tareas ligadas a los objetivos de esta tesis) mientras que otros completan la bibilioteca ya que son necesarios para el correcto funcionamiento e interacción en el sistema. Este es además el criterio con el que se van a agrupar en su presentación. Así pues los componentes propuestos son:

- **Componentes que participan en la explotación de enlaces**
  - **Servidor de documentos**
    Responsable de proporcionar documentos y todo aquello que pueda ser necesario para generarlos (metadatos y enlaces asociados). Las consultas de los usuarios pasan a través de este elemento, que es capaz de analizarlas y decidir cuáles son los pasos necesarios para atender la consulta con éxito. Participa en la implementación de los servicios de manipulación de documentos, búsqueda y de repositorio.
  - **Herramienta de búsqueda en enlaces**
    Realiza búsquedas en la base de enlaces. Implementa el servicio de búsqueda en enlaces y utiliza los servicios de traducción para las consultas. Es un *wrapper* para la herramienta de consulta, que oculta las peculiaridades del componente utilizado finalemente y asegura la disponibilidad de un lenguaje estándar en el sistema.
  - **Generador de enlaces**
    Genera enlaces en base a los resultados que obtiene del análisis del documento de entrada. Analiza el documento de entrada, de modo que detecta referencias a otros documentos y crea una colección de enlace que pueden ser insertados en cualquiera de las bases de enlaces. Implementa el servicio de detección de enlaces.
  - **Traductor de documentos**
    Traduce los documentos que llegan a la biblioteca a un formato común que garantiza la interoperabilidad en los datos de la biblioteca. Implementa el servicio de traducción. Participa en los procesos de inserción de documentos en la biblioteca, y es necesario antes de poder realizar cualquier otra operación sobre los documentos.
- **Otros componentes**

  Existen otros componentes que interaccionan con los que se han descrito hasta ahora. Estos componentes (u otros similares) están presentes en la mayoría de las bibliotecas digitales, y se incluyen en este modelo con la finalidad de mostrar la sencillez de la integración de los componentes que manipulan enlaces en cualquier biblioteca.

- **Herramienta de consultas en documentos**
  Garantiza la indexación y búsqueda en los documentos. Es un *proxy* para la herramienta de búsqueda, que será la que de hecho indexe y realice las búsquedas en las colecciones de documentos. En respuesta a la consulta que recibe devuelve la colección de los identificadores de los documentos que se ajustan a la consulta. Implementa el servicio de búsqueda en documentos y utiliza, de ser necesario, el servicio de traducción.
- **Traductor de consultas**
  Traduce las consultas del lenguaje común al sistema a los lenguajes locales de cada repositorio, y viceversa. Este tipo de componente es necesario en las bibliotecas donde se observa heterogeneidad en los lenguajes. Implementa el servicio de traducción de consultas.
- **Interfaz de usuario**
  Este componente ofrece la interfaz que permite al usuario interaccionar con la biblioteca. Ofrece al usuario formularios de consulta y adapta los resultados procedentes de otros componentes, con el fin de presentarlos de modo "amigable" al usuario. Interacciona con el servidor de documentos, al cual dirige las consultas del usuario, y del cual recibe los resultados. Implementa el servicio de interfaz de usuario.
- **Agente de actualización de las bases de datos**
  Se ocupa de todas las funcionalidades que suponen algún tipo de actualización en el sistema. Pone en marcha los procesos de inserción de documentos y generación de enlaces, que finalizan con la actualización de las correspondientes bases de datos. Implementa el servicio de actualización.
- **Adaptador**
  Permite enriquecer el sistema añadiendo nuevos servicios y clases de documentos. Implementa el servicio de adaptación.

Por último se pueden citar los repositorios de la biblioteca, disponibles mediante los servicios de acceso a repositorio:

- **Metadatos del sistema**
  Contiene los metadatos del sistema. Relacionada también con los servicios de administración. Será consultada por cualquier componente que desee consultar los metadatos del sistema.
- **Documentos**
  Este repositorio se corresponde con las bases de documentos, los metadatos relacionados con estas bases y los métodos que permiten acceder a ambos. También permite actualiza dichas bases.
- **Enlaces**
  Este repositorio se corresponde con las bases de enlaces, los metadatos relacionados con estas bases y los métodos que permiten acceder a ambos.

Una visión conjunta de la interacción entre componentes se encuentra en la figura 4.17 (*calls view*) y la figura 4.18 (*data flow*). También es posible describir los escenarios

en base a estos componentes, que se desarrolla en el apartado 4.2.7 de la memoria de tesis.

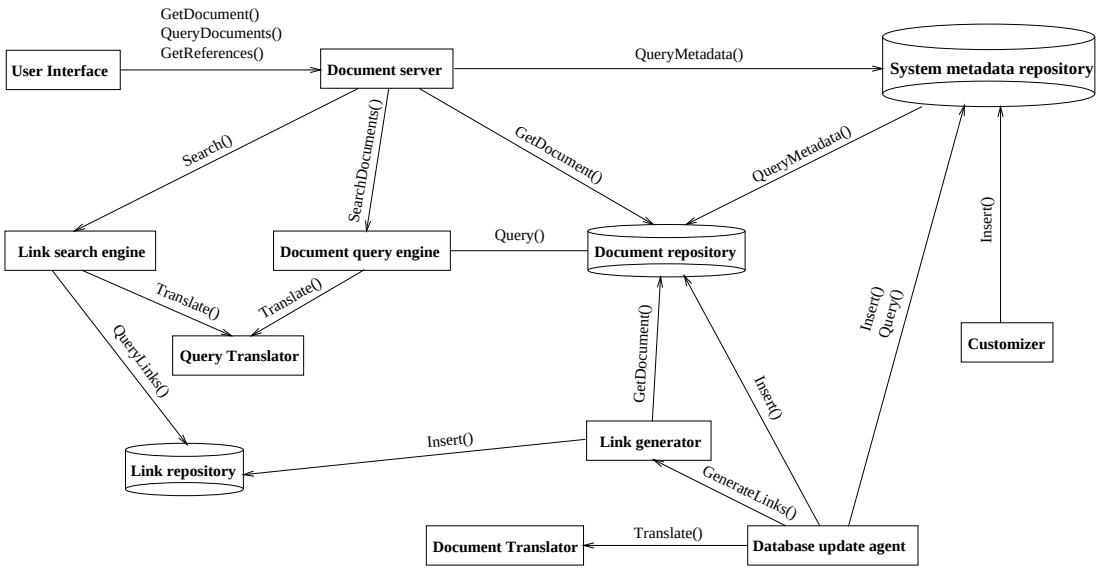


Figura 4.17: Interacción entre componentes. Invocación.

4.2.7 Interacción entre componentes

La interacción entre componentes se puede presentar en términos de los escenarios introducidos en la sección 4.2.2, del mismo modo que se hizo con los servicios. En este caso únicamente los escenarios donde los enlaces juegan un papel especialmente relevante son comentados, con el objetivo de que el capítulo no sea excesivamente extenso.

Consultas sobre relaciones

En este escenario, un usuario busca en la biblioteca información acerca de los documentos relacionados con cierto documento `D`, que especifica en su solicitud. El resultado será la lista de los identificadores de los documentos que tienen una relación con `D`, del tipo que interesa al usuario.

El usuario utiliza la interfaz de usuario (*User Interface*) para realizar su solicitud. Este componente se encarga de invocar alguno de los métodos del servidor de documentos (*Document Server*), al cual pasará el identificador del documento `D` así como los criterios que definen el tipo de relación que está buscando.

El servidor de documentos invoca a su vez a la herramienta de búsqueda en enlaces (*Link Search engine*), a la cual pasa una consulta en la que incluye los criterios de filtro para los enlaces. Como resultado de esta consulta se obtiene una colección de enlaces, que pasarán al servidor de documentos. Este seleccionará la información que deba presentar al usuario: identificadores de los documentos y tipo del enlace. Una vez

seleccionada, esta información será devuelta a la interfaz de usuario (*User Interface*), que la adecuará para presentarla al usuario del modo más conveniente.

En resumen, los componentes involucrados en este escenario son: interfaz de usuario (*User Interface*), servidor de documentos (*Document Server*), herramienta de búsqueda en enlaces (*Link Search Engine*) y repositorio de enlaces (*Link Repository*). Los datos intercambiados entre estos componentes también pueden observarse en la figura 4.19. Las consultas pasan del servidor de documentos (*Document Server*) desde la herramienta de búsqueda en enlaces (*Link Search engine*); el mismo recorrido corresponde a los datos respuesta, en sentido inverso. El servidor de documentos (*Document Server*) devuelve los documentos al componente de interfaz de usuario (*User Interface*).

Detección de relaciones

En este escenario un agente actualizador (*Database Update agent*) inicializa la operación, al realizar una solicitud de detección de relaciones y generación consiguiente de los enlaces cuyo origen se encuentre en un documento que proporciona, `D`. La detección y generación son responsabilidad del generador de enlaces (*Link Generator*). Esta generación supone localizar la copia digital de `D` disponible en el repositorio de documentos (*Document repository*). El análisis de su contenido lo realiza el generador de enlaces (*Link Generator*), del cual producirá una colección de enlaces listos para ser insertados en el repositorio de enlaces. La inclusión de los enlaces tendrá lugar en el repositorio correspondiente (*Link Repository*), cuyos métodos de inserción son invocados por el generador de enlaces.

En resumen, los componentes que participan en la operación de este escenario son: agente actualizador (*Database Update agent*), generador de enlaces (*Link Generator*), repositorio de documentos (*Document Repository*) y el repositorio de enlaces (*Link Repository*). Los datos intercambiados se pueden ver en la figura 4.20. El documento que debe ser analizado es recuperado del repositorio de documentos por el generador de enlaces. Una vez generados los enlaces, éstos pasan al repositorio de enlaces, para ser insertados.

Generación de versiones de documentos

En este escenario un usuario solicita una copia de un documento `D`, tal cual se encontraba en una cierta fecha. Tanto el identificador del documento como la fecha deben ser indicados en su solicitud. Esta se realiza a través de la interfaz de usuario (*User Interface*), la cual invocará alguno de los métodos del servidor de documentos (*Document Server*), incluyendo los parámetros antes mencionados en sus argumentos. El servidor de documentos descompone la consulta en las subconsultas y acciones correspondientes, algunas de las cuales se realizarán invocando la intervención de algún otro componente. La secuencia de pasos que conducen a la obtención de la versión solicitada es:

1. Localización y recuperación de la copia del documento a la cual serán aplicadas las modificaciones que permiten obtener la versión deseada.

2. Aplicación de las modificaciones representadas en los enlaces a dicha copia. Este es un proceso de composición que se expresa como una transformación de un

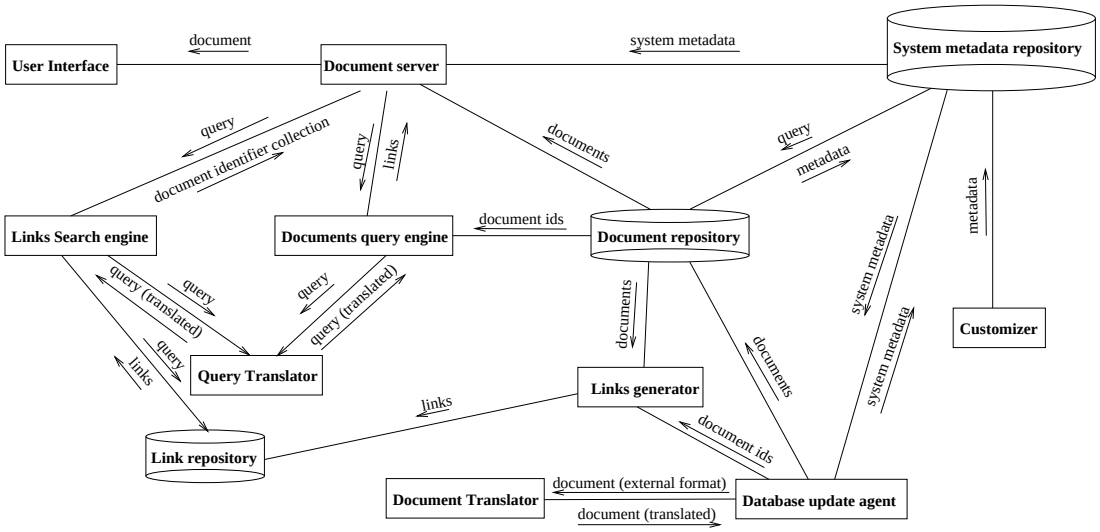
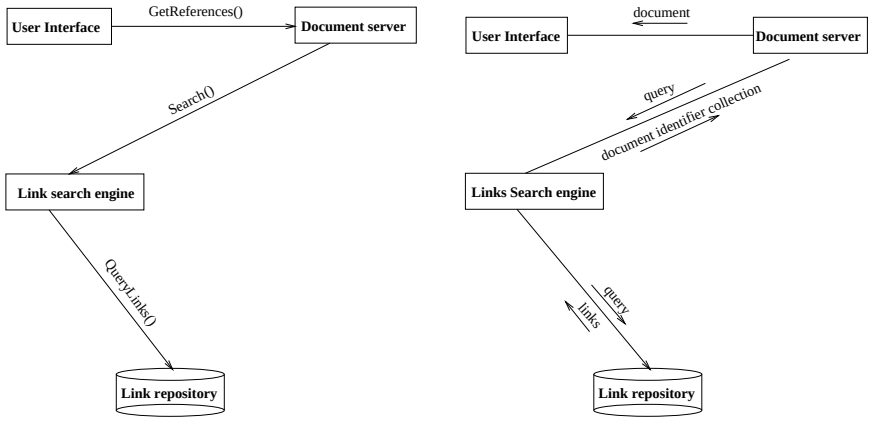
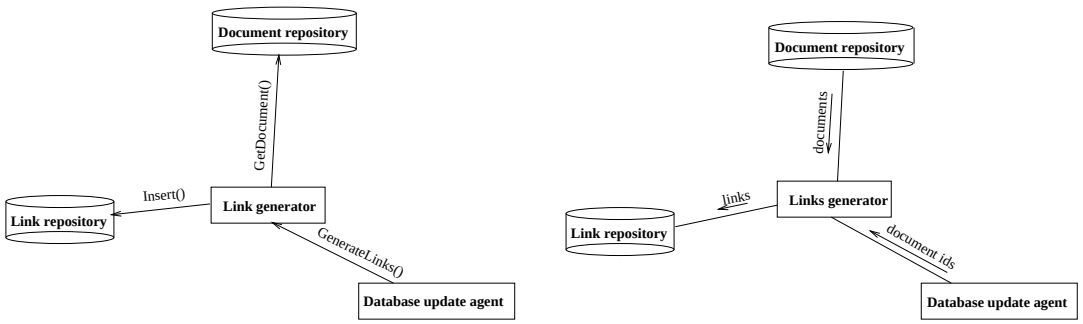


Figura 4.18: Flujo de datos entre componentes.



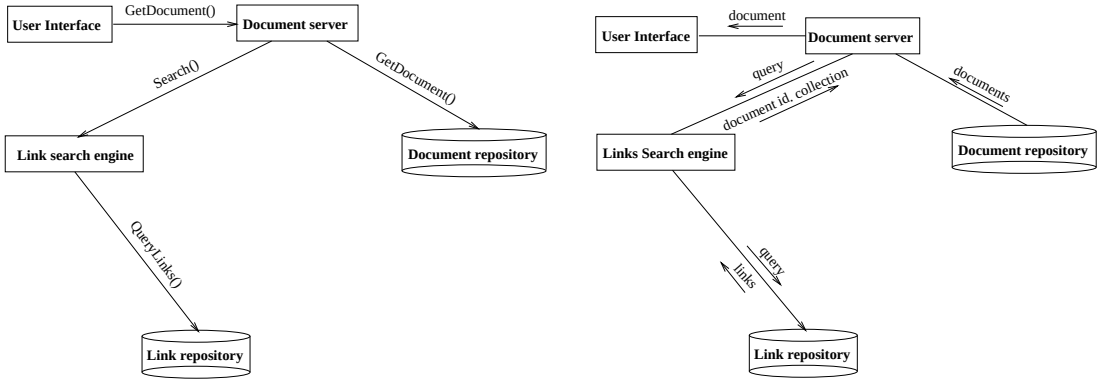
(a) "Calls" view. (b) Data flow.

Figura 4.19: Interacción entre componentes (invocación y flujo de datos) en el escenario de consulta sobre relaciones.



(a) "Calls" view. (b) Data flow.

Figura 4.20: Interacción entre componentes (invocación y flujo de datos) en el escenario de detección de relaciones.



(a) "Calls" view. (b) Data flow.

Figura 4.21: Interacción entre componentes (invocación y flujo de datos) en el escenario de generación de versiones de documentos.

documento origen (la copia mencionada), del cual se encarga el servidor de documentos.

3. Envío a la interfaz de usuario de la copia resultante, que podrá ser presentada al usuario por ésta.

En resumen, los componentes que participan en este escenario son: interfaz de usuario (*User Interface*), servidor de documentos (*Document Server*), herramienta de búsqueda en enlaces (*Link Search Engine*), repositorio de documentos (*Document Repository*) y repositorio de enlaces (*Link Repository*). Los datos intercambiados entre ellos se pueden ver en la figura 4.21. Las consultas viajan desde el servidor de documentos hasta la herramienta de búsqueda en enlaces, y en sentido inverso lo hará la información extraída de los enlaces que responden a la consulta. El servidor de documentos recupera los documentos digitales de los repositorios convenientes y envía a la interfaz de usuario aquellos que resultan de la operación.

4.2.8 Arquitectura de los datos

Tres grandes categorías de datos aparecen en la biblioteca: documentos, metadatos y enlaces. Los documentos son aquellos a los cuales el usuario podrá acceder. Los enlaces y metadatos contienen información sobre ellos, pero nunca serán accesibles al usuario tal cual se encuentran almacenados en las bases de datos de la biblioteca. También existen metadatos que describen el sistema, necesarios para su funcionamiento correcto.

Metadatos del sistema

Los metadatos requeridos en el sistema se deducen en función de las necesidades de los servicios.

Los servicios de manipulación de documentos (*Document Management*) son capaces de generar nuevos documentos a partir de los almacenados en la biblioteca y de los enlaces. Para ello necesitan:

- información sobre la ubicación de los repositorios de documentos y enlaces
- información sobre la estructura de los enlaces (campos), necesaria para consultar dichos datos
- información para calcular la equivalencia entre los identificadores de documentos utilizados por el usuario y los de sus respectivas copias digitales. Esta equivalencia la proporciona el servicio de nombres (*Naming service*).

Los servicios de interfaz de usuario (*User Interface*) proporcionan a los usuarios documentos cuyo aspecto resulte agradable, así como las interfaces necesarias para utilizar los servicios del sistema. "Traduce" las peticiones del usuario en las invocaciones de los métodos adecuados de los restantes servicios del sistema. Para ello necesita:

- saber cuáles son los servicios del sistema y los métodos que puede invocar. Es decir, necesita información acerca de los servicios de búsqueda y manipulación de documentos.

Los servicios de búsqueda (*Search*) permiten realizar consultas sobre los documentos y enlaces. Necesitan para ello:

- en el caso de heterogeneidad en los lenguajes, la dirección de los traductores que se encargarán de la tarea de traducción.

El servicio de detección de enlaces (*Link Detection*) analiza los documentos para crear enlaces ajustados a un modelo canónico que garantiza que otros servicios del sistema también sean capaces de utilizarlos. Necesita, por tanto:

- saber cuáles son los repositorios de enlaces
- información útil para calcular las equivalencias entre las referencias detectadas en el contenido de los documentos y las categorías (clases) de documentos en el sistema

- la estructura (campos) de los enlaces, que le permitan generar enlaces conformes a ella
- información útil para calcular las correspondencias entre los modos de caracterizar los fragmentos de los documentos en el lenguaje utilizado en los textos de los documentos y las descripciones equivalentes en la ontología interna al sistema.

El servicio de traducción de documentos (*Document Translation*) obtiene una copia digital de un documento, conforme a un formato y modelo comprensible por los servicios del sistema, a partir de copias cuyo modelo y formato son ajenos al sistema. Para llevar a cabo esta traducción necesita:

- información útil para calcular las correspondencias entre los modos de caracterizar los fragmentos de los documentos en el lenguaje utilizado en los textos de los documentos y las descripciones equivalentes en la ontología interna al sistema (coincide en este punto con el servicio de detección de enlaces).

Los servicios de traducción de consultas (*Query Translation*) operan sobre los lenguajes de consulta y los modelos de atributos. Una descripción detallada de lo que este tipo de servicios requiere se puede encontrar en [16].

Los servicios de administración y ampliación albergan los metadatos que describen el sistema y facilitan la inserción de nuevas categorías de datos y servicios en la biblioteca. Para ello necesitan:

- saber cuáles son los servicios y componentes del sistema
- poder acceder a los metadatos de cada repositorio

Como conclusión de las necesidades descritas, los metadatos que describen el sistema son:

- Lista de servicios en el sistema, sus métodos y ubicación.

Service name	*Service category*	*Location*	*Service methods*			
<name value>	<cat. value>	<loc. value>	m_1	m_2	...	m_n

- Clases de documentos en el sistema.
- Para cada clase de documentos, la equivalencia entre el vocabulario utilizado en las referencias textuales y la ontología interna a la biblioteca, tanto en lo que se refiere a la caracterización de los documentos como a sus fragmentos internos.

Document class	*Term or expression used in citations*			
<class value>	e_1	e_2	...	e_n

Element name	*List of equivalent expressions*			
<name value>	e_1	e_2	...	e_n

- En cada base de datos (repositorio), los metadatos que lo describen
- La estructura de los enlaces.

Arquitectura de los documentos

Un documento es la agregación de tres elementos de información: su contenido, los metadatos que lo describen y los enlaces que le afectan (ver figura 4.22).

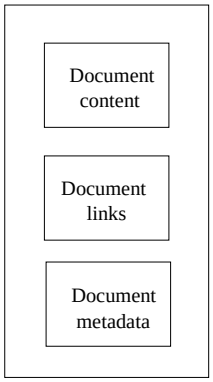


Figura 4.22: Los documentos son la agregación de tres elementos de información: contenido, metadatos y enlaces.

Contenido.
El contenido de un documento se almacena en una copia digital, cuya estructura lógica refleja la estructura semántica del documento abstracto (explicada en el capítulo 2).

Metadatos de los documentos.
Cada documento del sistema es designado por un identificador único que lo distingue de los restantes documentos de la biblioteca. Las ventajas de utilizar este tipo de identificador (ligado a la entidad abstracta en vez de a las copias digitales) son varias, pero a continuación se listan las más relevantes cuando se trabaja con servicios que explotan relaciones:

- Es posible deducir un Identificador Lógico para el documento a partir de las referencias encontradas en su contenido, pero nunca es posible calcular a partir de estas referencias direcciones físicas. En esta propiedad se basa la detección de referencias.
- Las relaciones detectadas a partir de las referencias vinculan documentos abstractos, nunca objetos digitales.
- Un identificador lógico permite abstraerse, si así se desea, de la existencia de versiones, así como de la ubicación física de la(s) copia(s) de un documento. Describe una entidad abstracta, lo cual es sumamente útil al usuario, que podrá realizar peticiones como "Deseo obtener la versión del documento `D`, tal cual se encontraba en la fecha `t`", donde `D` es referenciado por su identificador lógico.

El identificador lógico propuesto en esta tesis sigue los criterios establecidos el estándar DOI [94]. Considera, no obstante, que la parte local del identificador se obtiene en base a la clase a la que pertenece el documento, el identificador que discrimina el documento entre todos los de la clase y -si procede- el criterio que permite discriminar entre las distintas versiones de un mismo documento.

Ejemplo 23. Sea el siguiente identificador:

`urn:thisthesisprototypedomain:collection1/D1-13-1980`

Este identificador designa un documento, dentro del espacio de nombres correspondiente al prototipo de esta tesis, perteneciente a la colección `collection1`, y designado dentro de ella por el identificador `D1-13-1980`. □

La *resolución* de cada identificador de un documento da como resultado una copia que lo representa. El conjunto de metadatos que describe cada documento abstracto, que se utilizan en la resolución, aparecen en la tabla 4.1.

Field	*Field description*
Document Logical Identifier	Document identifier
Document Physical address	Address of the digital copy stored in the library databases.
Type of document	Class the document belongs to.
Document date	Date of the version the stored document copy corresponds to.
Title	A document description, readable by a human being.

Tabla 4.1: Identificador Lógico de un Documento (LDI).

Enlaces.
Los enlaces son objeto de un estudio exhaustivo en el capítulo 3.

4.2.9 Cualidades del sistema

Las cualidades en las cuales se ha hecho hincapié han sido la escalabilidad, la autonomía (federación) de los componentes, la interoperabilidad y la transparencia cara al usuario del modo en que opera el sistema y la representación la información. Algunas de estas cualidades son generales a cualquier sistema y se incluyen en directrices generales de la arquitectura del software [18]. Otras se pueden encontrar en las directrices específicas para las bibliotecas digitales especificadas por Arms [10] y como pautas de diseño en algunas bibliotecas [45, 25].

La arquitectura que integra servicios donde se tratan los enlaces debe ser, por tanto, *extensible* y *escalable*, lo cual significa que la incorporación de nuevos elementos y servicios no ha de ser una operación costosa. Por otro lado, un diseño basado en el modelo de servicios, como éste, conduce necesariamente a una arquitectura *abierta* (la funcionalidad del sistema está disponible bajo la apariencia de servicios o unidades funcionales individuales, cada una de las cuales publica la semántica de su operación a través de sus métodos de acceso, lo cual permite insertar o eliminar nuevos elementos sin afectar la semántica de los demás). Propiedades adicionales son que el sistema resultante es capaz de aumentar su funcionalidad mediante la incorporación de nuevos servicios que interaccionan con los actuales; es decir, se obtiene *federación* gracias a la autonomía de los servicios.

Una propiedad adicional de esta arquitectura se rescinde al campo de las bibliotecas digitales. Una cuestión importante en las bibliotecas digitales es que se debe tener en

cuenta que los usuarios que las utilizan lo hacen buscando trabajos intelectuales, no ficheros [10]. Esto supone que es responsabilidad del sistema ocuparse de calcular la equivalencia entre dichas entidades abstractas (las que busca el usuario) y los documentos digitales que contiene en sus repositorios. Esta propiedad es la que conduce a la necesidad de los metadatos que describen los documentos y de los servicios de nombres.

Algunas propiedades adicionales se listan a continuación:

- *Modificabilidad.* Las modificaciones se aplican a los servicios (o componentes en la fase de implementación), lo cual supone que nunca repercutirán de forma drástica en la biblioteca.
- *Interoperabilidad* en los datos, conseguida gracias a la utilización de un formato estándar para todos los datos de la biblioteca.
- *Integridad.* Los servicios que proporcionan funcionalidades similares, lo hacen de modo similar. Por ejemplo, todos los servicios de búsqueda operan del mismo modo: reciben la consulta, si es necesario la traducen, ejecutan la búsqueda en los índices correspondientes y devuelven los resultados (que previamente habrán filtrado y normalizado).

4.3 Discusión

Los servicios que conforman la aportación de esta tesis están relacionados con la manipulación y explotación de las relaciones (implementadas como enlaces) entre los documentos, o con la semántica de los documentos y relaciones. La arquitectura y el protocolo de interacción se han definido en esta tesis en términos de un conjunto de "servicios", siguiendo así el modelo más extendido en las bibliotecas digitales [11, 45, 104]. La utilización de este modelo facilita la *integración* de servicios, lo cual se corresponde con el objetivo de esta tesis de que debe ser posible integrar los servicios que explotan los enlaces en cualquier biblioteca. La descripción del sistema en base a una serie de vistas [18] es tal que describe aquellos aspectos que han sido considerados más importantes durante el diseño: los datos, el flujo de datos y la interacción entre servicios (y componentes en la fase de implementación).

Las cualidades que se han planteado como fundamentales son la integrabilidad e interoperabilidad, comunes a la mayoría de las propuestas para bibliotecas basadas en servicios, que, sin embargo, contrastan con la prioridad que adquiere la eficiencia en la propuesta de manipulación de documentos comentada en la subsección 4.1.3. La diferencia de objetivos de este último caso respecto a la propuesta que se está describiendo (eficiencia frente a integrabilidad) da como resultado la utilización de modelos distintos y, por tanto, arquitecturas diferentes para ciertas funcionalidades coincidentes. Además, generalizando en lo que a las arquitecturas y modelos para *manipulación* de documentos se refiere, es éste un concepto tan amplio, cuyo significado varía de una biblioteca a otra, que hace imposible la existencia de un protocolo estándar que regule dichas operaciones, lo cual conlleva la obligatoria definición de un protocolo particular en cada caso, como ocurre en la propuesta realizada en esta tesis.

La importancia de los servicios de nombres en los documentos es fundamental en esta tesis, y coincide en esto con los trabajos sobre referencias entre artículos [31, 94]. En este entorno se han realizado varias propuestas para "servicios de enlaces" (*link services*) [67]. Algunos de ellos se han incluido en esta propuesta bajo la denominación de "Detección de referencias", ya que es más precisa y permite establecer la distinción con otros "servicios de enlaces", como los que se ocupan de su mantenimiento [97] o los que facilitan la creación manual por parte del usuario de enlaces [32].

Las relaciones entre documentos y su modelización como enlaces que no tienen por qué ser insertados manualmente por los autores de los documentos es una de las principales aportaciones. Hasta ahora los enlaces han recibido un tratamiento secundario, quedando relegados a meras herramientas que facilitan la navegación en las bibliotecas; la consulta de estas relaciones y su explotación en procesos avanzados (como la generación de versiones) abre enormes posibilidades de extracción de información. La coexistencia de versiones es uno de los aspectos más interesantes en las bibliotecas digitales y que menos atención ha recibido. Aquí adquiere protagonismo, y se propone la generación de dichas versiones, en lugar de la coexistencia elegida en los trabajos examinados, con la intención de evitar los problemas de mantenimiento de las versiones y enlaces derivados de esta coexistencia. La factibilidad de dicha operación depende directamente de otro de los servicios aportados: la traducción de documentos en base a su semántica.

Un modo alternativo de modelar dichas relaciones hubiera sido en los metadatos de los documentos; este es el caso, por ejemplo, si se utiliza el elemento `Relation` del estándar *Dublin Core* [23]. En esta tesis se ha decidido no hacerlo así para facilitar la creación dinámica de documentos: los documentos que fluyen por el sistema pueden ser similares, aunque distintos (mismo contenido y distintas relaciones, según la petición hecha por el usuario). Por ejemplo, las diferentes versiones históricas de un mismo documento comparten el contenido (documento original al cual se aplican las modificaciones), pero difieren en sus enlaces (las modificaciones aplicadas a un documento en un momento t_p posterior a otro t_a suelen incluir las hechas en t_a además de las realizadas en el intervalo temporal $[t_a, t_p]$). Más razones que justifican esta decisión son las que se dan en el capítulo 3.

Se ha decidido no almacenar las copias de los documentos virtuales generados (versiones, por ejemplo) por varias razones: primero, no es el objetivo de esta tesis ocuparse de la actualización incremental de las colecciones en las bibliotecas digitales. En segundo lugar, hay evidencia [38, 15] de que dicha política puede llevar asociados varios problemas, relacionados con la cantidad de documentos almacenados, la validación de documentos y la manipulación de versiones. En particular, en lo que concierne a la manipulación de versiones esta tesis sigue una política completamente diferente: almacenar los enlaces, que pueden ser utilizados en dicha generación, además de para otras finalidades como, por ejemplo, realizar consultas sobre las relaciones. Aparte de la multiplicidad de usos que se pueden hacer de los enlaces se evitan así los problemas mencionados al principio de este párrafo.

Por último, las cualidades de la arquitectura ya descritas se pueden completar con la posibilidad de una distribución de los componentes de la biblioteca, lo cual no supondría la modificación de la arquitectura propuesta, si no más bien su compleción con nuevos elementos que administren los aspectos relacionados con la distribución.

5

El prototipo

Índice General

El prototipo en el cual se han implementado las propuestas de esta tesis es una biblioteca de documentos legislativos. Se han implementado algunos de los componentes propuestos en el capítulo sobre la arquitectura (capítulo 4). El trabajo se ha centrado en los componentes y los servicios relacionados con los algoritmos explicados en detalle en los capítulos 2 y 3. Se ha probado pues, la obtención de una copia de un documento estructurado cuya estructura lógica refleje la estructura semántica del documento abstracto, así como la creación y explotación de bases de enlaces donde se aprovechen las relaciones derivadas de referencias y modificaciones. El estándar elegido para modelar toda esta información ha sido XML (junto a sus estándares asociados), dado que proporciona algunos de los requisitos imprescindibles para que sea factible obtener un prototipo: acceso a fragmentos en documentos estructurados, etiquetado extensible, posibilidad de modelar y manipular los enlaces a voluntad, además de la disponibilidad de herramientas que proporcionan estas funcionalidades (*parsers*, interfaces estándar, lenguajes de manipulación, etc.).

Los documentos legislativos presentan un conjunto de propiedades que los convierten en la base de prueba idónea para las propuestas de esta tesis. Tienen una fuerte estructura semántica asociada al contenido, y además los textos normativos son objeto de frecuentes modificaciones que dan lugar a nuevas versiones de los documentos modificados.

Por otra parte, el acceso a las relaciones (por ejemplo, la obtención de todas las jurisprudencias relacionadas con una determinada ley) es importante para los especialistas que utilizan estos documentos. También lo es el acceso a todas las versiones de un documento (por ejemplo, para entender una sentencia de un tribunal es necesario disponer del texto de todas las leyes implicadas, tal como se encontraban en el momento que la sentencia fue dictada). Finalmente, otra característica de los documentos legislativos es que las modificaciones están incluidas dentro de otros documentos, de forma que primero aparece citado el documento a modificar, y más tarde se describe cómo deberá ser modificado. Un documento puede modificar varios documentos, así como las modificaciones que afectan a un documento pueden provenir de distintas fuentes.

5.1 Las interfaces de los componentes

Esta sección presenta las interfaces para los componentes[1] presentados en el capítulo 4. Los parámetros de entrada y datos de salida se han tenido en cuenta en los métodos de los componentes. Se incluye también una revisión de algunos escenarios presentados en el capítulo 4, implementados en el prototipo, donde se han utilizado algunos componentes que, aunque no son una aportación de esta tesis, son necesarios dado que se utilizan para implementar varios de los componentes del prototipo; por ejemplo, los componentes que implementan los servicios de búsqueda y manipulación de documentos utilizan un analizador XML (*parser XML*). Para la revisión de los escenarios se ha elegido la evolución de tareas y el flujo de datos entre los componentes. Los datos (documentos) son piezas fundamentales en este trabajo y también la razón de ser de una biblioteca; su disponibilidad, interoperabilidad y fácil manipulación son,

[1] (Notación UML.)

por tanto, requisitos imprescindibles. Los componentes están implementados como objetos. Todos los componentes y datos intercambiados en el sistema están en formato XML, que proporciona el máximo nivel de interoperabilidad a este nivel. Dado que la herramienta de acceso más popular en este momento son los navegadores Web, los documentos se transforman para su presentación al usuario a equivalentes HTML que cualquier navegador sea capaz de visualizar.

En el prototipo, el tipo `XMLdoc` designa un documento definido según se ha descrito en el capítulo 4, con tres elementos internos: los metadatos, el contenido y los enlaces que afectan al documento. Se trata de una cadena, que contiene 3 subcadenas, todas ellas datos XML. Los tipos `id-list` y `link-collection` son enumeraciones de elementos. La lista de identificadores es una lista de cadenas, donde cada item es un identificador de documento. La lista de enlaces contiene enlaces Xlink (xlinks), que cumplen las características mostradas en el capítulo 3. El LDI es el identificador de documentos que se resuelve para obtener la URL que designa la copia física del documento. La biblioteca está localizada en un servidor, lo que aquí significa que el universo donde el LDI es único coincide con el dominio del servidor.

A continuación se muestra de nuevo el diagrama de componentes (figura 5.1), y las interfaces del prototipo:

Servidor de documentos

- **GetDocument(id:LDI):XMLdoc**

  Recibe un identificador de documento (LDI). Devuelve la versión original (la versión disponible en la base de datos) del documento. Los enlaces que se adjuntan son aquellos con origen en este documento.

- **GetDocument(id:LDI, date:String):XMLdoc**

  Recibe un identificador de documento (LDI) y la fecha de la versión del documento demandado. Devuelve un documento, cuyo contenido es la versión correspondiente a la fecha indicada, y los enlaces son aquellos con origen en él.

- **GetReferences(id:LDI, type:String):XMLdoc**

  Recibe un identificador de documento y el tipo de referencia deseada. Devuelve una lista de enlaces relativos al documento `id`, de los tipos (`type`) seleccionados. El tipo puede ser o bien "referencia" (`citation`), "sustitución" (`substitution`), "inserción" (`insertion`) o "borrado" (`deletion`).

- **QueryDocuments(query:String):id-list**

  Recibe una consulta (`query`) como entrada. Devuelve la colección de identificadores de documentos que responden a la consulta recibida como argumento de entrada.

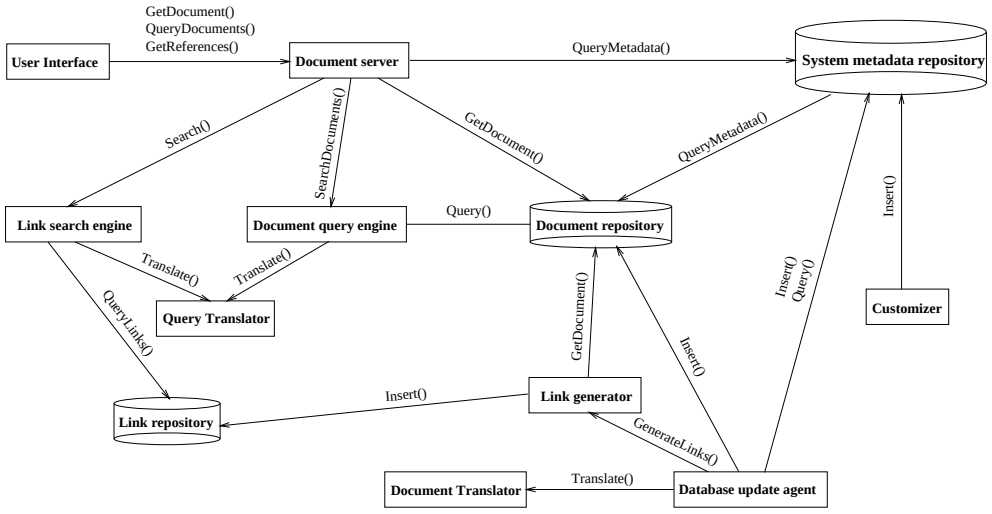


Figura 5.1: Interacción entre componentes.

Herramienta de búsqueda en documentos

Este componente oculta la herramienta de búsqueda (un indexador XML [54]) así como los índices que ésta utiliza.

- **Search(query:String):id-list**

  Recibe una consulta (`query`) como argumento de entrada. Busca en el documento de la base de datos documentos que cumplan esta consulta. Devuelve la lista de identificadores correspondientes al conjunto de documentos que encajan con la petición.

Herramienta de búsqueda en enlaces

- **Search((attribute:String,value:String)+):xlink-collection**

  Recibe una consulta que consiste en una lista de atributos y valores para los enlaces. Devuelve una lista, con los enlaces que cumplen todos los criterios descritos en la entrada. Los criterios están especificados como una secuencia de parejas (*atributo, valor*) y se presupone un operador AND entre los elementos de la secuencia.

Generador de enlaces

- **GenerateLinks(id:LDI):xlink-collection**

  Recibe un identificador de documento como argumento de entrada. Genera una colección de enlaces XLink (xlink) a partir del análisis del documento de entrada. La colección de enlaces resultantes está compuesta por enlaces cuyo origen se encuentra en el documento `id`.

Colección de metadatos del sistema

- **ListComponents():String**

  No necesita argumentos. Devuelve la lista de componentes del sistema.

- **ListComponentMethods(component:Name):String**

  Devuelve la lista de métodos disponibles para el componente especificado.

- **ListCollections():String**

  Devuelve la lista de colecciones de documentos disponibles en la biblioteca.

- **ListDocumentTypes():String**

  Devuelve la lista de clases de documentos disponibles en las colecciones de documentos de la biblioteca.

- **DocTypeOntMapping(doc-type:String):Mapdoc**

  Recibe el nombre de una clase de documentos (`document class`). Devuelve un objeto con dos elementos: la equivalencia entre las ontologías utilizadas en los textos y la utilizada por el sistema para los documentos de la clase, y la jerarquía de inclusión entre los elementos de la clase. Esta información es utilizada por el componente que lleva a cabo la traducción del documento.

Repositorio de documentos

- **GetDocument(id:LDI):Stream**

  Recibe el identificador de un documento. Devuelve la copia digital del documento, tal como se encuentra en la base de datos.

- **GetDocMeta(id:LDI):String**

  Recibe el identificador de un documento. Devuelve los metadatos sobre el documento.

- **GetDocLinks(id:LDI):String**

  Recibe el identificador de un documento. Devuelve todos los enlaces con origen en este documento.

Repositorio de enlaces

- **SearchLinks(Query:String):link-list**

  Recibe una consulta y devuelve la lista de enlaces que responden a la consulta.

Traductor de documentos

- **Translate(in InputDocument:String, out OutputDocument:String, in OntologyMapping:MapDoc)**

  Recibe como entradas un documento y la equivalencia entre ontologías, necesarios para traducir el documento de entrada a un documento equivalente, con el mismo contenido pero diferente estructura lógica.

5.2 Revisión de los escenarios

Esta revisión de los escenarios presentada en el capítulo sobre la arquitectura (capítulo 4) considera componentes que no han sido diseñados en este prototipo, aplicaciones externas que son, sin embargo, necesarias para su funcionamiento. Tras los servicios presentados en el capítulo 4, se ocultan algunos de estos componentes; por ejemplo, los servicios de búsqueda (*Search*) corresponden a la labor que realizan herramientas de indexación y búsqueda. Los aspectos que centran la atención en esta revisión son la interacción entre componentes y los diagramas de actividades que muestran la evolución de las tareas en el sistema, así como el flujo de datos entre componentes. Otra diferencia con la presentación hecha en el capítulo 4, es que ahora se presta más atención al tipo (o al formato) de los datos y documentos intercambiados entre los componentes del sistema. Por ejemplo, se puede ver que para conseguir interoperabilidad en los datos se utiliza el estándar XML.

Los escenarios considerados son los implementados en el prototipo: Recuperación de un documento (tal cual está almacenado en el base de de documentos), recuperación de una versión que debe ser generada, consultas sobre relaciones, y traducción de documentos.

5.2.1 Recuperación de un documento

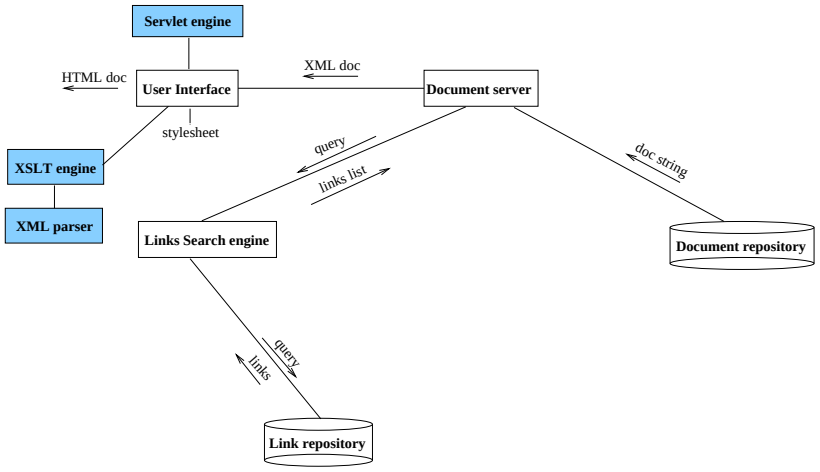
En este escenario (figura 5.2) se muestra la recuperación de un documento (y los enlaces con origen en el documento).

Este escenario presenta una operación similar a cualquier recuperación de documentos clásica, en un sistema donde éstos son presentados al usuario a través de un navegador. Se "preservan" los enlaces con origen en el documento, de manera que puedan ser utilizados para construir hipertexto navegable, si esa fuese la intención. No obstante, cualquier tipo de enlace (con origen o destino en el documento) se puede recuperar con estos métodos, con el único requisito para el usuario de especificar los criterios de selección de enlaces.

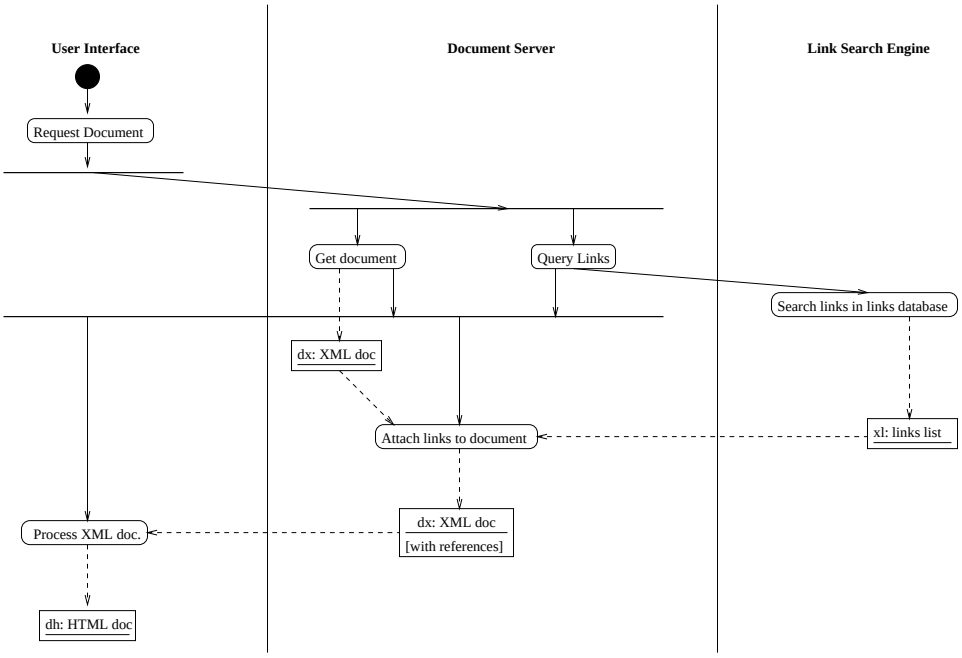
La consulta del usuario se pasa al servidor de documentos, que reacciona recuperando la versión del documento disponible en la base de documentos. Al mismo tiempo, solicita a la herramienta de búsqueda en enlaces todos los enlaces de tipo `citation` con origen en el documento. Una vez que el servidor dispone de estos dos elementos, puede construir y enviar un `XMLdoc` (contenido + enlaces) a la interfaz de usuario. Esta última aplica al documento recibido una hoja de estilos con la cual se obtiene la copia HTML que el usuario verá en su navegador.

5.2.2 Generación de la versión de un documento

Este escenario (figura 5.3) muestra cómo recuperar versiones de un documento. La consulta del usuario (que incluye el identificador del documento y la fecha de la versión requerida) se pasa al servidor de documentos, que obtiene las versiones disponibles en la base de documentos. También envía una consulta a la herramienta de búsqueda de enlaces con el fin de conseguir todos los enlaces de modificación que afectan al documento, y cuya fecha esté en el límite especificado. Una vez el servidor tiene ambos



(a) Componentes y flujo de datos entre ellos.



(b) Evolución de tareas.

Figura 5.2: Escenario correspondiente a la recuperación de un documento.

(documento y enlaces), puede crear la versión actualizada aplicando los enlaces a la copia obtenida (este método sigue el algoritmo descrito en la sección 3.7 del capitulo 3). La composición de esta versión puede requerir la recuperación de fragmentos de otros documentos, lo cual está incluido en el diagrama en la tarea `Process links on document`. El paso final corresponde a la interfaz de usuario, que recibe la versión actualizada (documento XML) y genera el equivalente HTML que el usuario visualizará.

5.2.3 Consultas sobre las relaciones

Algunas veces la información requerida se encuentra dentro de *documentos que referencian otro documentos* (por ejemplo, la jurisprudencia relacionada con un determinado texto normativo). Esta funcionalidad está descrita en los diagramas de la figura 5.4. Al igual que en la generación de versiones, es necesaria una búsqueda en los enlaces para obtener la lista de documentos relacionados con el documento sobre el cual se está consultando. Sin embargo, los criterios para filtrar enlaces son diferentes: en este caso son los enlaces de tipo `citation` los que interesan. Otra diferencia es que el documento no es necesario para responder a la consulta, razón por lo cual no se recupera.

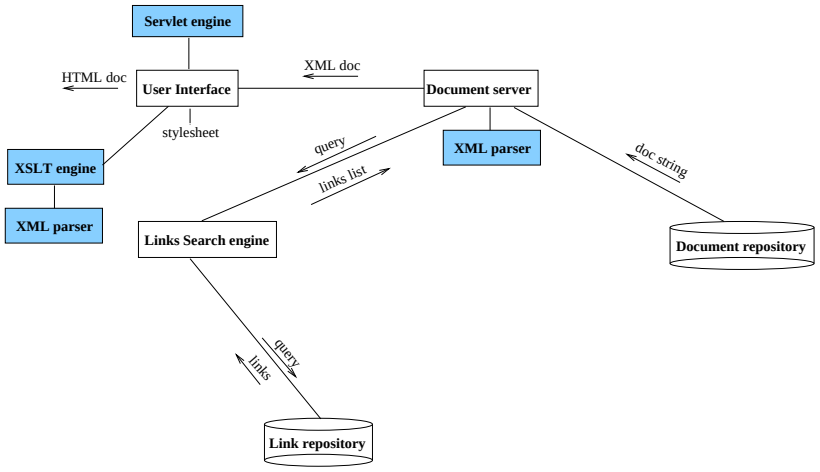
5.2.4 Búsqueda de documentos

Este método permite la búsqueda por términos clave en el contenido de documentos (consulta clásica). Un indexador o herramienta de búsqueda consulta la colección de documentos y devuelve el resultado al agente que le ha invocado (el servidor de documentos). En el diagrama (figura 5.5) el indexador está representado por una herramienta de búsqueda genérica (`Document Query Engine`), que permite abstraerse de sus peculiaridades.

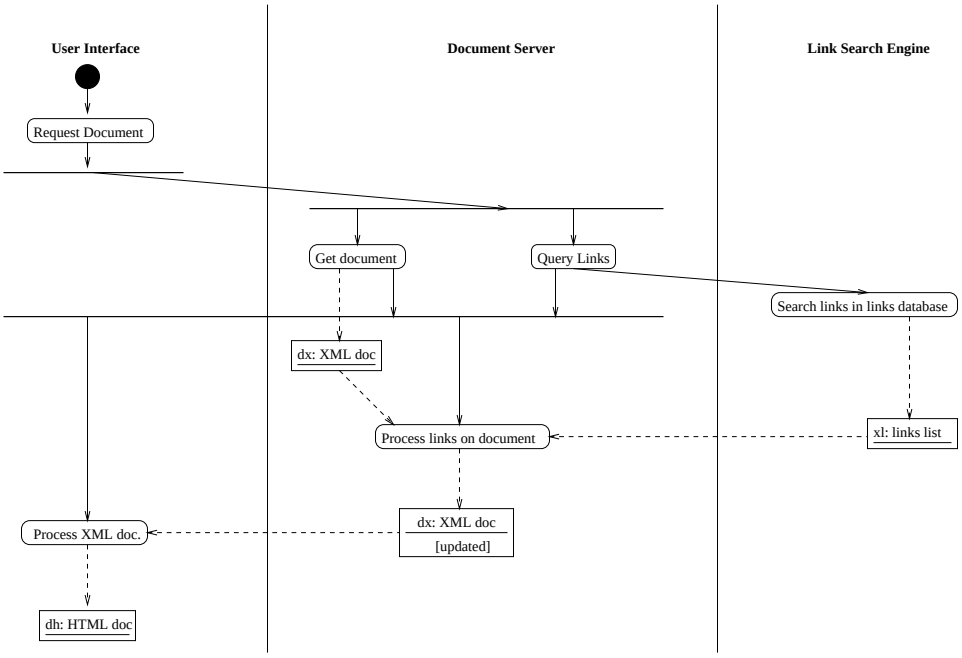
5.3 Las bases de documentos

Los documentos utilizados en el prototipo son de tipo legislativo. Dentro de ellos, se han considerado cuatro clases, que reflejan la clasificación semántica establecida por los juristas: *textos normativos*, *jurisprudencia*, *dossiers* y *comentarios*. Los textos normativos y la jurisprudencia son los más interesantes desde el punto de vista semántico, puesto que se ajustan a rígidas estructuras semánticas, las cuales están presentes en las referencias entre documentos y son, por consiguiente, cruciales para la detección de enlaces y generación de versiones. Son, por tanto, los candidatos ideales para experimentar la extracción de estructura lógico-semántica (traducción de documentos). Las dos últimas clases (dossiers y comentarios) admiten divisiones mucho más flexibles (tanto en los tipos de elementos aceptados como en la jerarquía de inclusión), lo cual hace preferible utilizar otras DTD estándar, como la TEI [48] que aportan los elementos y atributos necesarios para conseguir esta flexibilidad.

La subsección 5.3.1 incluye una breve descripción de la estructura lógica de cada una de estas clases. Los resultados de la experimentación de la traducción de documentos sobre un conjunto de textos normativos se comentan en la subsección 5.3.2.

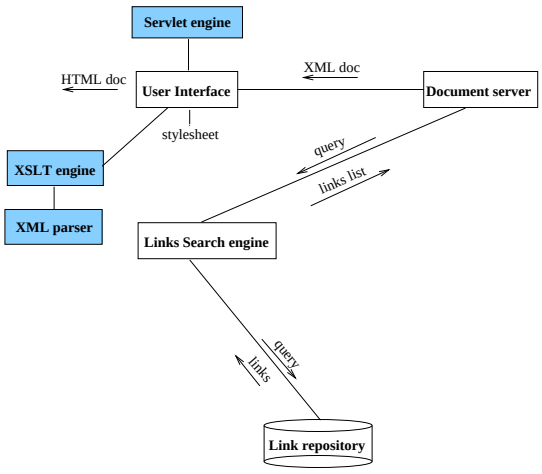


(a) Componentes y flujo de datos entre ellos.

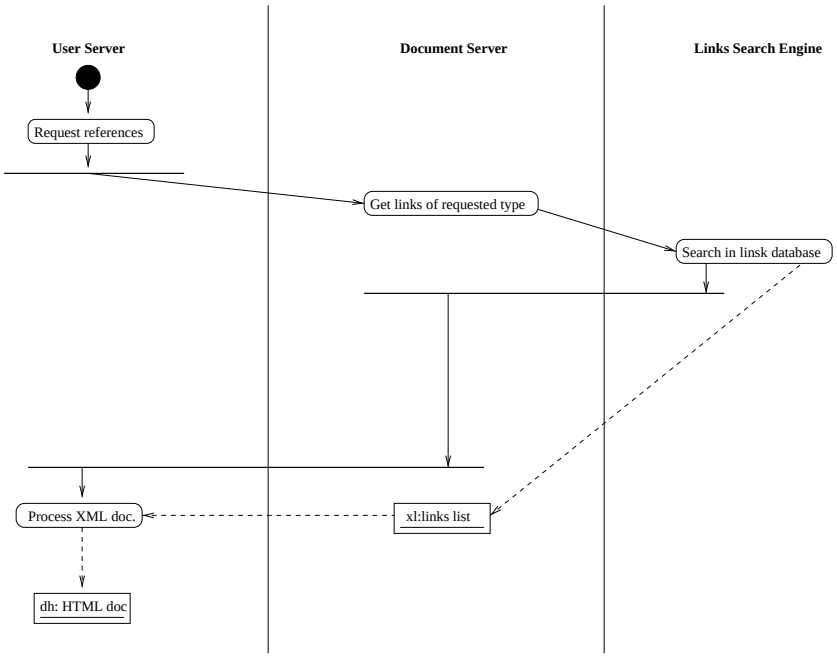


(b) Evolución de tareas.

Figura 5.3: Escenario correspondiente a la generación de versiones.

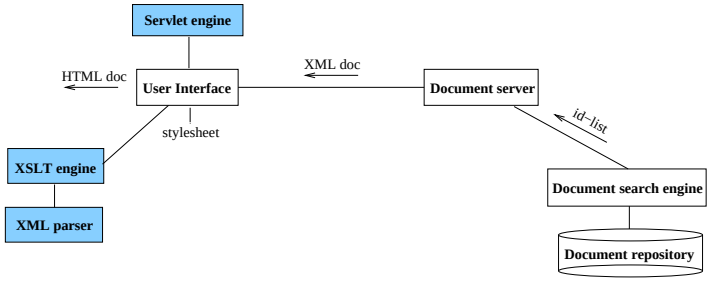


(a) Components and data flow between components.

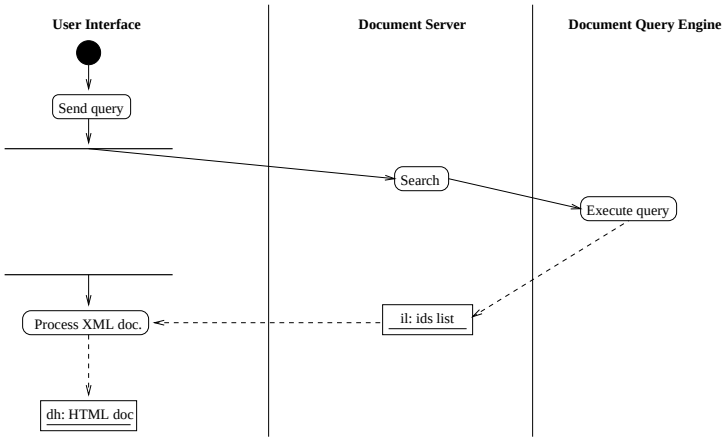


(b) Task evolution.

Figura 5.4: Escenario correspondiente a la consulta de relaciones.



(a) Componentes y flujo de datos entre ellos.



(b) Evolución de tareas.

Figura 5.5: Escenario correspondiente a la búsqueda en documentos.

5.3.1 Clases de documentos en una biblioteca legislativa

La biblioteca digital del prototipo contiene cuatro clases de documentos:

1. **Textos normativos**: La Constitución española, leyes, decretos, ...
2. **Jurisprudencia**: sentencias relacionadas con textos normativos, ya que están basadas en alguno de ellos.
3. **Comentarios** sobre normas o jurisprudencia.
4. **Dossiers** consistentes en recopilaciones de referencias a textos normativos y jurisprudencia, clasificadas según criterios particulares del autor de cada dossier.

Existe otra clase de documentos en el sistema de la cual el usuario no tiene consciencia, correspondiente a los metadatos que describen los documentos (*headers*).

Los **textos normativos** son documentos oficiales, que agrupan sus elementos siguiendo la definición y reglas de inclusión que se encuentran a continuación. Los elementos que pueden aparecer en un texto normativo son:

- *denominación.* Es el título del documento.
- *exposición de motivos.* Compuesto por una secuencia de párrafos situados entre el título del documento y el primer elemento que aparezca (entre los que se enumeran a continuación).
- *libro.* Contiene cualquier secuencia de elementos de tipo *titulo*, *capitulo*, *seccion* o *articulo*, en cualquier orden. Puede haber cero o más elementos de este tipo.
- *titulo.* Contiene cualquier secuencia de elementos de tipo *capitulo*, *seccion* o *articulo*, en cualquier orden. Puede haber cero o más elementos de este tipo.
- *capitulo.* Contiene cualquier secuencia de elementos de tipo *seccion* o *articulo*, en cualquier orden. Puede haber cero o más elementos de este tipo.
- *seccion.* Contiene cualquier secuencia de elementos de tipo *articulo*, en cualquier orden. Puede haber cero o más elementos de este tipo.
- *articulo.* Un texto normativo debe tener al menos un elemento de tipo *artículo* para ser considerado como tal. Puede haber uno o más elementos de este tipo.

Otro elemento, que los juristas no explicitan cuando describen la semántica de estos documentos, pero implícito en cualquiera de ellos, y que además se tiene en cuenta en las referencias es el *párrafo.*

No existen términos clave que permitan reconocer automáticamente los límites del elemento semántico *exposición de motivos.* Por ello, desaparece como tal en la clase utilizada en el prototipo, para ser reemplazado por la posible aparición de uno o más párrafos consecutivos. Las reglas gramaticales que expresan la inclusión jerárquica entre elementos se muestran en la figura 5.6. También en el árbol de la figura 5.7: todos los hijos o descendientes de un nodo son los tipos de elementos que se permite incluir dentro de algún elemento del tipo representado por el nodo.

$$
\begin{array}{lcl}
< norma > & ::= & < p >^*(< libro > | < titulo > | \\
 & & < capitulo > | < seccion > | < articulo >)^+ \\
 & & < disposicion >^* \\
< libro > & ::= & < title >?(< titulo > | < capitulo > | < seccion > | \\
 & & < articulo >)^+ \\
< titulo > & ::= & < title >?(< capitulo > | < seccion > | < articulo >)^+ \\
< capitulo > & ::= & < title >?(< seccion > | < articulo >)^+ \\
< seccion > & ::= & < title >? < articulo >^+ \\
< articulo > & ::= & < title >? < p >^+ \\
< disposicion > & ::= & < title >? < p >^+
\end{array}
$$

Figura 5.6: Gramática de los textos normativos españoles.

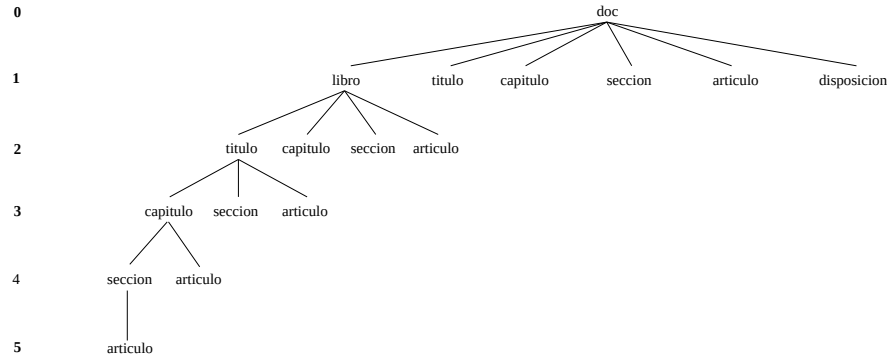


Figura 5.7: Jerarquía de inclusión entre elementos de la clase "texto normativo español". La representación es parcial: el árbol sólo está completamente expandido en la rama situada más a la izquierda.

La **jurisprudencia** también son documentos oficiales, con estructura rígida, pero más simple que la de los textos normativos. No hay aspectos especialmente relevantes que comentar acerca de su estructura lógica, cuya gramática y árbol asociado se muestran en las figuras 5.8 y 5.9 respectivamente.

Los **metadatos** (*headers*) asociados a los documentos legales se componen de los campos listado a continuación. La gramática que describe esta clase se muestra en la figura 5.10. Tanto el identificador de documento como el tipo de documento y su año son metainformación necesaria en los enlaces, que será usada por el algoritmo de generación de versiones. En cuanto a los boletines (`boletines`), son información interesante por los juristas. Así pues, los elementos permitidos en estos documentos son:

- *identificador* (`ldi`).
- *clase* (`type`).
- *título* del documento (`titulo`).
- *año* de publicación (`año`).

$$
\begin{array}{ll}
< jurisprudencia > ::= & < inicio >< fund - hecho >? < fund - derecho >< fallo > \\
< inicio > ::= & < p >^{+} \\
< fund - hecho > ::= & < p >^{+} \\
< fund - derecho > ::= & < p >^{+} \\
< fallo > ::= & < p >^{+}
\end{array}
$$

Figura 5.8: Gramática para la clase "jurisprudencia".

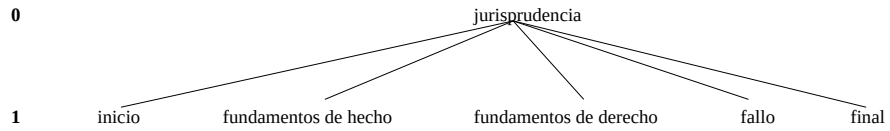


Figura 5.9: Jerarquía de inclusión entre elementos de la clase "jurisprudencia".

- *boletines* donde aparece (`boletines`)[2].
- *Fecha* de publicación (`fecha`).

5.3.2 La traducción de documentos

El algoritmo de traducción del capítulo 2 ha sido probado con un conjunto de textos normativos, procedentes de diferentes servidores públicos. Estos textos están marcados con diferentes etiquetas, dependiendo de su origen. Por ejemplo, mientras algunos documentos presentan una secuencia de etiquetas `<center><h3>` antes de los elementos semánticos de tipo `articulo`, otros delimitan este tipo de elementos con una secuencia de etiquetas `<p><b>`. El primer paso consistió en la eliminación de todos los elementos que no son semánticamente representativos (enlaces, índices, etc.). De este modo se consiguió que el grupo de documentos sobre el que se experimentó el algoritmo de traducción cumpliese los requisitos de entrada explicados en la sección 2.4 del capítulo 2. La traducción se probó sobre un conjunto de 1665 documentos. En 1583 casos el algoritmo obtuvo resultados satisfactorios; es decir, fue posible obtener una copia con estructura lógico-semántica en un 95% de los casos. Los documentos que dieron problemas presentaban al menos alguna de las características siguientes:

- Contienen tablas que no cumplen las normas XML.
- Contienen caracteres de entrada o entidades no reconocidas (`&icute;`, `&ordm;`, etc.).
- No contienen un elemento raíz.

[2]Este campo tiene sentido en los textos normativos.

$$
\begin{array}{ll}
< cabecera >::= & < ldi >< type >< titulo - doc >< año >< boletines >? \\
< boletines >::= & < boletin >^{+} \\
< boletin >::= & < numero >< fecha >< pgnas > \\
< fecha >::= & < dia >< mes >< año >
\end{array}
$$

Figura 5.10: Gramática de los metadatos.

5.4 Relaciones y enlaces en el prototipo

Como se ha dicho, los documentos jurídicos están muy relacionados entre sí. La jurisprudencia tiene referencias a leyes, las leyes referencian otras leyes y los comentarios lo son a cualquier otro tipo de documento. Otro aspecto, incluso más interesante, es el de las modificaciones que dan lugar a nuevas versiones. Esto ocurre a menudo en los textos normativos, donde se encuentran dos relaciones próximas:

1. Una referencia al documento a modificar, donde el fragmento afectado está claramente identificado, que dice explícitamente si el fragmento referenciado debe ser sustituido por otro, eliminado o sufrir una inserción.

2. El fragmento que debe ser insertado, o que sustituirá al referenciado previamente.

Esto convierte a estos documentos en el conjunto ideal para experimentar las propuestas de explotación de relaciones de referencia, modificación y eliminación. En la subsección 5.4.3 se presenta un ejemplo donde se aplica una sustitución.

5.4.1 Los atributos en los enlaces Xlink

La equivalencia entre los campos necesarios en los enlaces y los atributos utilizados en los enlaces xlink se encuentra en la tabla 5.1. El direccionamiento de un documento y el fragmento afectado por un enlace se modela dentro del campo `xlink:href`, el tipo del enlace en un atributo `xlink:role` y los campos definidos expresamente en esta tesis se modelan en atributos propios (`doctype` y `date`). Todos los atributos que aparecen en la DTD asociada a los enlaces se explican en las tablas 5.2 a 5.4. La figura 5.14 muestra un ejemplo de enlace. Cada enlace está representado como un elemento `enlace`, que tiene tres subelementos: el origen (`origen`), el destino (`destino`), y el arco (`arco`) que los conecta. El elemento `origen` representa el nodo que contiene la referencia o la modificación. El elemento `destino` es el nodo afectado por la modificación o referencia. En ambos casos, puede tratarse de un elemento o de un fragmento de texto dentro de un elemento. El elemento `arco` establece puentes entre los vértices. Los arcos pueden ser de diferentes tipos, correspondientes a las diferentes clases de relaciones existentes entre los vértices del grafo. Existen tantos tipos de arcos diferentes como tipos de relaciones relaciones. El tipo de enlace proviene del papel que juega el origen del enlace en la relación; por esto, el tipo del enlace se expresa en el atributo que describe el papel que desempeña el nodo origen en la relación.

Link field	*XLink attribute*
Document ID Internal locator	xlink:href (*)
Document type	doctype
Date	date
Link type	xlink:role (*)

Tabla 5.1: Correspondencia entre los campos de los enlaces y los atributos de los elementos xlink. Los atributos con un asterisco están predefinidos en el dominio de XLink. Los atributos sin asterisco están semánticamente definidos de forma local.

Attribute	*Value*	*Description*
xlink:href	?	Document ID + fragment locator
xlink:type	'locator'	Resource of an extended out-of-line link
xlink:role	('citation' | 'substitution' | 'insert' | 'delete')	Type of link
string	?	String inside the origin. It is the real origin in citations.
date	?	Document date
doctype	?	Document class

Tabla 5.2: Atributos para el origen de un enlace. El carácter '?' indica que se acepta cualquier cadena.

Attribute	*Value*	*Description*
xlink:href	?	Document ID + fragment locator
xlink:type	'locator'	Extended out-of-line link
xlink:role	'target'	It is the target of the link
string	?	String inside the target. It is the real target.
date	?	Document date
doctype	?	Document type

Tabla 5.3: Atributos para el destino de un enlace. El carácter '?' indica que se acepta cualquier cadena.

Attribute	*Value*	*Description*
xlink:from	?	Origin of the arc
xlink:to	?	Target of the arc
xlink:type	'arc'	It is an arc element

Tabla 5.4: Atributos para el arco de un enlace. El carácter '?' indica que se acepta cualquier cadena.

5.4.2 La influencia del tipo de documento en las relaciones de documentos

La naturaleza de los documentos puede determinar el tipo de relaciones permitidas entre dos documentos, prohibiendo ciertos tipos de relaciones entre clases de documentos.

Los tipos de documentos existentes en un sistema de información legal son:

1. Textos normativos.
2. Jurisprudencia.
3. Bibliografía.
4. Comentarios, notas, artículos y otros.

Las restricciones aplicables a las relaciones entre clases son las siguientes:

- Un texto normativo puede citar y modificar otros textos.
- Los comentarios, la jurisprudencia y la bibliografia pueden citar, pero *no* pueden modificar textos normativos.
- La jurisprudencia, la bibliografía, y otros documentos pueden tener referencias a documentos de alguna de las clases anteriores.

De esto se deduce que, en el proceso de obtención de versiones de un texto normativo, pueden participar (modificarlo) únicamente otros textos normativos.

5.4.3 Un ejemplo

El ejemplo considerado a continuación fue presentado en la sección 3.7 del capítulo 3. Los documentos de entrada al proceso de actualización son los textos que se muestran en la figura 5.11:

- Un documento fuente, `lo2-1980.xml`, cuyo primer elemento `articulo` debe ser reemplazado. El texto se encuentra en la figura 5.11(a).
- Un documento modificador, `l13-1986.xml`, que contiene el elemento que debe sustituir al artículo de `lo2-1980.xml`: el primer `articulo` dentro del primer elemento `disposicion`. El texto de este documento se muestra en la figura 5.11(b).

El documento de salida del proceso de modificación (en la figura 5.11(c)) es una nueva versión del documento fuente, donde el primer conjunto de nodos que constituye el elemento `articulo` en `lo2-1980.xml` se ha reemplazado como se ha indicado.

La figura 5.12 muestra el efecto de este proceso en el árbol de documento. En la figura 5.13 se puede ver el enlace que modela la modificación[3]. De acuerdo con lo comentado en el capítulo 3, se trata de datos XML. El elemento ORIGEN de la figura representa el vértice origen del enlace y DESTINO se corresponde con el vértice afectado por la modificación.

[3]Los atributos a cuya presencia obliga la especificación XLink (por ejemplo, `xlink:type`) no aparecen en la figura, ya que sus valores por defecto están declarados en la DTD de los enlaces

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<doc>
<articulo id="a1"><title>Artículo Primero. </title>
<p>El referendum en sus distintas modalidades, se celebrará de
acuerdo con las condiciones y procedimientos regulados en la
presente Ley Orgánica.</p>
</articulo>
<articulo id="a2"><title>Artículo Segundo. </title>
<p>Uno. La autorización para la convocatoria de consultas populares
por vía de referendum en cualquiera de sus modalidades, es competencia
exclusiva del Estado.</p>
</articulo>
</doc>
```

(a) Documento de entrada: *lo2-1980.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<doc>
<p>Ley 13/1986, de 14 de Abril de 1986, de Fomento y Coordinación
General de la Investigación Científica y Técnica</p>
<p>Don Juan Carlos I,Rey de España.</p>
<disposicion id="da"><title>DISPOSICIONES ADICIONALES. </title>
<p><a>Undécima.</a>
 1. Quedan modificados los artículos 1.°, 4.° y 8.° de la Ley
Orgánica 2/1980, de 30 de abril , que quedarán redactados en la
forma siguiente :</p>
<articulo id="da111"><title>Artículo 1.</title>
<p>Con la denominación de Instituto de Astrofísica de Canarias se crea
un Consorcio Público de Gestión , cuya finalidad es la investigación
astrofísica .</p>
<p>El Instituto de Astrofísica de Canarias estará integrado por la
Administración del Estado, la Comunidad Autónoma de Canarias la
Universidad de La Laguna y el Consejo Superior de Investigaciones
Científicas .</p>
</articulo>
<articulo id="da112"><title>Artículo 4.</title>
<p>El Consejo Rector estará integrado por el Ministro de Educación y
Ciencia , que actuará como Presidente; un Vocal en representación de la
Administración del Estado, que será nombrado a propuesta del
Ministerio de la Presidencia , y tres Vocales más en representación de
cada una de las restantes Administraciones públicas y Organismos que
se relacionan en el artículo 1.° Formará parte del Consejo Rector,
asimismo, el Director del Instituto , que será miembro nato.</p>
</articulo>
</disposicion>
</doc>
```

(b) Documento que contiene la modificación: *l13-1986.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<doc>
<articulo id="da111"><title>Artículo 1.</title>
<p>Con la denominación de Instituto de Astrofísica de Canarias se crea
un Consorcio Público de Gestión, cuya finalidad es la investigación
astrofísica .</p>
<p>El Instituto de Astrofísica de Canarias estará integrado por la
Administración del Estado, la Comunidad Autónoma de Canarias la
Universidad de La Laguna y el Consejo Superior de Investigaciones
Científicas .</p>
</articulo>
<articulo id="a2"><title>Artículo Segundo. </title>
<p>Uno. La autorización para la convocatoria de consultas populares
por vía de referendum en cualquiera de sus modalidades, es competencia
exclusiva del Estado.</p>
</articulo>
</doc>
```

(c) Documento modificado: versión actualizada de *lo2-1980.xml*

Figura 5.11: Generación de versiones. Documentos de entrada y salida.

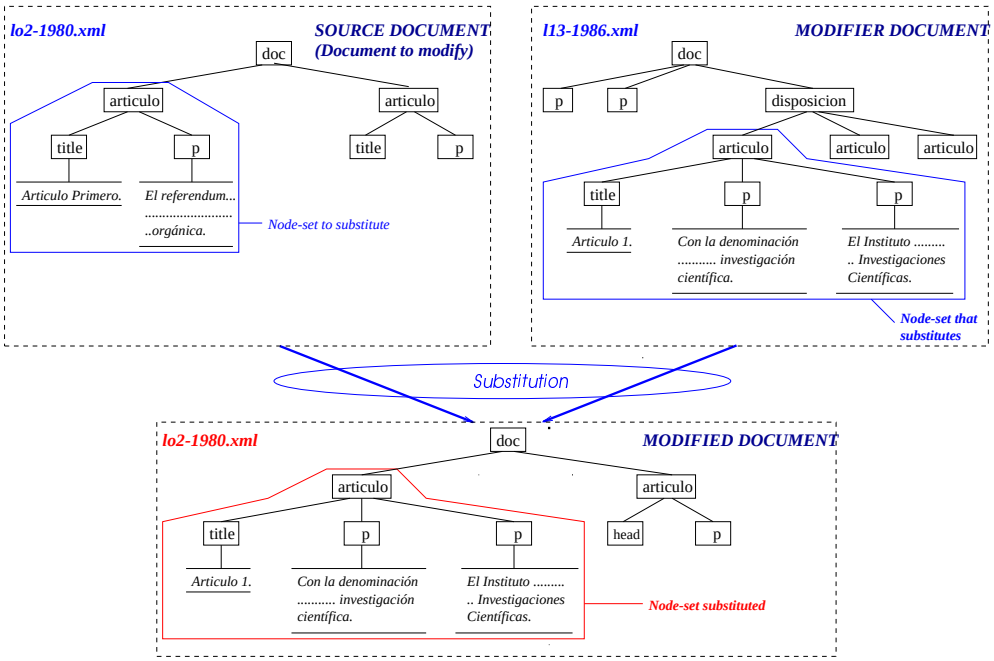


Figura 5.12: Sustitución de elementos, utilizando enlaces. El primer elemento `articulo` del documento `lo2-1980.xml` es sustituido por el primero elemento `articulo` que forma parte del primer elemento `disposicion`, dentro del documento `l13-1986.xml`. El resultado es una nueva versión del documento `lo2-1980.xml`.

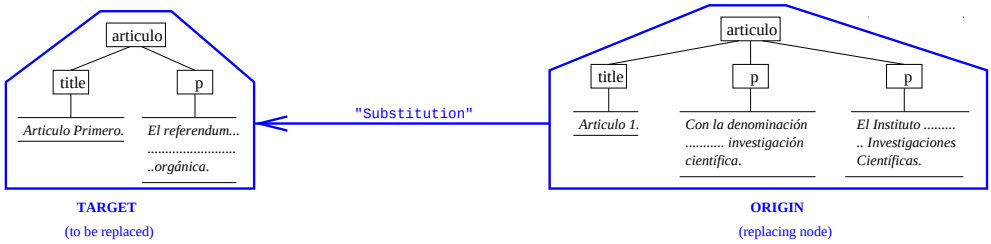


Figura 5.13: Enlace de sustitución. El origen (ORIGIN) reemplaza al destino (TARGET) cuando se genera una nueva versión del documento fuente. El origen es un subárbol del documento fuente formado por el elemento `articulo` y todos sus descendientes. El destino es el subárbol con raíz en el primero `articulo` dentro del primer elemento de tipo `disposicion`, tal como se ve en la figura 5.14.

```
<ENLACE>
<ORIGEN  xlink:href= ‘‘113-1986.xml#xpointer(child::disposicion[1]/articulo[1])’’
         xlink:role=’’substitution’’
         date= ‘‘1981’’
         doctype= ‘‘norma’’   />
<DESTINO xlink:href= "lo2-1980.xml#xpointer(child::articulo[1])"
         xlink:role=’’target’’
         date= ‘‘1986’’
         doctype= ‘‘norma’’   />
  <ARCO    xlink:from=’’substitution’’ xlink:to=’’target’’
         xlink:show=’’undefined’’    xlink:actuate=’’undefined’’/>
</ENLACE>
```

Figura 5.14: Texto del enlace XML (xlink) del ejemplo.

5.5 La generación de versiones

La generación de versiones se ha probado sobre textos normativos, que son los que sufren mayor número de modificaciones.

5.5.1 Tipos de datos en el proceso de generación

Las bases de datos que intervienen en el proceso de generación son las de documentos normativos y la base de enlaces. La figura 5.15 es un extracto de la arquitectura de componentes del capítulo 5, focalizada en las bases de datos utilizadas para la generación de documentos virtuales; el *Document resolver* representa la colaboración de los componentes que genera las versiones de documentos (*virtual documents*).

- De la *base de documentos* se extraen aquellos que participan en el proceso de generación de versiones: aquel al que se aplican las modificaciones y los que contienen los textos que se utilizan en cada una de ellas.
- De la *base de enlaces* (*links database*) se obtienen los enlaces que expresan las relaciones de modificación entre los documentos.

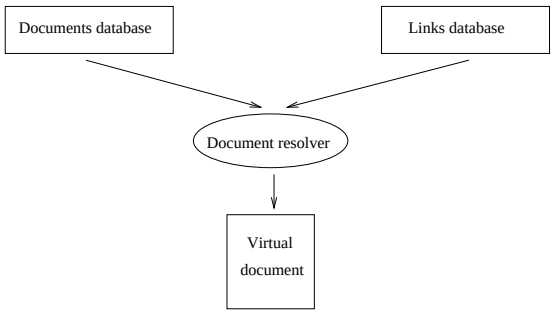


Figura 5.15: Generación de un documento virtual.

5.5.2 La implementación del algoritmo

Para aplicar las modificaciones a un documento es necesario disponer de la información acerca de aquellas que le afectan. Esto significa que para reconocer los nodos afectados por una sustitución se debe realizar una fase de búsqueda de enlaces anterior al tratamiento del documento fuente. De este modo, el proceso completo que conduce a la obtención de una nueva versión se lleva a cabo en dos pasos consecutivos:

1. Búsqueda de enlaces.
2. Aplicación de enlaces al documento fuente.

Paso 1: búsqueda de enlaces

Esta fase consiste en la creación de una variable de enlaces que será utilizada durante el tratamiento de los nodos del documento fuente. Esta variable se crea a partir de los resultados de las consultas a la base de enlaces.

El algoritmo 4 filtra los enlaces y recupera los vértices de los enlaces, que inserta en una variable de SUSTITUCIONES. Como resultado, esta variable contiene una lista de elementos de tipo `enlace` (un ejemplo se muestra en la figura 5.16).

Algoritmo 4 Creación de enlaces variables.

```
Entradas: SUSTITUCIONES: empty links-list; I:links-collection; D:document
  for all links of type "substitution" with origin in D do
    recover origin node and create an element origen with it;
    recover target node and create an element destino with it;
    place both elements in an element enlace;
    add this enlace to SUSTITUCIONES;
  end for
```

Paso 2: resolución de los enlaces

El algoritmo 2 del capítulo 3 aplica las sustituciones a la versión inicial del documento abstracto. Los enlaces utilizados se encuentran en la variable SUSTITUCIONES creada en el paso 1.

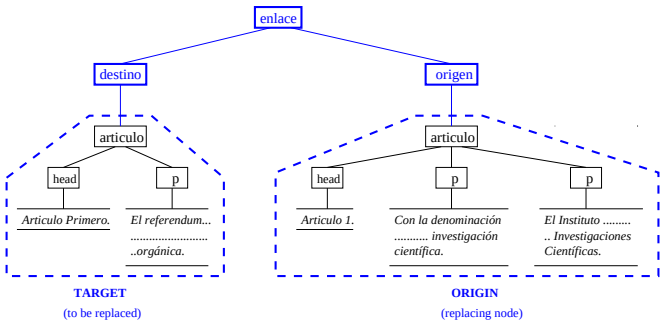


Figura 5.16: Elementos de una variable de enlaces. La variable contiene un único elemento `enlace`. Este elemento está compuesto a su vez por dos elementos: `origen` - que contiene el subárbol origen del enlace de substitución-, y `destino`, que contiene el nodo destino del enlace.

5.6 Discusión

El sistema

En este prototipo se han implementado los servicios relacionados con la explotación de relaciones. En lo que se refiere a servicios clásicos de bibliotecas digitales, también se han incluido aquellos que son necesarios para demostrar los objetivos de esta tesis: recuperación de documentos y búsqueda en documentos. Otras funcionalidades, como la navegación y la búsqueda de documentos, podrían mejorar la biblioteca.

Los componentes implementados expresamente para esta tesis son el servidor de documentos, la interfaz de usuario y el traductor de documentos. Todos ellos se han implementado como objetos, y la interfaz de usuario es accesible vía una interfaz Web, controlada por un *servlet*. El servidor de documentos asume los métodos que permiten la generación de versiones y la recuperación de documentos.

Las cualidades destacables del prototipo son:

- *Portabilidad.*

  La utilización de la tecnología Java garantiza la independencia de plataforma.

- *Reutilización.*

  En este prototipo se han reutilizado componentes procedentes de implementaciones existentes. Este es el caso de la herramienta de búsqueda en documentos [54]. A su vez, los componentes de esta biblioteca pueden ser reutilizados en otros sistemas, de modo aislado, en función del servicio que se pretenda incorporar.

La biblioteca del prototipo está centralizada en un servidor, lo cual ha simplificado aspectos como la manipulación y resolución de los identificadores de documentos. Sin embargo, en un caso de multiplicidad o distribución de los repositorios, el tratamiento sería el mismo (mismos componentes, multiplicados); incluso los principios de trabajo con identificadores se mantendrían, si bien la resolución sería necesariamente más compleja.

Traducción de documentos

La traducción de documentos se ha probado con un conjunto de documentos legislativos obtenidos de distintos servidores públicos. Se trata de páginas HTML etiquetadas según criterios variables en función del proveedor[4]. Antes de utilizarlos se pasó por una etapa de "limpieza" para adaptarlos a los requisitos de entrada del algoritmo de traducción. El requisito de que fuesen documentos XML bien formados fue añadido al algoritmo, ya que la aplicación que lo implementa trabaja sobre un analizador XML. Este pasa elementos de tipo "elemento", "texto", etc. a la aplicación, según las especificaciones de la API SAX 1.0[5]. Tanto la equivalencia entre ontologías como las reglas de inclusión se pasan a la aplicación en los parámetros de entrada del método de traducción.

Petición de enlaces y generación de versiones de documentos

La petición de enlaces y la generación de versiones también han sido implementadas usando un parser XML. Disponer tanto de los documentos como de los enlaces en formato XML proporciona directamente dos lenguajes de direccionamiento de fragmentos internos (XPointer) y de manipulación de documentos (XSLT). Además, el modelo arborescente asociado a cualquier documento XML es totalmente adecuado para los recorridos recursivos utilizados en la generación de versiones. Este tratamiento se ha probado sobre un conjunto de textos normativos, tanto en los casos de modificaciones coincidentes como los de solapamientos parciales en las modificaciones. Sin embargo, las modificaciones transitivas requieren para su implementación mayor flexibilidad que la que ahora mismo ofrecer las herramientas XML. No obstante, la rápida estabilización de XML y sus estándares asociados permite pensar que en breve se dispondrá de herramientas suficientemente flexibles para llevar a buen término la implementación de esta posibilidad.

La Detección de referencias

Las referencias y las modificaciones en los documentos legislativos se pueden detectar por la presencia de citas. La rigidez lingüística y sintáctica característica de estos textos (la figura 5.17 muestra algunos ejemplos extraídos de textos normativos españoles) facilita la detección automática de referencias [121] frente a otros contextos más "flexibles" en sus estructuras lingüísticas. Esto ha permitido implementar la detección de referencias mediante un reconocimiento de claves dentro del texto. Se quiere resaltar que el problema de la detección de referencias no es objetivo primordial de esta tesis. Sin embargo, sí lo es incorporar un servicio semejante a una biblioteca donde se exploten los enlaces, objetivo que se considera queda probado en el prototipo.

[4]Se trata de los servidores de distintos ministerios españoles (http://www.igsap.map.es/cia/dispo/-Ibe.htm, http://www.map.es/gobierno/legisla) y el servidor público "La Ley" (http://www.laley.net).

[5]Disponible en http://www.meggison.com/SAX.

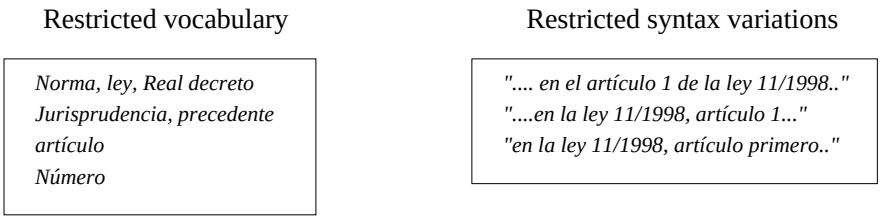


Figura 5.17: Variaciones en los modos de referenciar documentos en los textos normativos españoles. La sintaxis y el vocabulario son rígidos. Por ejemplo, las variaciones en las formas de referenciar un *artículo* son mínimas, como se puede ver en la lista de la derecha.

6

Conclusiones

Índice General

El campo de aplicación de esta tesis son las bibliotecas digitales; los documentos estructurados, las relaciones entre ellos y la generación de nuevos documentos son los aspectos que han centrado este estudio. Del objetivo de explotar las relaciones más allá de la obtención de hipertexto navegacional surgen otros objetivos que es necesario conseguir para realizar el primero. Establecer la diferencia clara entre el documento abstracto y sus copias digitales permite definir con precisión el primero de ellos: obtener una copia digital cuya estructura lógica sea imagen fiel de la estructura del documento abstracto. De este modo se podrá también modelar y explotar convenientemente las relaciones de referencia y modificación entre los documentos, las cuales se expresan habitualmente en base a la estructura lógico-semántica del documento abstracto. Las referencias se expresan citando la porción de documento afectada por la relación en base a su posición relativa en el árbol estructural; del mismo modo, las modificaciones, que suelen encontrarse próximas en el texto a alguna referencia que involucra al mismo fragmento que se modifica, afectan a fragmentos localizables por su posición relativa en la jerarquía estructural. Ésta es la jerarquía del documento abstracto.

Una vez se dispone de esta copia, se puede avanzar en el siguiente objetivo: consultar las relaciones, y generar automáticamente versiones históricas de los documentos. La consulta de las relaciones es posible si éstas se expresan como enlaces, los cuales pueden ser objeto de consultas al igual que cualquier otro dato almacenado en una base de datos. La base de datos que alberga los enlaces es en realidad la implementación de un grafo de relaciones, con la información sobre tres clases de relaciones: estructurales, referencias y modificaciones. La generación de versiones es posible mediante un recorrido del grafo de relaciones, cuyos nodos son fragmentos de documentos, localizables por su posición relativa en el árbol del documento.

6.1 Aportaciones y conclusiones

Los enlaces se han considerado en esta tesis como objetos de primer nivel, sacándolos del lugar secundario a que han sido relegados hasta ahora. Es necesario extraerlos de los documentos y dedicarles un estudio propio, que permita extraer de ellos la valiosa información que albergan acercan de los documentos que relacionan. Así, la funcionalidad de una biblioteca digital se puede enriquecer con la introducción de servicios de traducción de documentos, y servicios que hacen uso de los enlaces: detección de referencias y generación de versiones. Los enlaces utilizados en esta tesis no sirven para expresar las reglas de composición que ligan un documento patrón con sus integrantes, sino que a partir de ellos se han obtenido las directrices de composición implícitas en el grafo relacional, mediante un recorrido del grafo de enlaces, aprovechando la información semántica implícita en ellos, yendo así más allá de la composición de documentos por reglas explícitas. Se ha aportado, pues, un modelo de expresión de las relaciones, que expande las posibilidades potenciales de las propuestas basadas en hipertexto y utilización de metadatos, así como soluciones para la consulta avanzada y explotación dinámica (composición de documentos) de las relaciones entre documentos.

Las relaciones y generación de versiones exigen tener en cuenta la visión mixta del grafo de relaciones donde cada nodo es a su vez la raíz de un conjunto de nodos que se

puede caracterizar localizando el primero. El grafo final resultante contiene las relaciones estructurales entre elementos de los documentos, las referencias y las modificaciones. A partir de él se pueden obtener grafos parciales y subgrafos cuyos recorridos facilitan la obtención de información sobre las relaciones, o la generación de versiones cuyo eje de trabajo es un recorrido recursivo.

Las versiones de documentos han recibido una atención relevante. No es necesario expresar de modo explícito las reglas de composición de un documento, si se dispone de la información semántica que está en el origen de su existencia. Esta tesis aporta un método para deducir estas reglas al tiempo que se construye el documento por composición. Se pueden considerar tres aspectos en lo que a las versiones concierne: la detección de cambios entre versiones, la representación de estos cambios y las consultas sobre ellos. El modo que se ha escogido para detectarlas es la extracción a partir del contenido de los documentos, que es a fin de cuentas el origen de dichos cambios y evita comparar versiones. Los cambios (modificaciones) se han modelado como enlaces, lo cual reduce el problema de consultar los cambios a consultas sobre los enlaces. Por otro lado, no se asume la existencia de identificadores en los elementos afectados por una relación (lo cual implicaría una posible arbitrariedad en la generación de estos identificadores) si no que se trabaja en todo momento en base a la posición relativa de los elementos en el árbol documental.

Se ha tenido en cuenta en la proposición para explotar las modificaciones que éstas forman parte del contenido de algún documento, con entidad como tal, que no debe descomponerse en fragmentos aislados para facilitar la manipulación de enlaces. Se mantiene así la integridad de los documentos de partida, que no se ven afectados por ningún proceso de creación de enlaces, sin replicar por ello información o introducir nuevos elementos en la base de datos que no tienen entidad suficiente para ser considerados documentos.

Se ha introducido una nueva visión de los documentos; cada documento se modela como la agregación de tres elementos de información: metadatos que lo describen, contenido y los enlaces que le afectan (los enlaces no habían sido considerados hasta ahora elementos de la misma relevancia que contenido o metadatos). Considerarlo de este modo aporta varias ventajas. En primer lugar, el contenido se puede manipular sin afectar por ello a los enlaces ni metadatos asociados al documento. En segundo lugar, tanto los enlaces como los metadatos se pueden consultar independientemente del contenido. Además los enlaces son accesibles desde cualquiera de los elementos implicados en la relación: se puede atravesar el grafo de relaciones multidireccionalmente.

El algoritmo propuesto en el capítulo 2 tiene su mayor ventaja en la obtención de una copia semánticamente estructurada. No solo el hecho de preservar la semántica del documento es importante, sino que esto permite realizar correctamente la correlación entre las copias disponibles en la base de documentos y los enlaces almacenados en la base de enlaces. El documento es recorrido de modo secuencial, simulando el modo en que una persona reconoce la estructura durante una lectura. La limitación del algoritmo se encuentra en la imposibilidad de obtener un algoritmo general, que trabaje sobre cualquier etiquetado o estructura en la entrada. El desconocimiento sobre dicho etiquetado y la jerarquía de contención obliga a imponer exigencias sobre la entrada que restringen la generalidad del algoritmo. Las condiciones de los documentos que se utilizaron en el prototipo decidieron las restricciones a aplicar: el anidamiento entre

elementos semánticos se prohibe, para evitar confusiones entre los comienzos de elementos y referencias, a la vista de que los documentos de prueba nunca presentan un anidamiento de estas características.

Por último, está claro que las soluciones propuestas se benefician en las bibliotecas digitales de la organización de estos sistemas, pero que son aplicables a otros entornos masivos de información que comparten con ellas algunos de sus problemas, como es el caso de la omnipresente Internet.

6.2 El prototipo y la tecnología

El prototipo de documentos legislativos utilizados reune las características de estructuración, multiplicidad de relaciones y frecuentes modificaciones adecuados para esta tesis.

La traducción de documentos se ha probado con éxito sobre un conjunto de textos normativos, que son los documentos base sobre los cuales se implementa la obtención de versiones. Tanto la traducción como cualquiera de las explotaciones de enlaces se implementa utilizando la tecnología XML, que permite acceder a los documentos, sus fragmentos y manipularlos. Por otro lado, la visión arborescente asociada a cualquier documento XML se adapta perfectamente al algoritmo recursivo de creación de versiones.

Por último, es importante remarcar que la existencia de un estándar como XML, que permite modelar semánticamente los documentos estructurados, y de sus estándares asociados ha sido determinante en el prototipo. En caso de no disponer de lenguajes que permiten acceder a los fragmentos internos de los documentos (XPath), modelar los enlaces y su semántica (XLink), e incluir las referencias a porciones de documentos en dichos enlaces (XPointer), la implementación de las propuestas de esta tesis pasaría obligatoriamente por la definición de herramientas equivalentes. Éstas, además de la evidente dificultad adicional de su definición, tendrían el inconveniente de no ser estándares aceptados, lo cual afectaría negativamente a la interoperabilidad en el sistema.

6.3 Trabajo futuro

Un tratamiento automático de las relaciones estaría completo con una detección automática general de las relaciones a partir de las referencias en los textos de los documentos. Las experiencias en generación de identificadores en las bibliotecas de artículos técnicos muestran que este objetivo está cada día más cercano. Se podría así ampliar esta detección a marcos de trabajo diferentes del entorno legislativo del prototipo.

Los servicios propuestos en el capítulo 4 pueden ampliarse con otros servicios que enriquezcan la obtención de información a partir de los enlaces (por ejemplo, la obtención de datos estadísticos) o que faciliten la intervención del usuario. El interés de los usuarios por obtener información "relacionada"es creciente, tanto en las bibliotecas digitales como en Internet, lo cual convierte a estos servicios relacionados con los enlaces

en algo necesario además de deseable.

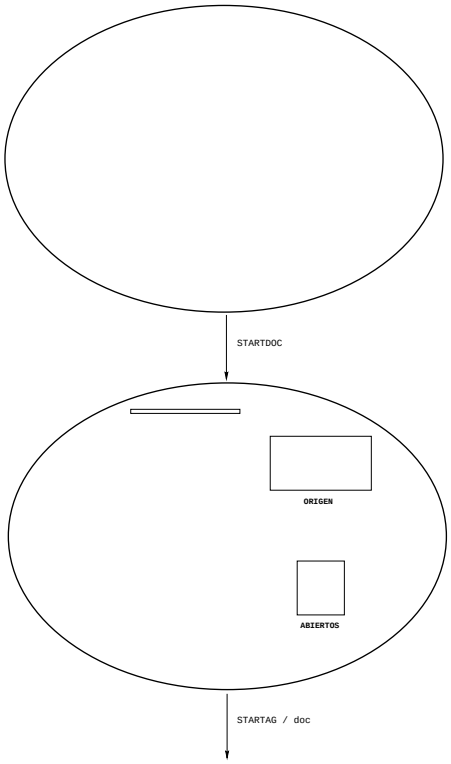
Como se ha dicho, el algoritmo de traducción de documentos se ve limitado en su generalidad por la posibilidad de confusión entre los comienzo de elementos semánticos y las referencias a otros documentos. Este algoritmo podría mejorarse, por tanto, con la inclusión de algún método que permita establecer dicha distinción mediante un análisis del contenido.

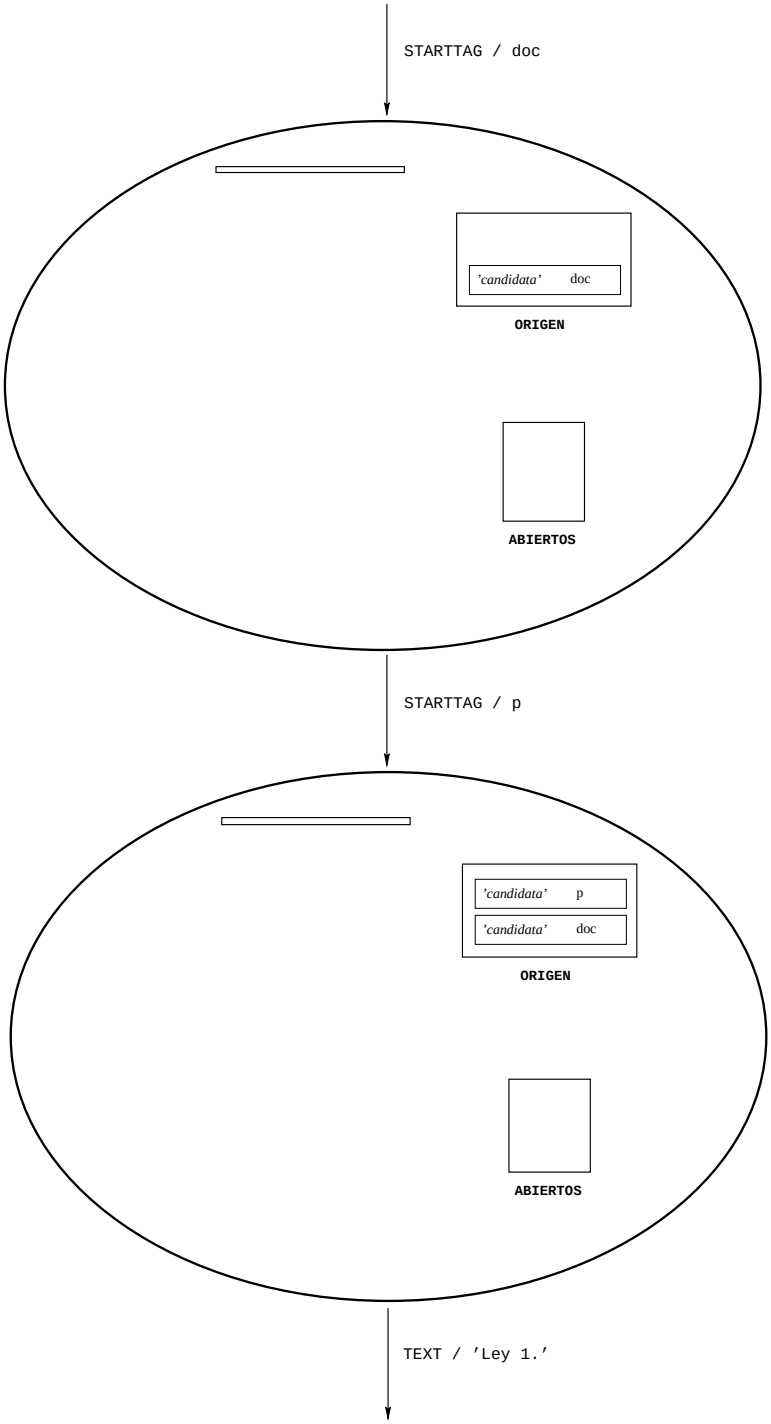
También las limitaciones de la generación de versiones en los casos de solapamientos conflictivos se verían eliminadas si fuese posible distinguir las modificaciones incoherentes de aquellas que no lo son. Para ello, la forma idónea es introducir al usuario como elemento activo en la generación, que guíe a la aplicación ayudándola a discernir entre conflictos resolubles y aquellos que debe ignorar.

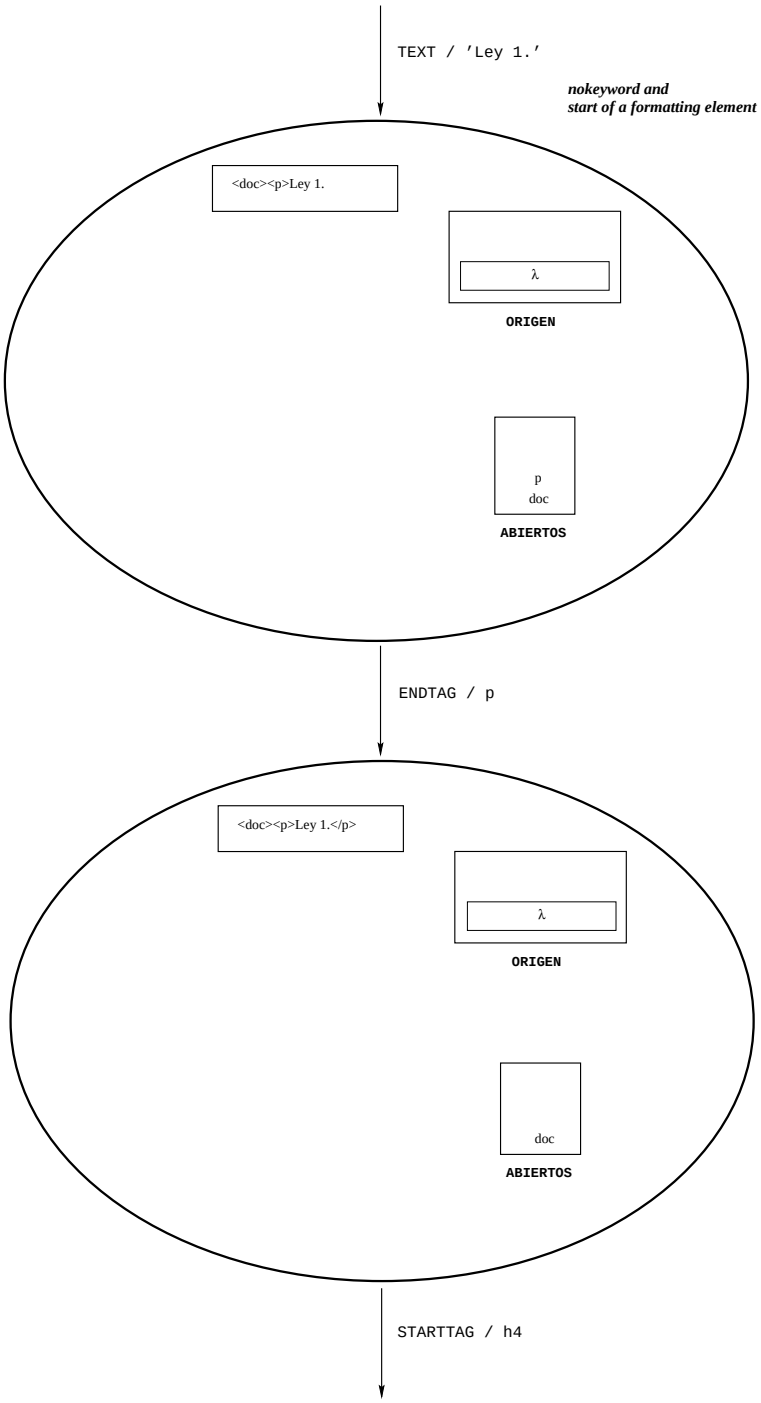
Por último, las propuestas de explotación del grafo de relaciones pasan para su expansión futura por una etapa de profundización en el estudio de los aspectos formales de éste, que permita caracterizar propiedades tales que conduzcan a un modelo formal bajo el cual se puedan expresar, tanto los tratamientoss mencionados, como otros que pudieran proponerse en el futuro.

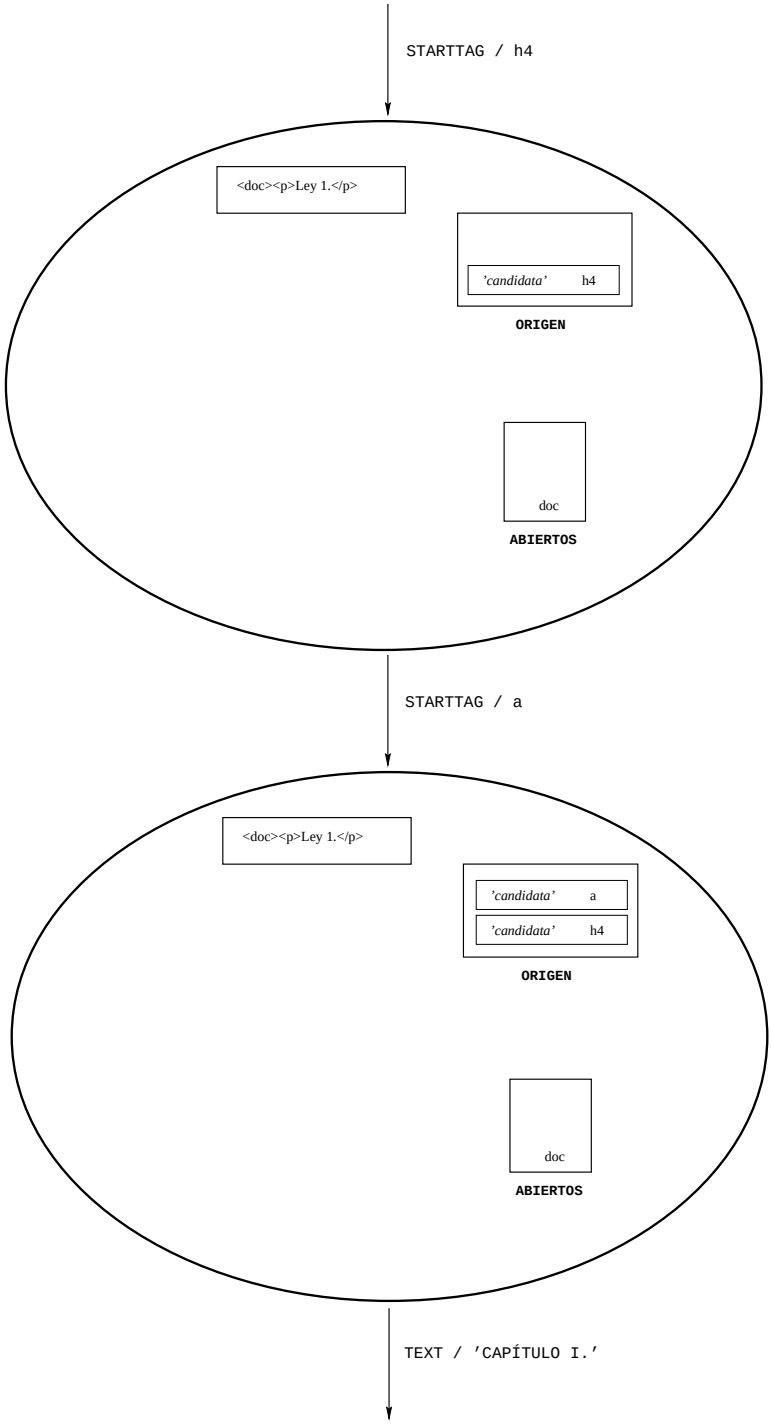
A

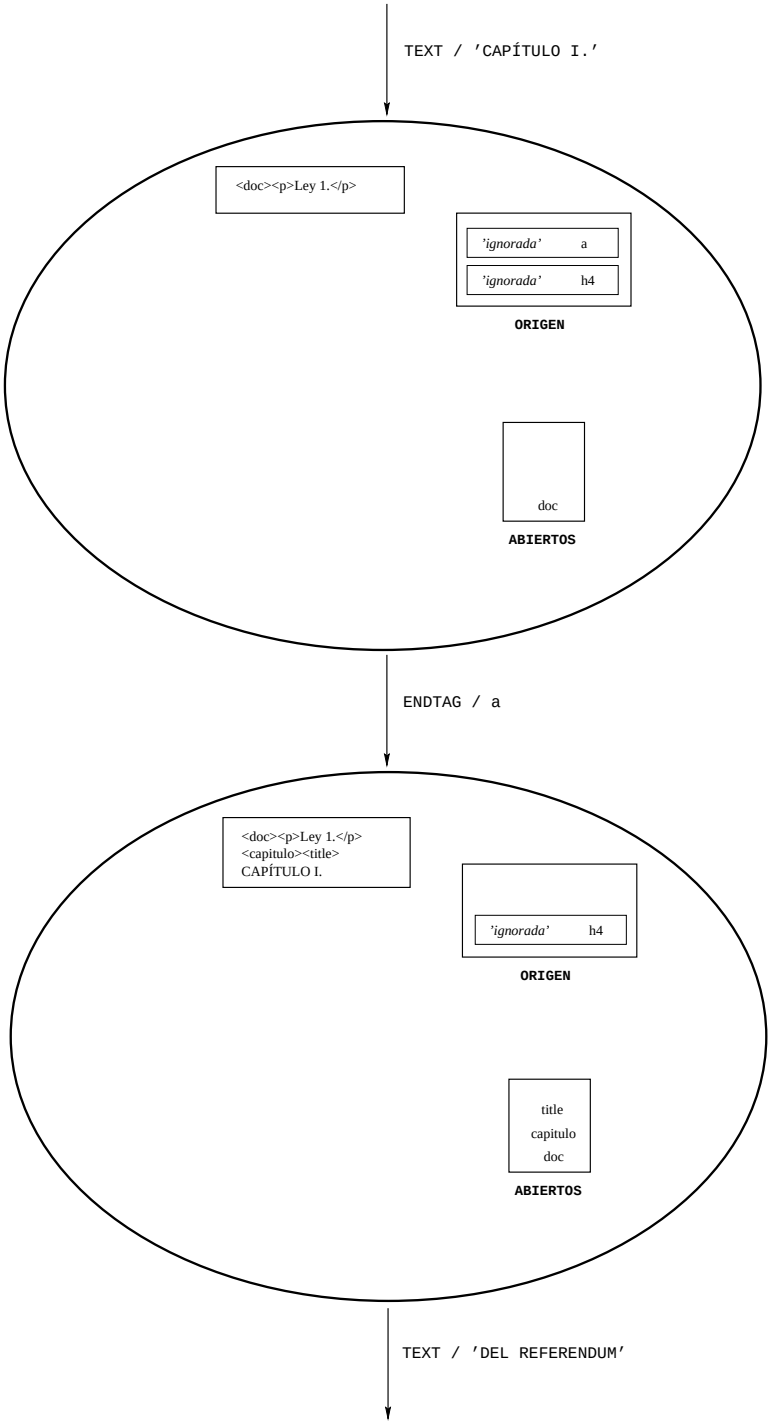
Ejemplo de aplicación del algoritmo de traducción de documentos

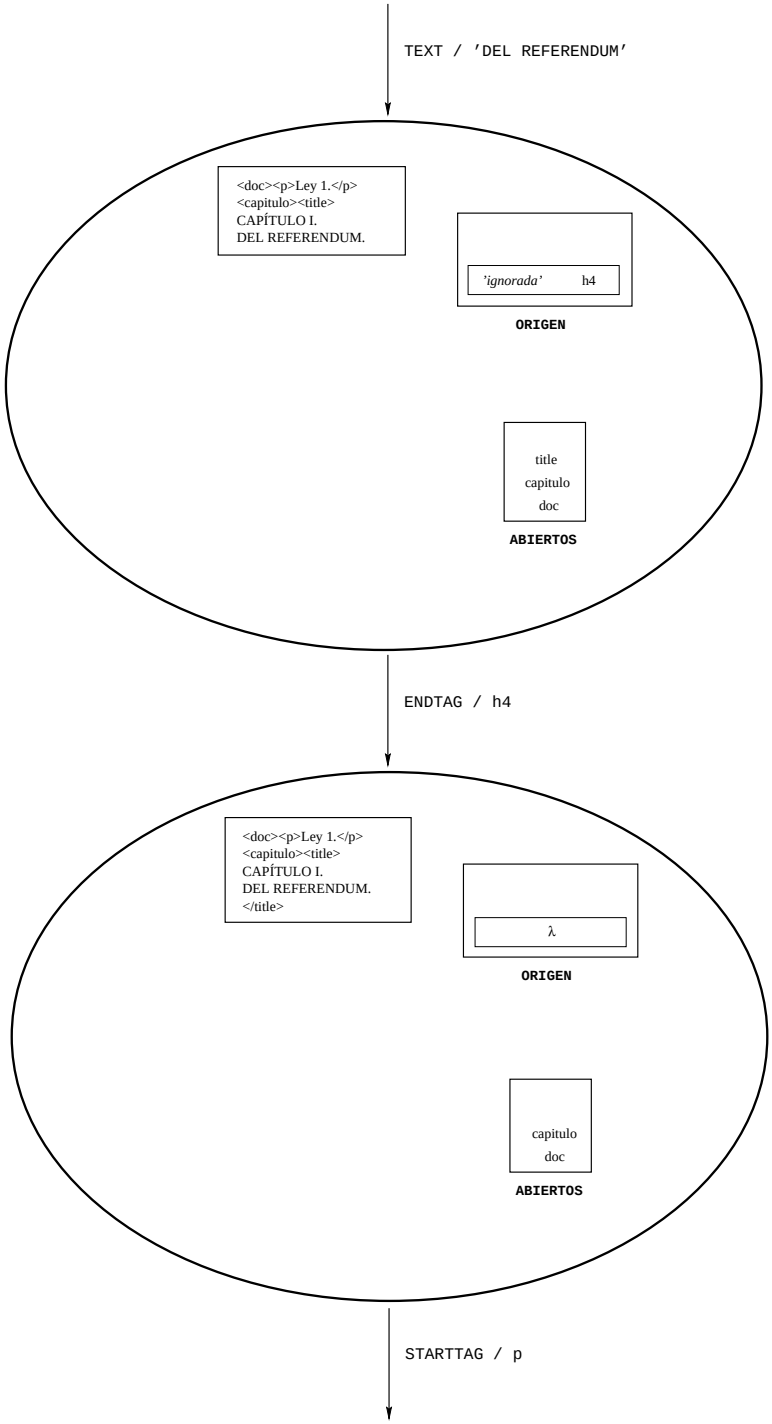


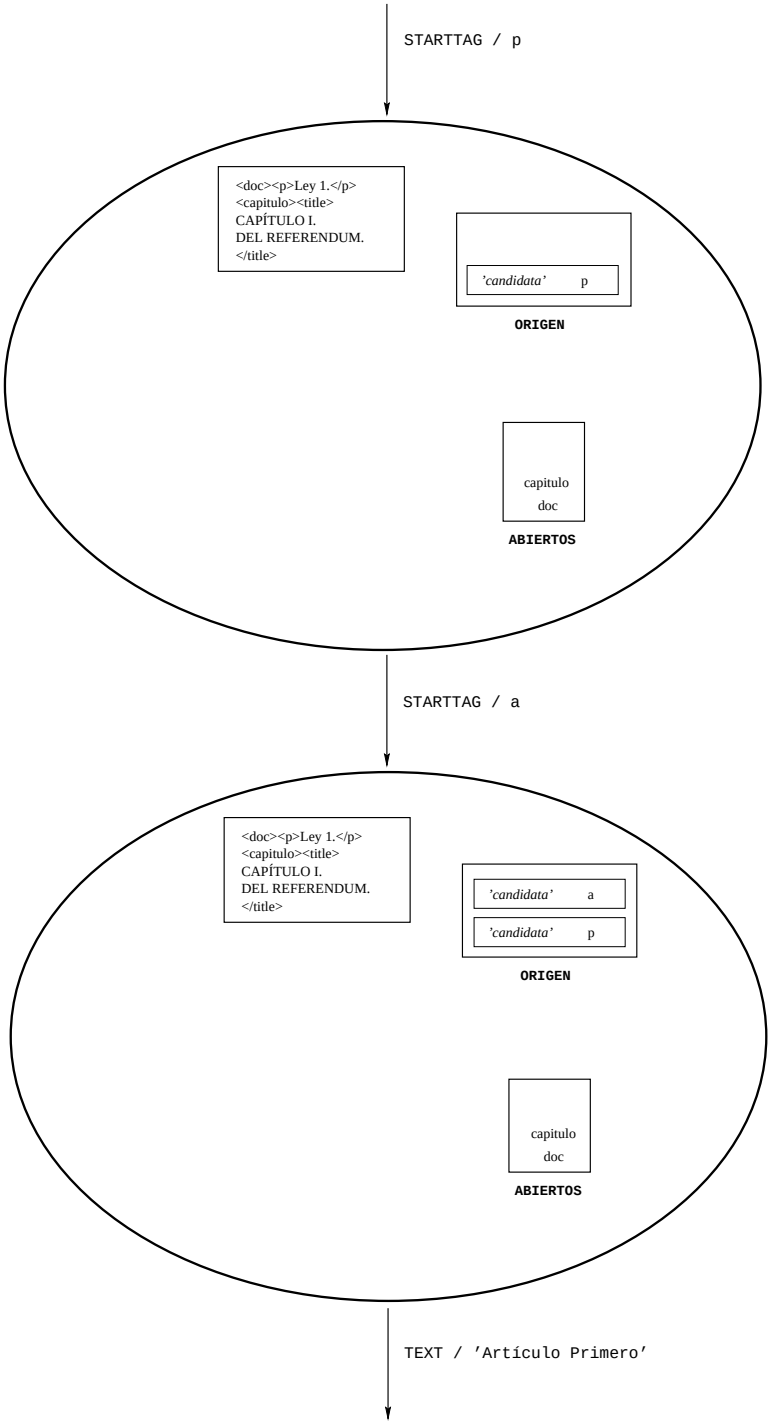

STARTTAG / doc
'candidata' doc
ORIGEN
ABIERTOS
STARTTAG / p
'candidata' p
'candidata' doc
ORIGEN
ABIERTOS
TEXT / 'Ley 1.'

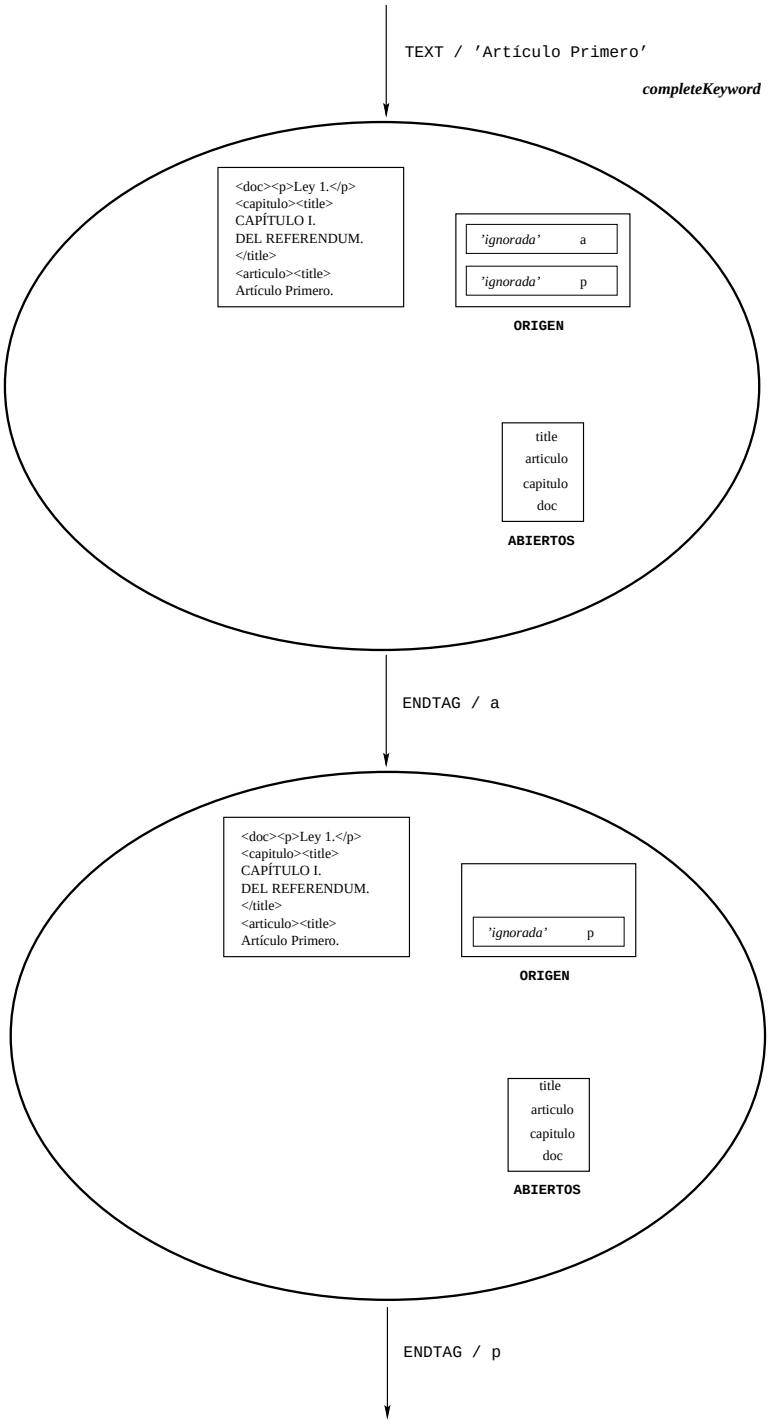

TEXT / 'Ley 1.'
nokeyword and
start of a formatting element
<doc> Ley 1.
λ
ORIGEN
p
doc
ABIERTOS
ENDTAG / p
<doc> Ley 1.
λ
ORIGEN
doc
ABIERTOS
STARTTAG / h4

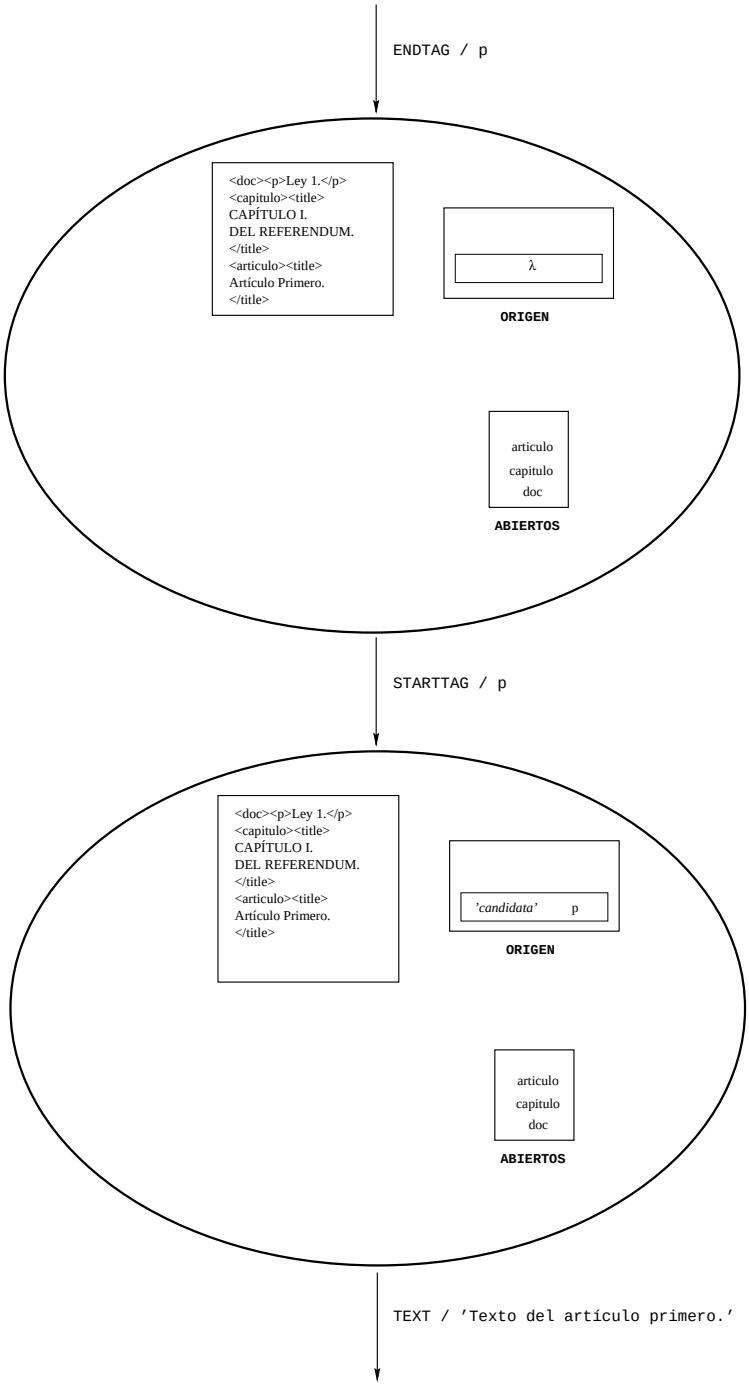

STARTTAG / h4
<doc> Ley 1.
'candidata' h4
ORIGEN
doc
ABIERTOS
STARTTAG / a
<doc> Ley 1.
'candidata' a
'candidata' h4
ORIGEN
doc
ABIERTOS
TEXT / 'CAPÍTULO I.'

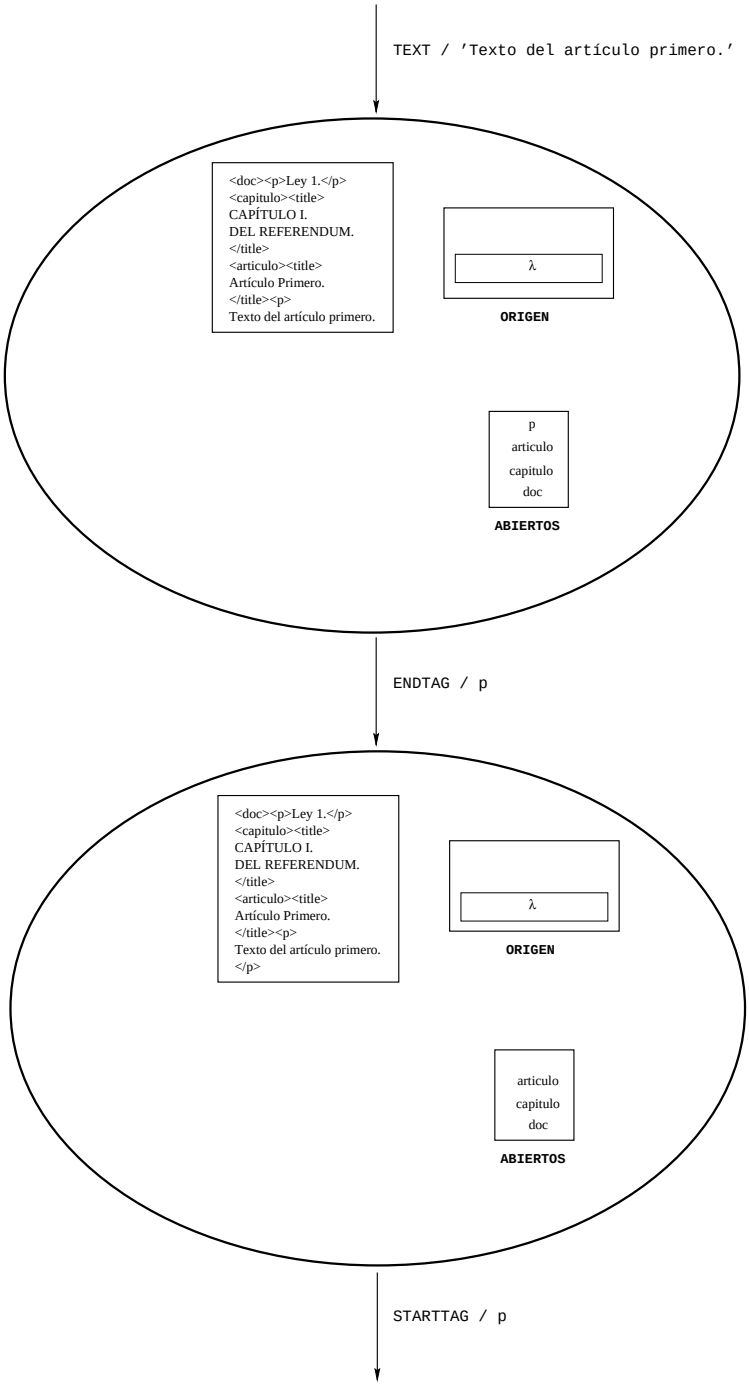

TEXT / 'CAPÍTULO I.'
<doc> Ley 1.
'ignorada' a
'ignorada' h4
ORIGEN
doc
ABIERTOS
ENDTAG / a
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
'ignorada' h4
ORIGEN
title
capitulo
doc
ABIERTOS
TEXT / 'DEL REFERENDUM'


TEXT / 'DEL REFERENDUM'
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
'ignorada'
h4
ORIGEN
title
capitulo
doc
ABIERTOS
ENDTAG / h4
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
λ
ORIGEN
capitulo
doc
ABIERTOS
STARTTAG / p

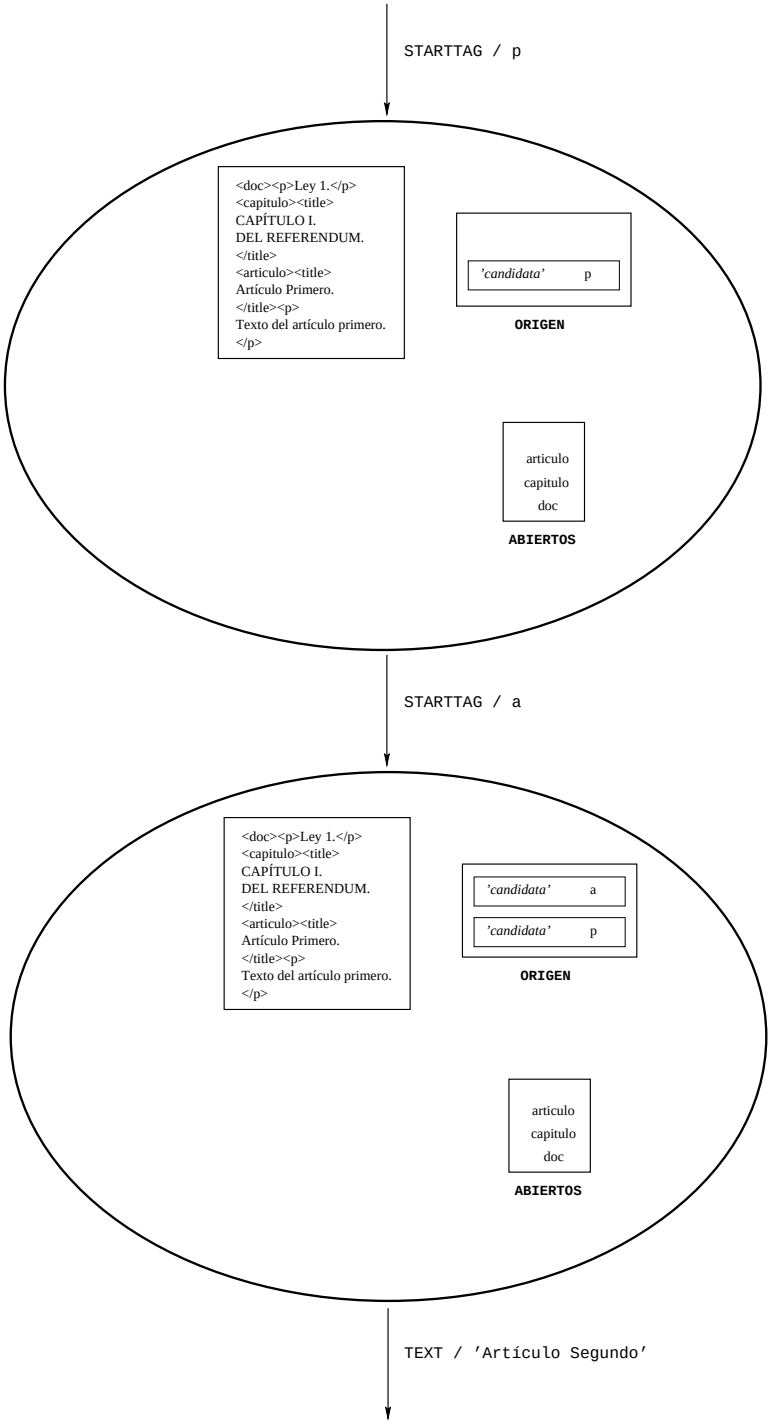

STARTTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
'candidata' p
ORIGEN
capitulo
doc
ABIERTOS
STARTTAG / a
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
'candidata' a
'candidata' p
ORIGEN
capitulo
doc
ABIERTOS
TEXT / 'Artículo Primero'


TEXT / 'Artículo Primero'
completeKeyword
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
'ignorada' a
'ignorada' p
ORIGEN
title
articulo
capitulo
doc
ABIERTOS
ENDTAG / a
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
'ignorada' p
ORIGEN
title
articulo
capitulo
doc
ABIERTOS
ENDTAG / p


ENDTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
λ
ORIGEN
articulo
capitulo
doc
ABIERTOS
STARTTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
'candidata' p
ORIGEN
articulo
capitulo
doc
ABIERTOS
TEXT / 'Texto del artículo primero.'

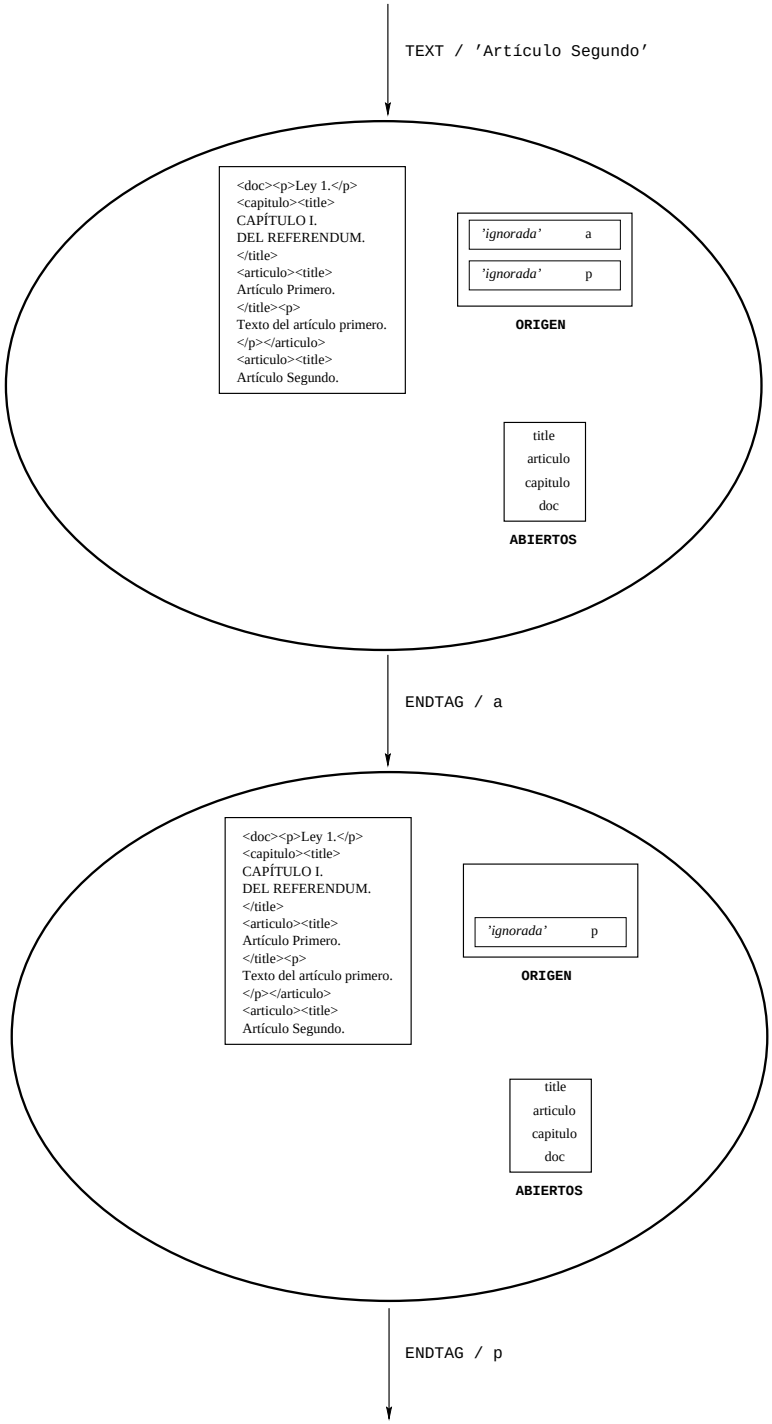

TEXT / 'Texto del artículo primero.'
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
λ
ORIGEN
p
articulo
capitulo
doc
ABIERTOS
ENDTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.

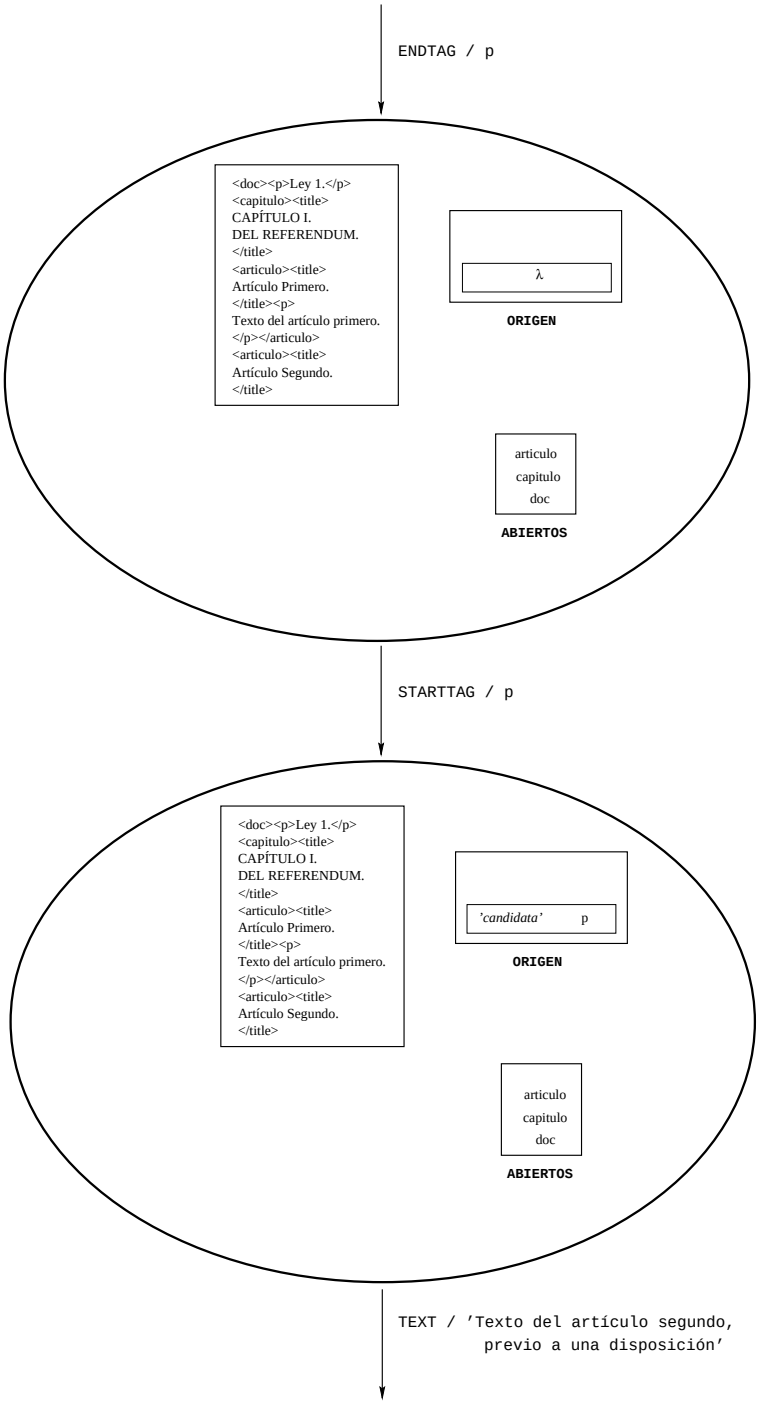
λ
ORIGEN
articulo
capitulo
doc
ABIERTOS
STARTTAG / p

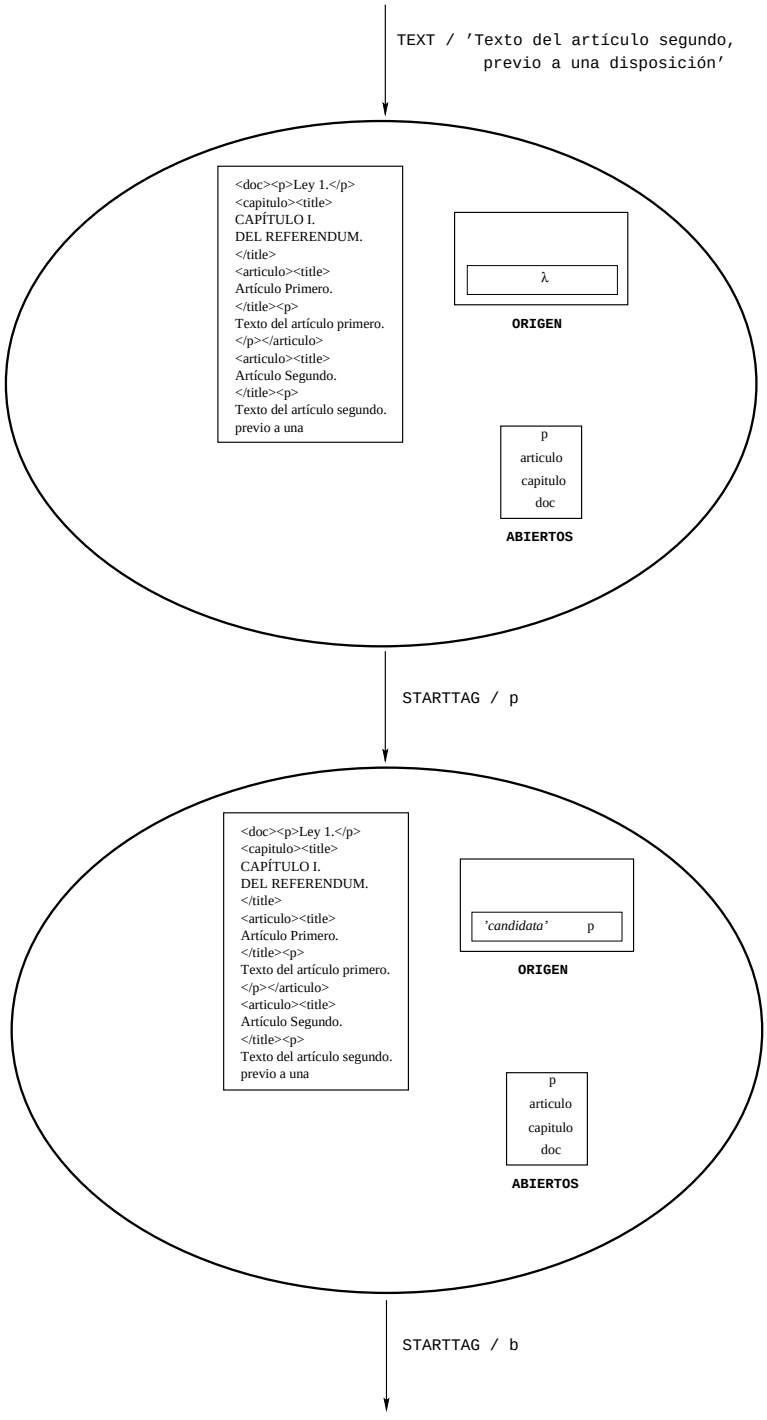

STARTTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.

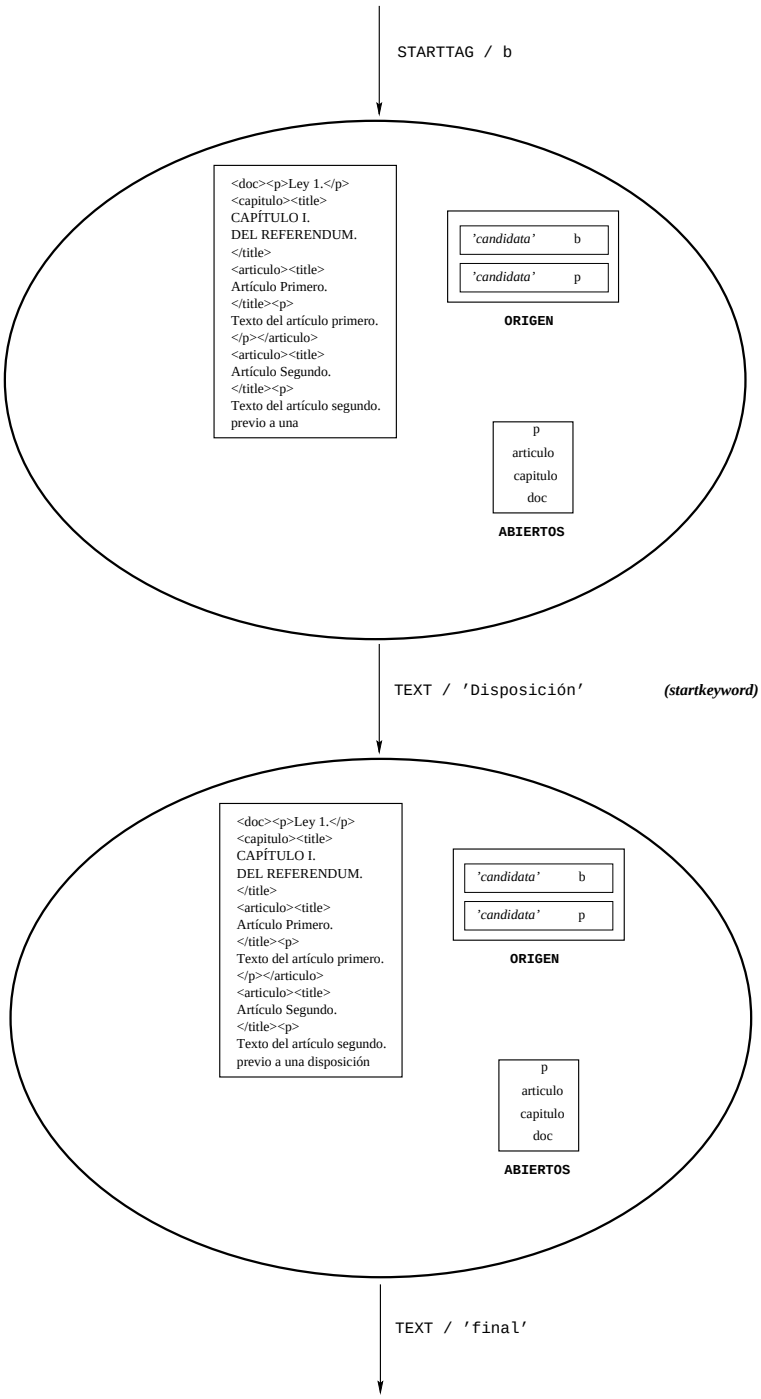
'candidata' p
ORIGEN
articulo
capitulo
doc
ABIERTOS
STARTTAG / a
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.

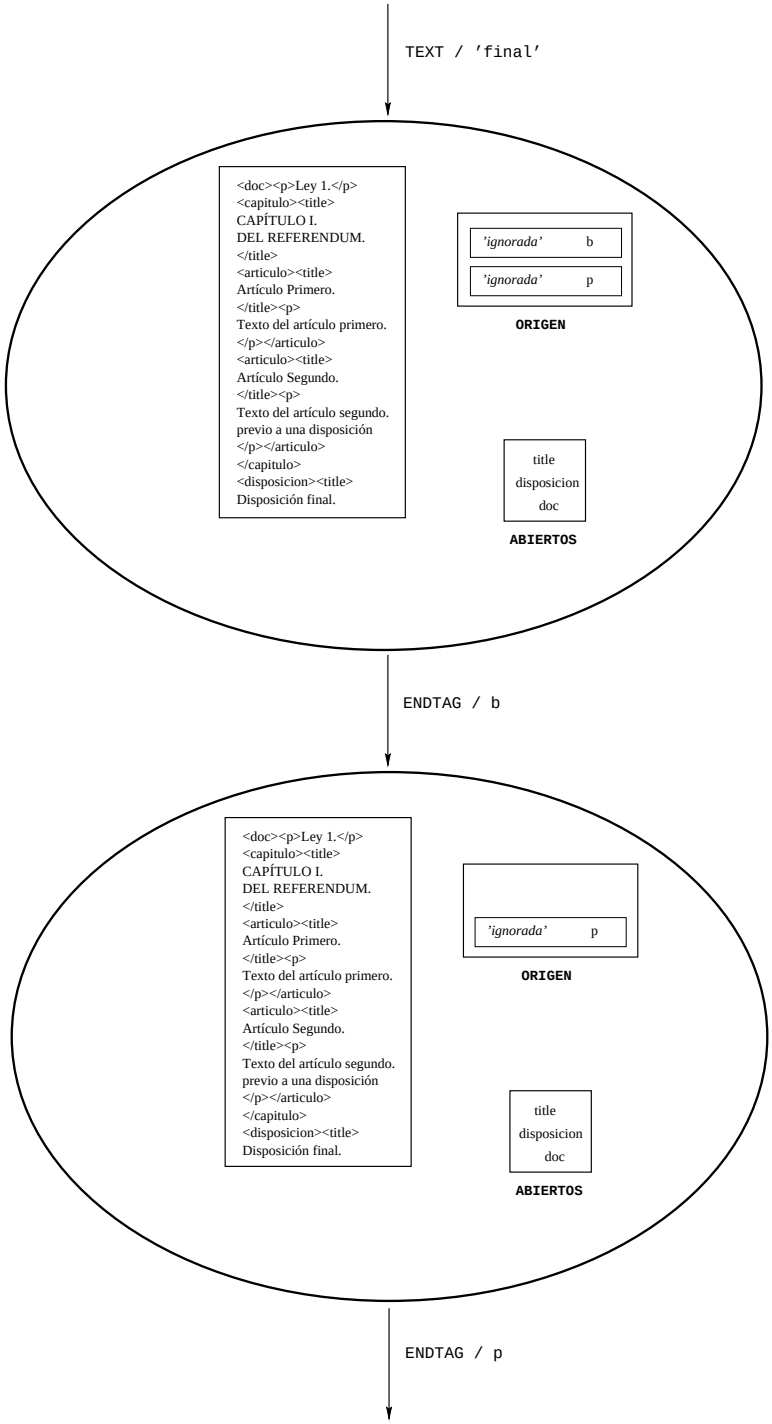
'candidata' a
'candidata' p
ORIGEN
articulo
capitulo
doc
ABIERTOS
TEXT / 'Artículo Segundo'

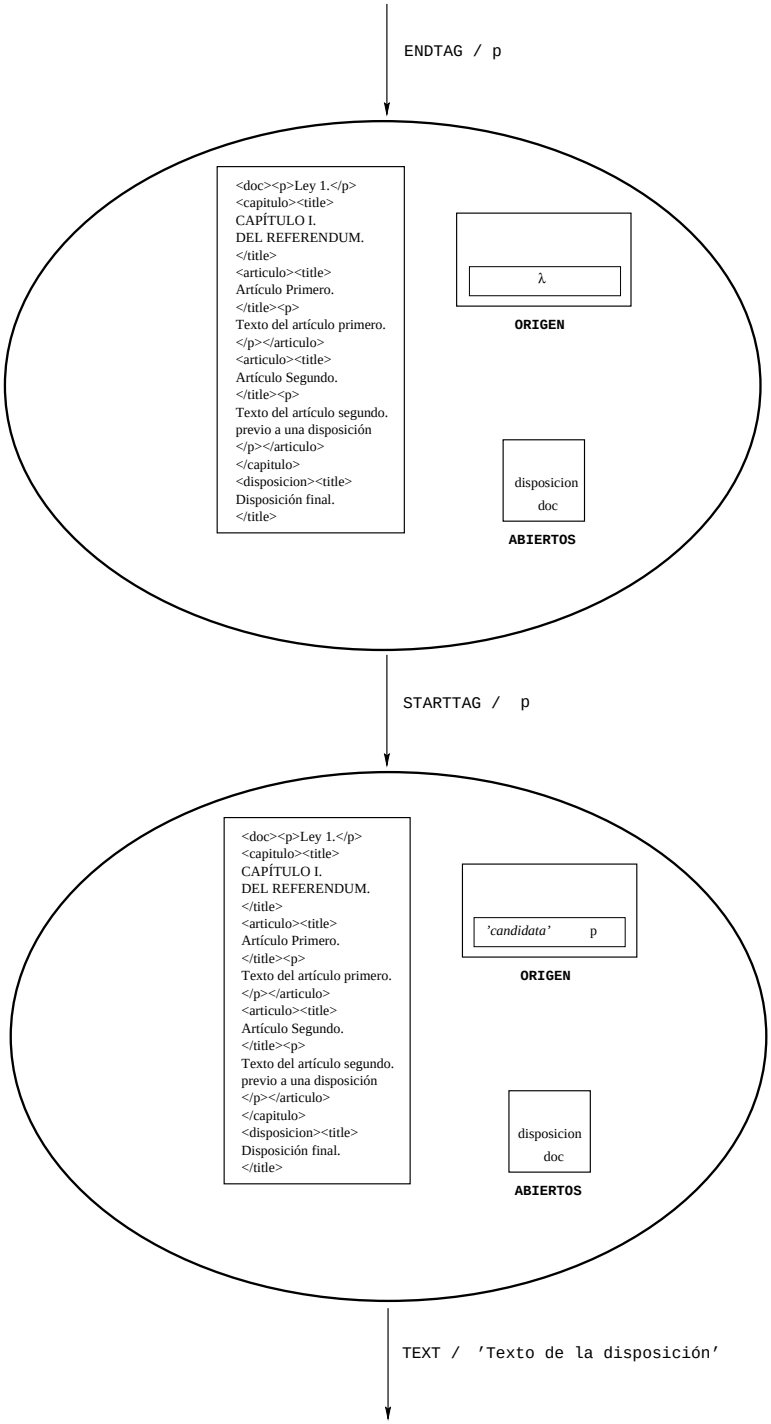

TEXT / 'Artículo Segundo'
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
'ignorada' a
'ignorada' p
ORIGEN
title
articulo
capitulo
doc
ABIERTOS
ENDTAG / a
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
'ignorada' p
ORIGEN
title
articulo
capitulo
doc
ABIERTOS
ENDTAG / p

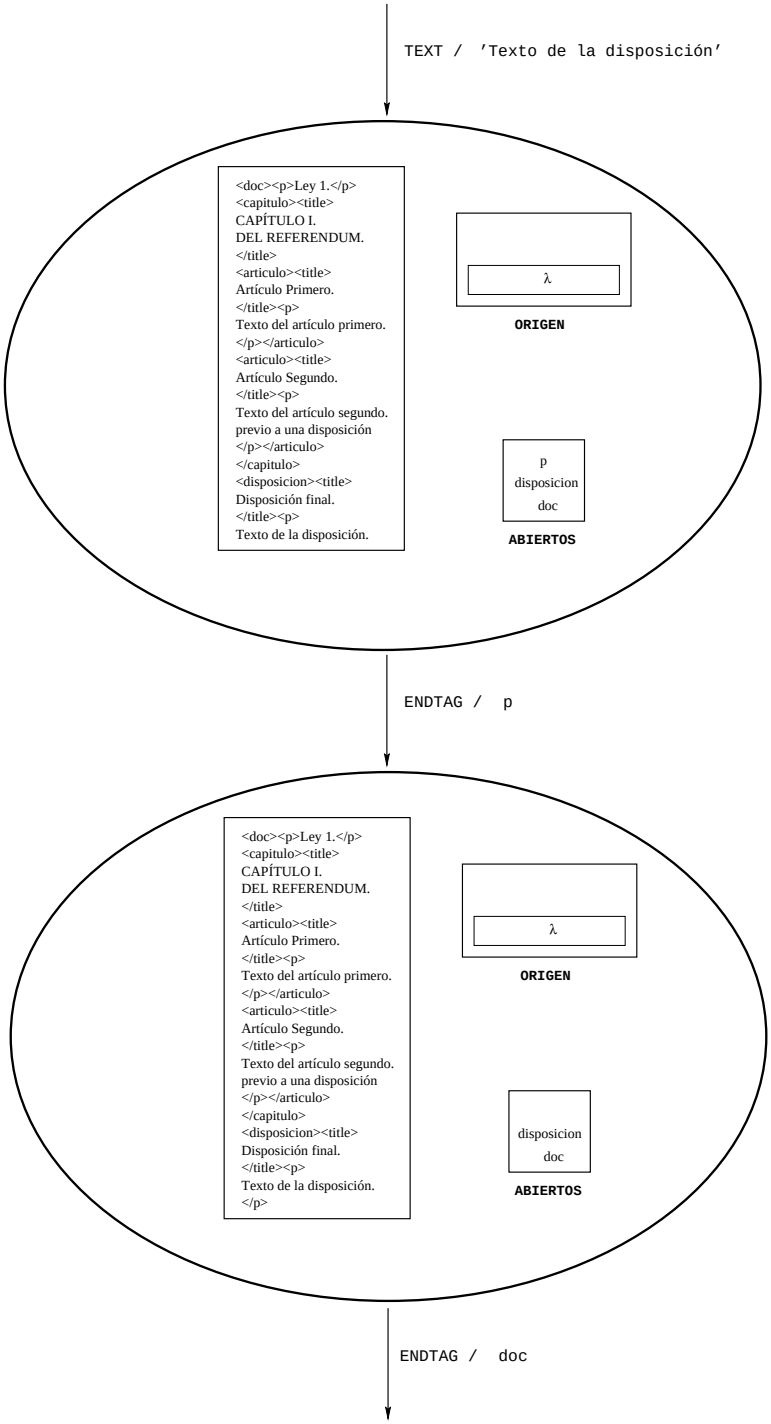

ENDTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
λ
ORIGEN
articulo
capitulo
doc
ABIERTOS
STARTTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
'candidata' p
ORIGEN
articulo
capitulo
doc
ABIERTOS
TEXT / 'Texto del artículo segundo,
previo a una disposición'


TEXT / 'Texto del artículo segundo,
previo a una disposición'
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una
λ
ORIGEN
p
articulo
capitulo
doc
ABIERTOS
STARTTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una
'candidata' p
ORIGEN
p
articulo
capitulo
doc
ABIERTOS
STARTTAG / b

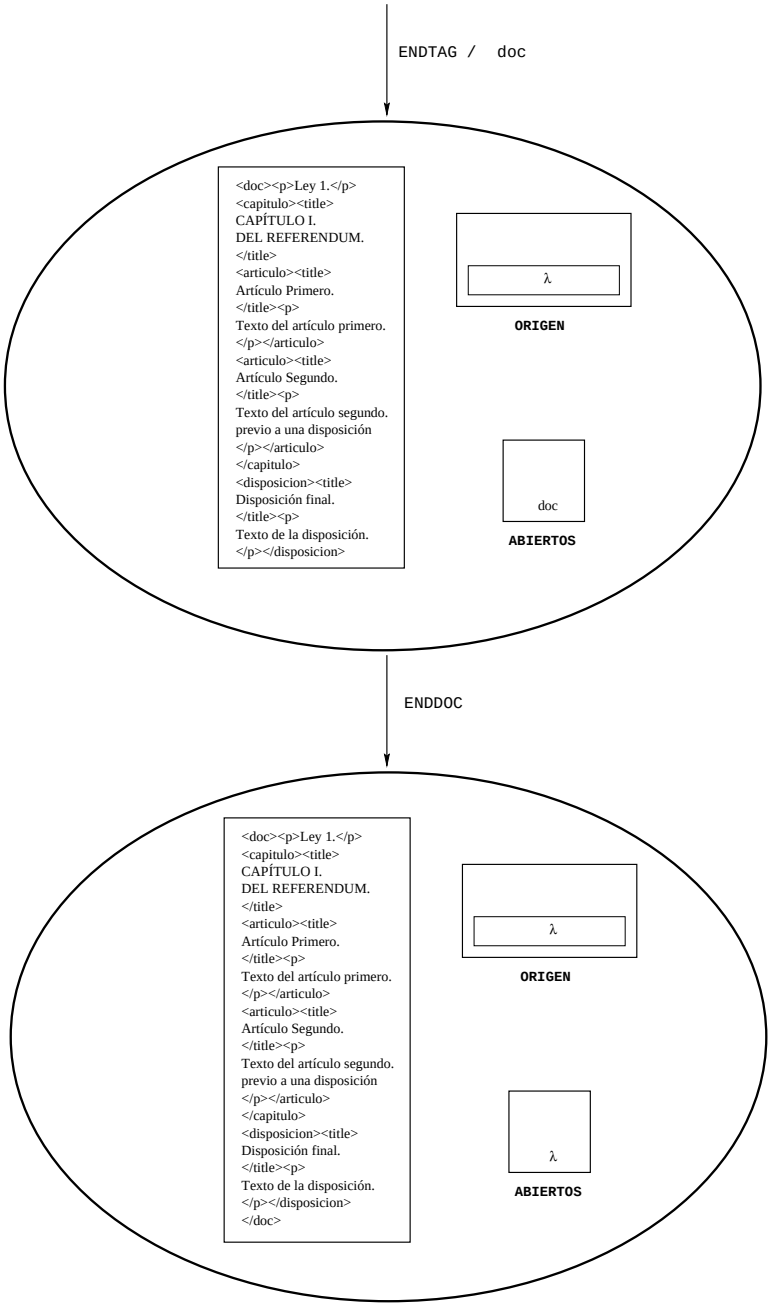

STARTTAG / b
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una
'candidata' b
'candidata' p
ORIGEN
p
articulo
capitulo
doc
ABIERTOS
TEXT / 'Disposición'
(startkeyword)
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
'candidata' b
'candidata' p
ORIGEN
p
articulo
capitulo
doc
ABIERTOS
TEXT / 'final'


TEXT / 'final'
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
'ignorada' b
'ignorada' p
ORIGEN
title
disposicion
doc
ABIERTOS
ENDTAG / b
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
'ignorada' p
ORIGEN
title
disposicion
doc
ABIERTOS
ENDTAG / p


ENDTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
</title>
λ
ORIGEN
disposicion
doc
ABIERTOS
STARTTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
</title>
'candidata' p
ORIGEN
disposicion
doc
ABIERTOS
TEXT / 'Texto de la disposición'


TEXT / 'Texto de la disposición'
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
</title>
Texto de la disposición.
λ
ORIGEN
p
disposicion
doc
ABIERTOS
ENDTAG / p
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
</title>
Texto de la disposición.

λ
ORIGEN
disposicion
doc
ABIERTOS
ENDTAG / doc


ENDTAG / doc
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
</title>
Texto de la disposición.
</disposicion>
λ
ORIGEN
doc
ABIERTOS
ENDDOC
<doc> Ley 1.
<capitulo><title>
CAPÍTULO I.
DEL REFERENDUM.
</title>
<articulo><title>
Artículo Primero.
</title>
Texto del artículo primero.
</articulo>
<articulo><title>
Artículo Segundo.
</title>
Texto del artículo segundo.
previo a una disposición
</articulo>
</capitulo>
<disposicion><title>
Disposición final.
</title>
Texto de la disposición.
</disposicion>
</doc>
λ
ORIGEN
λ
ABIERTOS

Bibliografía

[1] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, y J. Simeon. Querying documents in object databases. *International Journal on Digital Libraries*, 1(1), 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, y J. L. Wiener. The Lorel query language for semi-structured data. *Internation Journal on Digital Libraries*, 1(1), 1997.

[3] Maristella Agosti, Roberto Colotti, y Girolamo Gradenigo. A two-level hypertext retrieval model for legal data. En *14th ACM-SIGIR International Conference on Research and Development in Information Retrieval*, págs. 316–325, Dipartamento di Elettronica e Informatica, Universita'di Padova, Octubre 1991. Chicago, IL USA.

[4] James Allan. Automatic Hypertext Link Typing. En *Hypertext'96, the Seventh ACM Conference on Hypertext*, págs. 42–52, 1996.

[5] J. André, R. Furuta, y V. Quint. Structured documents: What and why? En J. André, R. Furuta, y V. Quint, editors, *Structured Documents*. Cambridge University Press, 1989.

[6] Jacques André, D. Decouchant, V. Quint, y Helene Richy. Vers un atelier editorial pour les documents structures. En *Congrès AFCET Bureautique, Document, "Groupware", Multimédia*, págs. 63–72, Versailles (France), Junio 1993.

[7] Jacques André, Richard Furuta, y Vincent Quint, editors. *Structured documents*, volume 2 of *The Cambridge series on Electronic Publishing*, Cambridge, New Rochelle, Melbourne, 1989. Cambridge University Press.

[8] Heimburger Anneli. A Structured Link Document as a new means for composing and publishing technical customer documentation in extranets and intranets. En *Electronic Publishing'99*, 1999.

[9] AQUARELLE. The Information Network on the Cultural Heritage. http://aqua.inria.fr/Aquarelle/.

[10] William Y. Arms. Key Concepts in the Architecture of the Digital Library. *D-Lib Magazine*, Julio 1995.

[11] William Y. Arms, Christophe Blanchi, y Edward A. Overly. An Architecture for Information in Digital Libraries. *D-Lib Magazine*, Febrero 1997.

[12] Timothy Arnold-Moore, Phil Anderson, y Ron Sacks-Davis. Managing a digital library of legislation. En *2nd ACM International Conference on Digital Libraries, ACM DL 1997*, págs. 175–183, Philadelphia, PA, USA, Julio 1997. ACM Press.

[13] Timothy Arnold-Moore, Michael Fuller, Alan Kent, Ron Sacks-Davis, y Neil Sharman. Architecture of a content management server for XML document applications. En *1st International Conference on Web Information Systems Engineering (WISE 2000)*, Hong Kong, Junio 2000.

[14] Timothy Arnold-Moore, Michael Fuller, y Ron Sacks-Davis. Approaches for structured document management. En *Markup Technologies'99*, Philadephia, PA, USA, Diciembre 1999.

[15] Timothy Arnold-Moore, Michael Fuller, y Ron Sacks-Davis. System architectures for structured document data. *Markup Languages: Theory and Practice*, 2(1):15–44, 2000.

[16] M. Baldonado, C. Chang, L. Gravano, y A. Paepcke. The Stanford Digital Library Metadata Architecture. *International Journal of Digital Libraries*, 2(1), Febrero 1997.

[17] Michelle Baldonado, Q. Wang, y Terry Winograd. SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests. En *Proceedings of the Conference on Human Factors in Computing Systems, CHI'97*, págs. 11–18, Atlanta, Ga., 1997. ACM Press, New York.

[18] Len Bass, Paul Clements, y Rick Kazman. *Software Architecture in Practice.* Addison Wesley, 1998.

[19] Abdel Belaid y Amos David. The use of Information Retrieval Tools in Automatic Document Modeling and Recognition. En *Tenth International Workshop on Database and Expert Systems Applications (DAUD'99), Florence, Italia*, págs. 522–526, Septiembre 1999.

[20] Donna Bergmark. An Architecture for Reference Linking. Presentation of the Cornell Digital Library Research Group, Mayo 2000.

[21] Mark Bernstein. An apprentice that discovers hypertext links. En N. Streitz, A. Rizk, y J. André, editors, *Hypertext: Concepts, Systems and Applications.*, The Cambridge Series on Electronic Publishing, págs. 212–223, Noviembre 1990.

[22] S. Biagioni, J.L.Borbinha, R. Ferber, P. Hansen, S. Kapidakis, L. Kovacs, F. Roos, y A. M. Vercoustre. The ERCIM Technical Reference Digital Library. En *Second European Conference on Research and Advanced Technology for Digital Libraries*, Septiembre 1998.

[23] Stefania Biagioni, Carlo Carlesi, y Donatella Castelli. Supporting retrieval by 'relation amog documents' in the ERCIM Technical Reference Digital Library. En *11th ERCIM Database Research Group Workshop on Metadata for Web Databases*, Mayo 1998.

[24] William P. Birmigham. An agent-based architecture for digital libraries. *D-Lib Magazine*, Julio 1995.

[25] William P. Birminghan. An Agent-Based Architecture for Digital Libraries. *D-Lib Magazine*, Julio 1995.

[26] William James Blustein. *Hypertext Versions of Journal Articles: Computer-aided linking and realistic human-based evaluation.* PhD thesis, University of Western Ontario, London, Ontario, Canada, 1999.

[27] S. Bonhomme y C. Roisin. Transformations de documents électroniques. *Document Numérique*, 1(3), 1997.

[28] Stéphane Bonhomme. *Transformation de documents structurés, une combinaison des approches explicite et automatique.* PhD thesis, Université Joseph Fourier (Grenoble), Diciembre 1998.

[29] Tim Bray. Comparison of SGML and XML. World Wide Web Consortium Note 15-December-1997.

[30] Heather Brown. Standards for structured documents. *The Computer Journal*, 32(6):505–514, Diciembre 1989.

[31] Priscilla Caplan y William Y. Arms. Reference Linking for Journal Articles. *D-Lib Magazine*, 5(7/8), 1999.

[32] L. A. Carr, W. Hall, y S. Hitchcock. Link Services or Link Agents? En *9th ACM Conference on Hypertext and Hypermedia*, 1998.

[33] Leslie Carr, Wendy Hall, y David De Roure. The evolution of hypertext link services. *ACM Computing Surveys*, 31(4), Diciembre 1999.

[34] Wojciech Cellary y Geneviève Jomier. *Building an object-oriented databse system. The story of O_2.*, capítulo Consistency of Versions in Object-Oriented Databases. Number 19 in The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1992.

[35] Sudarshan S. Chawathe, Serge Abiteboul, y Jennifer Widom. Managing historical semistructured data. *Theory and Practice of Object Systems*, 5(3):143–162, Agosto 1999.

[36] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, y Jennifer Widom. Change detection in hierarchically structured information. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2), 1996.

[37] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Control*, 2(3):113–124, 1956.

[38] Martin Choquette, Daniel Poulin, y Paul Bratley. Compiling Legal Hypertexts. En Norman Revell y A. Min Tjoa, editors, *Database and Expert Systems Applications, 6th International Conference, DEXA'95*, volume 978 of *Lecture Notes in Computer Science*, págs. 449–458. Springer, Septiembre 1995.

[39] Jeff Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, 1987.

[40] V. Cristophides, S. Abiteboul, S. Cluet, y M. Scholl. From structured documents to novel query facilities. En *ACM SIGMOD Conference on Management of Data*, págs. 312–324, 1994.

[41] *CVS. Concurrent Versions System.* http://www.cvshome.org/docs/manual/.

[42] J. R. Davis. Creating a Networked Computer Science Technical Report Library. *D-Lib Magazine*, Septiembre 1995.

[43] J. R. Davis y C. Lagoze. A protocol and server for a distributed digital technical report library. Technical Report TR94-1418, Computer Science Department, Cornell University, 1994.

[44] J. R. Davis y Carl Lagoze. The Networked Computer Science Technical Report Library. Technical report, Cornell University, 1996.

[45] James R. Davis y Carl Lagoze. NCSTRL: Design and Deployment of a Globally Distributed Digital Library, 1999.

[46] Herbert Van de Sompel y Patrick Hochstenbach. Reference linking in a hybrid library environment. *D-Lib Magazine*, 5, Abril 1999.

[47] Steven J. DeRose. Expanding the Notion of Links. En Norman Meyorwitz, editor, *Proceedings of Hypertext'89*, págs. 249–255, Pittsburgh, PA Baltimore, 1989. Association for Computing Machinery Press.

[48] Steven J. DeRose y David G. Durand. The TEI Hypertext Guidelines. *Computing and the Humanities*, 29(3), 1995.

[49] Steven J. DeRose, C.M. Seperberg-McQueen, y Bill Smith. Queries on Links and Hierarchies. En *Proceedings of QL'98 - The Query Languages Workshop* , Boston, december 1998.

[50] Steven J. DeRose, C.M. Seperberg-McQueen, y Bill Smith. XQuery: A unified syntax for linking and querying general XML documents. En *Proceedings of QL'98 - The Query Languages Workshop* , Boston, december 1998.

[51] Jack Doggen. FORMEX V3L Tagging the Laws: SGML Used for Complex Multilingual Documents. En *SGML'96: Celebrating a Decade of SGML*, Boston, 1996.

[52] Bob DuCharme. Links That Are More Valuable Than the Information They Link? *XML.com*, Julio 1998.

[53] Bob DuCharme. What XLink Can Do for Your Applications. *XML Magazine*, Spring 2000.

[54] Jacques Ducloy. DILIB, une plate-forme XML pour la génération de serveurs WWW et la veille scientifique et technique. *MicroBulletin du CNRS*, págs. 3–9. http://www.loria.fr/projets/DILIB/dilib-0.2/DOC/index.html.

[55] Eulegis. http://www.eulegis.net.

[56] Nicholas Finke. TEI Extensions for Legal Text. En *Text Encoding Initiative Tenth Anniversary User Conference*, Providence, Rhode Island, USA, Noviembre 1997.

[57] Edward A. Fox y Robert K. France. Architecture of an Expert System for Composite Document Analysis, Representation, and Retrieval. *International Journal of Approximate Reasoning*, 1(2):151–175, Abril 1987.

[58] R. Furuta y P. D. Stotts. Object structures in paper documents and hypertexts. En *Workshop on Object-Oriented Document Manipulation (WOODMAN'89)*, Rennes, France, Mayo 1989.

[59] Richard Furuta. Concepts and models for structured documents. En J. André, R. Furuta, y V. Quint, editors, *Structured Documents*, págs. 7–38. Cambridge University Press, 1989.

[60] Richard Furuta y P. David Stotts. Specifying structured document transformations. En J.C. van Vliet, editor, *Document Manipulation and Typography*, págs. 109–120, Nice (France), Abril 1988.

[61] George H.Brett II. An Integrated System for Distributed Information Services. *D-Lib Magazine*, Diciembre 1996.

[62] Rosa Maria Di Giorgi y Roberta Nannucci. Hypertext systems for the law. En *Informatique et droit/ Computers and law*, Montreal, 1992.

[63] L. Gravano, C.C.K.Chang, H. Garcia-Molina, y A. Paepcke. STARTS. Stanford Protocol Proposal for Internet Retrieval and Search. En *ACM SIGMOD Conference on Management of Data*, 1997.

[64] Luis Gravano, Héctor García-Molina, y Anthony Tomasic. The effectiveness of *GlOSS* for the text-database discovery problem. En *Proceedings of the International Conference on Management of Data*, Mayo 1994.

[65] Georg Haider, Cecilia Magnusson Sjöberg, Gerald Quirchmay, y Verena Sebald. The Comparative Part of the Corpus Legis Project - Using SGML for Intelligent Information Retrieval of Legal Documents. En A.Ñiku-Lari., editor, *EXPERSYS-96, Artificial Intelligence Applications.*, Technology Transfer Series, págs. 181–186, 1996.

[66] Frans C. Heeman. Granularity in structured documents. *Electronic Publishing*, 5(3):143–155, Septiembre 1992.

[67] S. Hitchcock, L. Carr, S. Harris, J.M.N. Hey, y W. Hall. Citation Linking: Improving Access to Online Journals. En *Second ACM International Conference on Digital Libraries*, págs. 115–122, 1997.

[68] Steve Hitchcock. Linking Electronic Journals: Lessons from the Open Journal Project. *D-Lib Magazine*, Diciembre 1998.

[69] John E. Hopcroft y Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation.* Addison Wesley, 1979.

[70] T. Hu. *New methods for robust and efficient recognition of the logical structures in documents.* PhD thesis, Université de Fribourg, 1994.

[71] Karen Hunter. Adding Value by Adding Links. *Journal of Electronic Publishing*, 3, Marzo 1998. http://www.press.umich.edu/jep/03-03/hunter.html.

[72] IFLA Study Group on the Functional Requirements for Bibliographic Recrods. Functional requirements for bibliographic records. Deutsche Bibliotek, Frankfurt-am-Main, 1997. http://www.ifla.org/VII/s13/frbr/frbr.pdf.

[73] Indecs Home Page. http://www.indecs.org/.

[74] La infopista jurídica. http://www.juridica.com/.

[75] International Organization for Standardization, Geneve. *Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML) (ISO 8879:1986)*, 1986.

[76] Eila Kuikka y Martti Penttonen. Tansformation of structured documents with the use of grammar. En *Electronic Publishing*, volume 6, págs. 373–383, dec 1993.

[77] Carl Lagoze y David Fielding. Defining Collections in Distributed Digital Libraries. *D-Lib Magazine*, Noviembre 1998.

[78] La Ley - Web Site. http://www.laley.net.

[79] LEGGIO - Noticias Jurídicas. http://www.leggio.com/.

[80] Légifrance. L'essentiel du droit français. http://www.legifrance.gouv.fr.

[81] Legislación Básica del Estado - Centro de Información Administrativa - MAP - España. http://www.igsap.map.es/cia/dispo/Ibe.htm.

[82] Barry M. Leiner. The NCSTRL Approach to Open architecture for the Confederated Digital Library. *D-Lib Magazine*, Diciembre 1998.

[83] Seung-Jin Lim y Yiu-Kai Ng. WebView: A Tool for Retrieving Internal Structures and Extracting Information from HTML Documents. En *Sixth International Conference on Database Systems for Advanced Applications (DASFAA)*, págs. 71–78, Hsinchu, Taiwan, Abril 1999. IEEE Computer Society.

[84] G. Lindén. *Structured Document Transformations*. PhD thesis, Dept. of Computer Science, University of Helsinki, 1997.

[85] Philippe Martin y L. Alpay. Conceptual structures and structured documents. En Peter W. Edlund, Gerard Ellis, y Graham Mann, editors, *Conceptual Structures: Knowledge Representation as Interlingua, 4th International Conference on Conceptual Structures, ICCS '96,*, volume 1115 of *Lecture Notes in Computer Science*, págs. 145–159, Sydney, Australia, Agosto 1996. Springer.

[86] Guido De Mets. Consleg Interleaf: SGML Applied in Legislation. En *SGML'96: Celebrating a Decade of SGML*, Boston, 1996.

[87] National Institute of Information Standards, http://sunsite.berkeley.edu/SICI. *SICI: Serial Item and Contribution Identifier Standard*, ANSI/NISO Z39.56 Version 2 edición.

[88] Peter J. Nürnberg, Richard Furuta, John J. Leggett, Catherine C. Marshall, y Frank M. Shipman III. Digital libraries: Issues and architectures. En *Digital Libraries 95*. Center for the Study of Digital Libraries, Texas A&M University, 1995.

[89] Association of Research Libraries. Definition and Purposes of a Digital Library, Octubre 1995.

[90] Andreas Paepcke. Digital libraries: Searching is not enough. *D-Lib Magazine*, Mayo 1996.

[91] Andreas Paepcke, Che Chuan K. Chang, Héctor García Molina, y Terry Winograd. Interoperability for Digital Libraries Worldwide. *Communications of the ACM*, Abril 1998.

[92] H. Van Dyke Parunak. Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning. En *Proceedings of the Third ACM Conference on Hypertext*, págs. 233–242, San Antonio, Texas, USA, Diciembre 1991.

[93] H. Van Dyke Parunak. *Hypertext/Hyermedia Handbook*, capítulo Ordering the information graph, págs. 299–325. Intertext Publications, 1991.

[94] Norman Paskin. DOI: Current Status and Outlook May 1999. *D-Lib Magazine*, Mayo 1999. http://www.dlib.org/dlib/may99/05paskin.html.

[95] James Powell y Edward A. Fox. Multilingual Federated Searching Across Heterogeneous Collections. *D-Lib Magazine*, Septiembre 1998.

[96] S. Ranwez y M. Crampes. Conceptual documents and hypertext documents are two different forms of virtual documents. En *Workshop on Virtual Documents, Hypertext Functionnality and the Web, Eight International World Wide Web Conference, Toronto, Canada*, 1999.

[97] Antoine Rizk y Dale Sutcliffe. Distributed link service in the AQUARELLE project. En Mark Bernstein, Les Carr, y Kasper Østerbye, editors, *8th ACM Conference on Hypertext and Hypermedia*. ACM, 1997.

[98] Martin Roscheisen, Michelle Baldonado, Kevin Chang, Luis Gravano, Steven Kechpel, y Andreas Paepcke. The Stanford InfoBus and Its Service Layers, Agosto 1997. http://www-diglib.stanford.edu.

[99] R. Sacks-Davis, T. Arnold-Moore, y J. Zobel. Database systems for structured documents. En *International Symposium on Advanced Database Technologies and Their Implementation.*, págs. 272–283, Nara, Japan, Octubre 1994. Invited paper.

[100] Bruce Schatz, William Mischo, Timothy Cole, Ann Bishop, Susan Harum, Eric Johnson, Laura Neumann, Hsinchun Chen, y Dorbin Ng. Federated Search of Scientific Literature. *Computer*, 32(2), Febrero 1999.

[101] Cecilia Magnusson Sjöberg. DTD development for the legal domain. En *Swedis SGML 97*, 1997. http://info.admin.kth.se/SGML/.

[102] D. Smith y M. Lopez. Information extraction for semistructured documents. En *Workshop on Management of Semi-structured Data*, Tucson, Arizona, 1997.

[103] C. M. Sperber-McQueen y Lou Burnard, editors. *Guidelines for Electronic Text Encoding and Interchange*. ALLC/ACH/ACL Text Encoding Initiative, 1994.

[104] Stanford Digital Libraries Project. http://www-diglib.stanford.edu.

[105] Steve B. Cousins. A task-oriented interface to a digital library. En *CHI 97 Conference Companion*, págs. 103–104, 1996.

[106] K. Summers. Toward a taxonomy of logical document structures. En *Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies (DAGS '95)*, págs. 124–133. Boston, 1995.

[107] Kristen Maria Summers. *Automatic Discovery of Logical Document Structure*. PhD thesis, Cornell University, aug 1998.

[108] Janet Verbyla. Unlinking the Link. *ACM Computing Surveys*, 31(4):34–, Diciembre 1999.

[109] Anne-Marie Vercoustre y François Paradis. Reuse of Linked Documents through Virtual Document Prescriptions. *Lecture Notes in Computer Science. Lecture Notes in Artificial Intelligence*, Mayo 1998.

[110] W3C, the World Wide Web Consortium. *Cascading Style Sheets, level 2 CSS2 Specification. W3C Recommendation 12-May-1998.*

[111] W3C, the World Wide Web Consortium. *Extensible Stylesheet Language (XSL). Version 1.0.* W3C Working Draft 27-March-2000.

[112] W3C, the World Wide Web Consortium. *HTML 4.01 Specification.*

[113] W3C, the World Wide Web Consortium. *XHTML[tm] 1.0: The Extensible HyperText Markup Language. A Reformulation of HTML 4 in XML 1.0.* W3C Recommendation 26-January-2000.

[114] W3C, the World Wide Web Consortium. *Namespaces in XML*, Enero 1999. W3C Recommendation. http://www.w3.org/TR/REC-xml-names.

[115] W3C, the World Wide Web Consortium. *XML Path Language (XPath)*, Noviembre 1999. W3C Recommendation. http://www.w3.org/TR/1999/xpath.

[116] W3C, the World Wide Web Consortium. *XML Pointer Language (XPointer)*, Diciembre 1999. W3C Working Draft. http://www.w3.org/TR/xptr.

[117] W3C, the World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Second Edition)*, Octubre 2000. W3C Recommendation. http://www.w3.org/TR/REC-xml.

[118] W3C, the World Wide Web Consortium. *Resource Description Framework (RDF) Schema Specification 1.0*, Marzo 2000. W3C Candidate Recommendation 27 March 2000. http://www.w3.org/TR/rdf-schema.

[119] W3C, the World Wide Web Consortium. *XML Linking Language (XLink)*, Febrero 2000. W3C Working Draft 21-February-2000. http://www.w3.org/TR/2000/WD-xlink-20000221.

[120] Norman Walsh y Leonard Muellner. *DocBook: The Definitive Guide.* O'Reilly, 1st edición, Octubre 1999.

[121] Eve Wilson. Links and structures in hypertext databases for law. En Antoine Rizk, Norbert A. Streitz, y J. André, editors, *European Conference on Hypertext, ECHT'90*, The Cambridge Series on Electronic Publishing, págs. 194–211, Paris (France), 1990. Cambridge University Press.

[122] XSilfide (Serveur Interactif pour la Langue Francaise, son Identité, sa Diffusion et son Etude). http://www.loria.fr/projets/XSilfide/.

[123] Yi Xu. *An Incremental Apporach to Document Structure Recognition.* PhD thesis, GMD - Forschungszentrum Informationstechnik GmbH, 1998.

[124] Z39.50 Maintenance Agency. *ANSI/NISO Z39.50-1995. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*, Julio 1995.