Relationship-based dynamic versioning of evolving legal documents

M. Mercedes Martínez[1], Pablo de la Fuente[1], Jean-Claude Derniame[2], and Alberto Pedrero[3]

[1] Universidad de Valladolid, Edificio TIT, Campus "Miguel Delibes" s/n, 47011 Valladolid (Spain)
{mercedes,pfuente}@infor.uva.es
[2] LORIA, BP 239
54506 Vandoeuvre Cedex, France
derniame@loria.fr
[3] Universidad Pontificia de Salamanca, Salamanca (Spain)
apedrero@upsa.es

Abstract. Rule evolution is usually performed by creating a new document which explicitly details changes to *specific* parts inside other rules's content. Obtaining (virtual) document versions corresponding to a rules's state at a specific date is thus left to document users, who manually extract from library collections, and compose, the pieces of text needed to obtain the desired version. When changes are numerous this can be a tedious task. We propose a solution to dynamically *generate* virtual rule versions on user demand, respecting the library documents integrity. References to other documents and modification relationships can be automatically detected and are modelled as typed links –modelled with XLink– in a relationships graph. This graph can be used to query relationships, to create hypertext, and to dynamically generate rule versions. In this paper, we focus on the version generation process: a dynamic document composition based in a graph traversal, during which we intelligently infer the composition rules of the desired version.

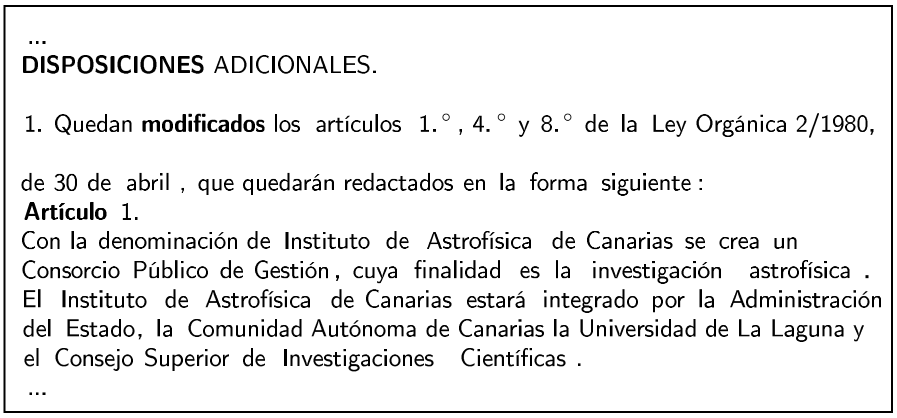
Keywords: Rule evolution, virtual document, document versions, document relationships, dynamic document generation, relationship graph, typed links, XLink, structured documents, document composition.

1 Introduction

Legislative documents are intensively related (references to other documents are numerous) and rules can be the target of temporal sequences of partial amendments, that result in different versions of the document, valid during the period of time between two modifications. To be able to access the version of a document as it was valid at a certain moment is crucial –but not evident– for analysing some other documents [23]. For example, to understand a sentence, it is necessary to read the rules that justify it, as they were valid at the moment the sentence was made; neither previous versions of the rule, nor later versions can be used.

If we consider this problem from the digital library perspective, the new rule version obtained by the application of changes to the initial version is a virtual one, in the sense that the library users know it exists, but there is no physical copy of it available. Indeed, even if not physically, in the collective conscience of library authors and users, the original version, the modified version, and the document containing the modification (which is a separate document with its own identity) coexist. This leads to a problem that can be stated as follows: "Given an abstract document[4] D and the collection of versions of D (also abstract entities), is it possible to access any version of D in this library?"

There is an example in figure 1: a fragment of a Spanish rule, where a modification to a previous rule (*Ley Orgánica 2/1980*) is expressed: the *articles* (*artículos*) 1st, 4th, and 8th have to be replaced by the ones included in this modifying rule. A new version of the document *Ley Orgánica 2/1980* is this way created, even if no physical copy of it is provided.



...
DISPOSICIONES ADICIONALES.

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 :
Artículo 1.
Con la denominación de Instituto de Astrofísica de Canarias se crea un Consorcio Público de Gestión, cuya finalidad es la investigación astrofísica . 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 .
...

Fig. 1. A fragment of modifier rule.

Rule versioning has, however, some additional characteristics, to be considered before facing an automatic treatment of this problem:

1. In legislative documents, modifications are embedded inside other documents, so that the document to be modified is referenced while how it should be modified is expressed later. A *modification* is always "geographically" close to a *reference.*

[4] An "abstract document" is the intellectual entity a user or creator has in mind. It may or not correspond unidirectionally to any of the physical objects stocked in the library's database.

2. A document is modified by the replacement of some of its fragments by fragments coming from other documents. Modifications are not available as complete documents.
3. A document can modify several other documents, and modifications to a given document can come from various sources.
4. Legal documents are highly structured (their component elements can be accurately addressed), and this structure is used in references to rules portions to be modified. In the example above, it was clearly stated which are the specific fragments (*artículos*) to be replaced.

We can appreciate some of these peculiarities in the example in figure 2. It is extracted from a Spanish legal documents digital library. The upper left document (*Real Decreto 685/1982*) –a decree– is the target of several modifications, ennonced in the bottom document (*Real Decreto 775/1997*). The fragments to modify are referenced (by structure –*apartados* inside *párrafos* inside *artículos*–), and it is also indicated *how* to modify them. The application of these modifications deals to a new version of the *Real Decreto 685/1982*, as shown in the upper right corner of the figure. Even if we can not see it in the figure, the modifier document (*Real Decreto 775/1997*) modifies some other parts of *Real Decreto 685/1982* and also some other laws.

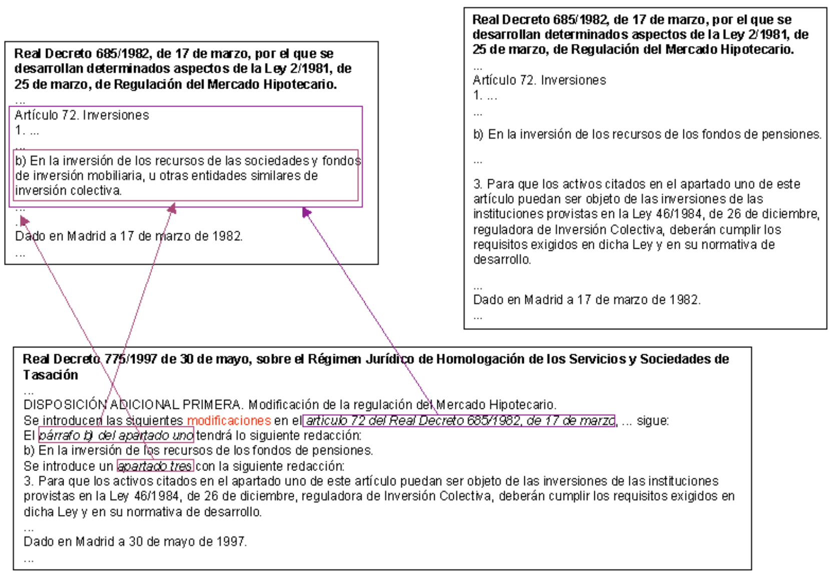
In this paper we present a proposal to automate the obtaintion of rule versions (in figure 2, the document in the upper left corner), by dynamically composing them. We deal with this problem as a **relationship treatment** functionality. Our thesis is that rule versions are due to modifications made to previous versions (*historical versions* [11]), which, in the end, are *just another type of relationship.*

Every version of a document shows the resulting document version after applying all modifications made to the document in the time interval from document creation and requested version date to the original version, extracted from modifier documents. The user specifies in the inputs the document to retrieve and the date. The presumption is that the initial version of the document is available, as well as documents that hold modifications to it. Version generation is tackled as a graph problem: composition rules that allow obtaining new documents are dynamically inferred during a graph traversal.

The approach presented in this paper works on structured documents[5], in which modifications concern document fragments whose limits can be specified in terms of the document structure. With structured documents, it is feasible to have a *structural* graph, whose vertices are node sets (sets of nodes inside the document's tree); to this graph, we add *reference* and *modification* relationships, detected inside documents content. The resulting graph is used to query relationships, to create navigational hypertext, and to dynamically generate rule versions.

Relationships can be automatically detected by analysing documents content (searching *references*), and stored in a links database, that contains the information about the relationships graph. We profit from the expressiveness

[5] for example, XML documents



Real Decreto 685/1982, de 17 de marzo, por el que se desarrollan determinados aspectos de la Ley 2/1981, de 25 de marzo, de Regulación del Mercado Hipotecario.
...
Artículo 72. Inversiones
1. ...
...
b) En la inversión de los recursos de las sociedades y fondos de inversión mobiliaria, u otras entidades similares de inversión colectiva.
...
...
Dado en Madrid a 17 de marzo de 1982.
...

Real Decreto 685/1982, de 17 de marzo, por el que se desarrollan determinados aspectos de la Ley 2/1981, de 25 de marzo, de Regulación del Mercado Hipotecario.
...
Artículo 72. Inversiones
1. ...
...
b) En la inversión de los recursos de los fondos de pensiones.
...
3. Para que los activos citados en el apartado uno de este artículo puedan ser objeto de las inversiones de las instituciones provistas en la Ley 46/1984, de 26 de diciembre, reguladora de Inversión Colectiva, deberán cumplir los requisitos exigidos en dicha Ley y en su normativa de desarrollo.
...
Dado en Madrid a 17 de marzo de 1982.
...

Real Decreto 775/1997 de 30 de mayo, sobre el Régimen Jurídico de Homologación de los Servicios y Sociedades de Tasación
...
DISPOSICIÓN ADICIONAL PRIMERA. Modificación de la regulación del Mercado Hipotecario.
Se introducen las siguientes modificaciones en el *artículo 72 del Real Decreto 685/1982, de 17 de marzo*, ... sigue:
El *párrafo b) del apartado uno* tendrá lo siguiente redacción:
b) En la inversión de los recursos de los fondos de pensiones.
Se introduce un *apartado tres* con la siguiente redacción:
3. Para que los activos citados en el apartado uno de este artículo puedan ser objeto de las inversiones de las instituciones provistas en la Ley 46/1984, de 26 de diciembre, reguladora de Inversión Colectiva, deberán cumplir los requisitos exigidos en dicha Ley y en su normativa de desarrollo.
...
Dado en Madrid a 30 de mayo de 1997.
...

Fig. 2. Dynamic generation of updated versions of legislative documents.

that characterises XML and its associated standards (XLink, XPath, XPointer) to model document structure, to access document fragments –preserving documents integrity– and to represent semantic relationships in the most accurate manner.

Section 2 gives a quick overview of aspects related with legal document versioning. Section 3 presents the graph relationship treatments are based on. Section 4 describes how this graph is exploited to generate rule versions. In section 5, we discuss implementation aspects: what the data architecture and format should be, to make feasible a translation of algorithms to operating applications.

2 Document versioning and relationships

2.1 The document versions problem

The document versioning problem has been treated in legislative digital libraries in three ways:

1. Maintaining simultaneously all versions as individual documents in system databases [9, 20]. Versions of the same document are linked by revision links.

Its main difficulty is to keep these links up to date, as well as links that affect documents (which may, in consequence, affect all the versions of the involved document).
2. Modelling modifications as attributes. This solution is, however, compatible with the previous solution. These attributes are considered as "links", whose resolution (obtaining the link's target) is left to the user [12]. Its main aim is to facilitate queries about the "history" of a document (changes made to it): the user can know the document has been changed and where to find the changes; however, it is up to the user to obtain the versions if this is his/her wish.
3. Keeping the rules that allow the generation of document versions [6]. For every version there are associated rules that allow it to be generated automatically at user's request.

2.2 Modelling relationships

Relationships between documents can have varied causes and meanings. They can be *semantic* (documents that share the subject, author, ...), they can be *referential* (derived from *references* inside documents to other documents), or they can be *explicit* links (as the case of HTML links) [1][6].

Relationships can be represented by a *hypertext* graph, which allows users to "navigate" within related (linked) documents [10]. Other option is to model relationships in the documents metadata –this is done, for example, when using metainformation standards as the *Dublin Core* [7]– or as links. There are examples in the legal domain of using hypertext to model relationships [2, 9, 14, 15].

2.3 Structured documents and relationships between document *fragments*

Documents made by composing well–delimited pieces of content (that can present an inclusion hierarchy between them) are said to have a *logical structure*, and called *structured documents* [3]. This structure is represented with a *tree* model, where internal nodes are document elements (fragments), and leaves are the document content [13].

3 A relationship graph to represent relationships between documents

In this section we present the conceptual model that allows us to manage relationship information (references and other relationships derived from references),

[6] *Semantic* and *referential* relationships can be considered both *implicit* relationships, by contrast with *explicit* relationships where there is a mark that explicits the existence of a relationship (see [1]).

and in which we base the solution for rule versioning explained in section 4. All the information we deal with at this level are *abstract* entities, that correspond to the entities that library users (who are supposed to have no knowledge about computing information systems) would expect to found in the library [4, 19]. For example, rules, decrees, etc.

The relationships that receive our main attention are those that derive from *references* between documents. These relationships can be recognised by the presence of a citation or reference in the text of a document `A` to some portion of some other document `B`. `A` and `B` are related documents. The reference can be merely a citation in `A` to some portion of `B`, or a reference to some portion of `B` indicating how to *modify* the portion referenced (The first *disposición* of the rule in figure 1 references articles 1, 4 and 8 –to afterwards say how to modify them– of the *Ley Orgánica 2/1980*). This granularity is a very important feature, especially in the virtual document generation shown in section 4. These cases where modifications are extracted from document content have the property that modifications are always "geographically" close to a citation that designates the target of the modification that follows. There are two heterogeneous links (a citation and a modification) that share the target, but which have different origins.

3.1 Obtaining a relationship graph

We model the information about relations in a relationship graph. The graph is constructed from *references* detected inside document content.

This places us in a situation such that we are able to develop a complete automatic treatment of relationships (from detection to exploitation). To have a graph that can be later exploited, it is needed:

1. To obtain a **graph**, whose nodes correspond to the fragments of documents referenced (or which reference). A document *normalisation* step provides what will be the nodes of the relationship graph. As a result, a 'normalised' structured document (XML) is obtained, whose elements correspond to abstract document fragments. The normalisation is done with a linguistic analysis of the document, detecting starts of elements by the presence of vocabulary keywords and expressions that are systematically used in document content. The graph is constructed on the forest formed by related documents trees (we call it *structural graph*).
2. **Detecting relationships**. This step is based on similar principles than normalisation: a linguistic analysis of documents, in which references are recognised. Linguistic and syntactic rigidity in this type of texts eases the automatic detection of citations [23], by comparison with other contexts more 'flexible' in their language structures.
   To the structural tree forest, we add some more labelled arcs that represent other types of relationships. The label of the arc depends on the type of relationships (this way the semantic of the relationship is considered).

Thus, the resulting graph is formed as follows:

1. Arcs from the document tree represent the hierarchical relationships between nodes inherent to the document logical structure. These are *structural* arcs and are not labelled in the examples in figures 3 and 4, but drown in bold.
2. Arcs that represent a citation in the origin to the target have the label *citation*. They represent citation links and they connect tree nodes.
3. Arcs associated to a modification link are labelled *modification*. They also connect tree nodes.
4. Other types of relationships would be represented by corresponding typed links. They are not needed for the versioning, thus, we will obvious them hereafter.

Document D in figure 3 is composed of three elements (d_1, d_2,d_3); element d_2 is itself formed by two other elements (d_{21},d_{22}). Document P is composed of three elements (p_0, p_1, p_2); element p_1 is itself formed by two other elements (p_{11},p_{12}). Similarly, documents T and N are composed by elements (t_1,t_2) and (n_1, n_2, n_3) respectively. The figure shows a relationship graph, where these *structural* relationships (bold lines in the figure) and *modification* relationships, m_1, m_2, m_3 and m_4 (dashed arrows in the figure), are represented.

3.2 Manipulating the graph

To manipulate information in this graph, we need tools that allow to **address** document fragments we are interested in, and to store the information about the graph in a digital format. XML related standards, *XLink*, *XPointer* and *XPath*, have the required characteristics.

4 Dynamic generation of document versions, exploiting the relationship graph

We introduce in this section the process we follow to obtain a copy of a document in its valid state at a certain date. The dynamic generation of versions relies on an exploitation of the relationship graph, in which the composition rules of the desired versions are dynamically inferred.

4.1 Versioning graphs

Every document version is the result of resolving a versioning graph obtained from the modification graph (the graph that contains all modification relationships). The modification graph is a subgraph of the relationship graph presented in section 3.

The versioning graph associated to the requested document version is obtained with:

1. The tree, T, of the document version available in the document database. This version is the source document for the versioning process.

2. The set of modification links, M, that reach some node in T.
3. Modification links that reach some source node of links in M should also join M.

Modification links in the versioning graph have a priority attribute: the date of the link (the date when the modification was stated). This priority attribute will be used to resolve conflicts such as that found when a node is affected by more than one modification (see part 4.2).

The graph in figure 3 is in fact a versioning graph. D is the document to be versioned. The tree with D in the root represents the version available in the document database. Modification links m_1, m_2 and m_3 directly reach nodes in D. As for modification m_1, as its target is itself the source of a modification link reaching D, it also is a composing item of the graph.

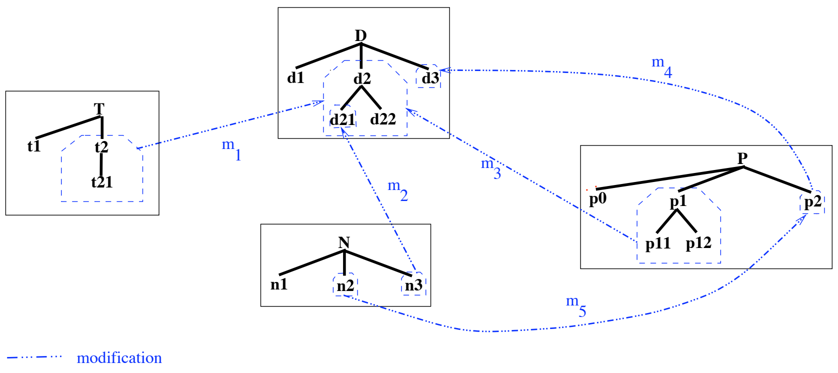


Fig. 3. Versioning graph.

4.2 The output version

A document is considered as a tree with elements. The tree that represents the output document from the version generation process is at its basis the same as the initial one (that associated to the initial version), where some nodes have been replaced, deleted or inserted. Nodes in the source document (initial version available in the library) not affected by any modification (nodes that are not the target of a link) remain untouched in the output version.

The document version generation algorithm deals with the source document tree in a recursive manner, beginning at the root node and treating its descendants in the same manner till there are no more nodes to consider. When the algorithm reaches a node, this can be in one of two possible categories:

– It is a node affected by a modification (it is the root of some modification link's target). The modification link is *resolved.*

- It is not affected by modifications (there are no modification links that reach it).

In the example in figure 3, node d_1 is not affected by any modification, while nodes d_2 and d_3 are both affected by some modification, which should be resolved.

To *resolve* a modification link is to apply the modification expressed by it. That is, to *replace* the target vertex content by the source vertex content. When the current node is affected by some modification, there can be several situations to consider:

- *Simple case.*
  The node is affected by a unique modification link, l. Treatment of the node limits to resolve l, that is, to substitute the node by l's origin.
- *Transitive modifications.*
  There is a sequence of historical modifications: the origin node of some modification is itself the target of other modifications (it is modified somewhere else). Transitivity implies that the target will be replaced by the node at the end of the modifications chain.
- *Modifications overlapping.*
  There is a conflict due to the fact that a node set is the target of several modifications. When all modifications apply exactly to the same node set it is resolved by applying the *priority* criteria to the modifications: only the most recent is applied[7]. In other cases, there can be further criteria to consider; a detailed explanation of these situations can be found in [17].

4.3 An example

Let´s take the versioning graph in figure 3 and briefly describe the way the algorithm process it.

Modications m_1 and m_3 affect the same document fragment (d_2); there is a conflict –only one of them has to apply– that is resolved using the date criteria (the most recent one is applied). On the other hand, the resolution of m_2 has to take in count the transitive modification (m_2's origin is itself modified in document N); m_2's target will be replaced in the end by m_4's origin.

The application of these modifications to D gives as result a new version of D, obtained as follows:

- Document T is more recent than document P, what means the modification expressed in T (m_1) gets relevant over the modification expressed in P (m_3): element t_2 (with its descendant t_{21}) replaces element d_2 in D.

[7] A sequential application could, of course, be envisaged, and it would be the only way to resolve modifications in case there were no clear priority criteria. The result being the same, we prefer to take profit from information priority criteria provides us with, to reduce algorithm steps. It is to note that –at least in our experience– *exact* overlapping can always be resolved on time criteria.

– Element n_2 replaces element d_3 in D. This modification is the transitivity (m_2, m_4) .

The version generation algorithm starts with node D. A modification to this node would suppose a replacement of the whole document by another document or document fragment. As this node is not affected by any modification, the algorithm continues exploring the possibility that its descendants –d_1, d_2 and d_3– are themselves the object of some modification. Node d_1 remains untouched. d_2 is replaced by t_2 and its descendants while d_3 is replaced by n_2, thereby completing the recursion. The resulting document can be seen in figure 4.

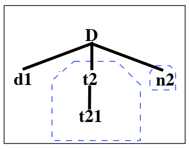


Fig. 4. Output version obtained by resolving the versioning graph in figure 3.

5 Data architecture

The data architecture is conceived to facilitate access of abstract document entities, as well as manipulating the information items modelled in the relationship graph. Abstract document entities determine the document architecture.

5.1 Document architecture

A document is an aggregation of three information elements: document content, metadata that describe it, and document links (see figure 5).

Document content. Document content is stored in a digital copy of the document, whose logical structure reflects the abstract document entity inherent structure. It is a XML document, that adapts to specific DTDs. For Spanish rules, there is a document class, with its own DTD. This copy is obtained in the normalization process (see section 3.1).

Document metadata. Every document in the system is designated by a unique identifier that distinguishes it from other documents in the library universe[8]. The advantages of such an identifier are several, but to deal with references and derived relationships, the main advantages that have been critical in the decision to use such an identifier instead of a physical locator are as follows:

[8] This identifier follows the criteria used in the DOI standard [19].

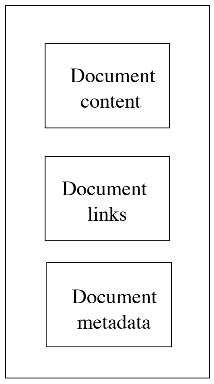


Fig. 5. Documents are the aggregation of three information elements: content, metadata and links.

1. It is possible to deduce such an identifier from citations in document content, but it will never be possible to deduce physical addresses from citations. This property is the one that citation detection (section 3.1) is based on.
2. Links detected from citations will therefore be between abstract document entities, and never between physical copies of the document.

Links. Links represent the arcs in the relationship graph. They are *typed*, on the semantic of the relationship they represent. The link composition derives from the requirements to the links:

1. Links must be able to address internal document fragments.
2. The links database should accept queries about linking information.
3. Links can be queried multidirectionally. A citation or modification can be exploited in both directions. For example, it can be asked *What documents modify this document?* or *What documents are modified by this document?*.
4. There is n-arity in the graph: a node can participate in more than one link. An example: the algorithm in section 4 needs to access the set of nodes that modify a given node.
5. The links database must be accessible separately from the documents: to query this database, it must not be necessary to enter the documents. This condition is necessary to exploit multidirectionality and n-arity.

5.2 Using XLink to model the relationships

XML has associated standards that allow relationships between documents to be modelled. Linking with XML includes rules to link resources (XLink) [21] and to address internal fragments inside linking resources (XPointer) [22]. XLink allows traditional unidirectional links embedded inside a document (as HTML links are), but also more complex links: they can have associated semantic, they

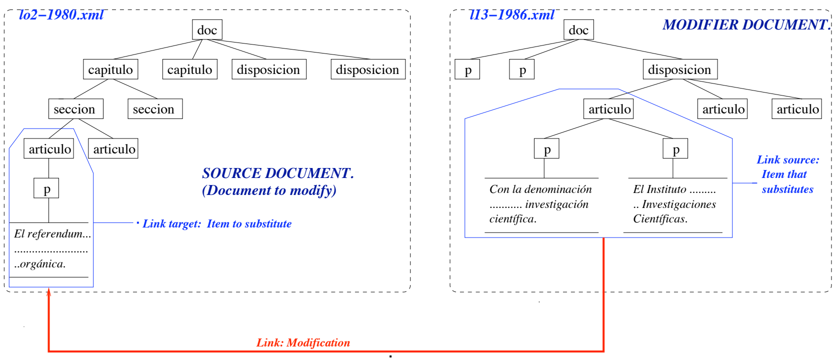


Fig. 6. "Modification" link. The ORIGIN will replace the TARGET when generating a new version of the source document. The TARGET is a subtree of the source document made up of the element `articulo` and all its descendants. The ORIGIN is the subtree in the modifier document tree whose root is the first element `articulo` inside the first `disposicion`, as can be seen in figure 7.

can link multiple resources, and it is possible to create independent link databases [21].

Xlinks are, in consequence, a means to model graph information, accomplishing all the requirements expressed in part 5.1: vertices are resources (documents, images, etc.), arcs in the graph are arcs inside an xlink, arc labels are modelled with `arcrole` attributes and other metadata about arcs or vertices can be joined as (resources or arc) attributes.

Figures 7 and 6 show respectively the code of an xlink and the link it models. In this case, the link relates two node sets (document fragments) extracted from two Spanish rules. It express a modification, consisting in the substitution of a specific *articulo* element (the link's target) by a new one (the link's origin). Its equivalent xlink consists in one ENLACE element, that contains three subelements that completely describe the link composition and semantics. The ORIGEN element specifies the target of the link to be the first *articulo* element inside the document *lo2-1980.xml*. The DESTINO element indicates the origin of the link (the element that should replace the target) to be the first *articulo* element that can be found inside the first *disposicion* of the rule *l13-1986.xml*. The ARCO element tells about the link's type: *substitution*. More semantic about the relationship can be found inside the link's elements: the documents's dates and the documents's classes.

6 Related work

Document evolution has been considered from different perspectives and related with other problems. Most of the times, the perspective taken is to study its impact on the systems that suffer this evolution: maintaining the documents

```
 <ENLACE>
 <ORIGEN  xlink:href= "l13-1986.xml#xpointer(child::disposicion[1]/articulo[1])"
          xlink:label="l13-1986-d1a1"
          date= "1981" doctype= "norma"   />
 <DESTINO xlink:href= "lo2-1980.xml#xpointer(child::articulo[1])"
          xlink:label="lo2-1980-a1"
          date= "1986" doctype= "norma"   />
  <ARCO    xlink:from="l13-1986-d1a1" xlink:to="lo2-1980-a1"
          xlink:arcrole="http://.../substitution"/>
</ENLACE>
```

Fig. 7. Text for the example link.

database [9], obtaining the revision links this evolution provokes [9, 8], representing this evolution [8], or its influence on link evolution –and validity– [20, 18, 5]. As it can be seen, the study is frequently focused on the effect of document versioning on links; that is, a document evolution causes new relationships or degrades them. In all of them, versions are supposed to be manually generated. We have taken a novel perspective to deal with document evolution: what is the impact of relationships on document evolution? Can they be the key information items that allow to know about changes and to obtain the different versions that compose a document evolution?

There are two possibilities for representing changes aside from the one chosen in this work –which were presented in section 2–: storing all versions caused by a change and linking them [9, 20], and representing changes as annotations [8, 12]. The solution of maintaining all versions simultaneously has shown to have its main difficulty in links maintenance. Moreover, this approach presupposes that someone has taken care of generating the different versions. This fact is not always guaranteed, which is the case when modifications that cause different versions to appear come from citations: the documents that should be used to obtain every version are available, but the different versions must be composed by users following the modifications and applying them. This is the reality we deal with. In addition, the problem of maintaining revision links detected in [9, 20] disappears: versions are automatically generated.

As for representing changes as annotations, this facilitates queries about the history of a document, but it is not the most appropriate choice to facilitate version generation. The information about relationships does not appear as a link that can make part of a graph, but as attributes of nodes. In the end, these attributes express a relationship between two nodes. So, our decision has been to model this type of relationship as any other type of relationship -with typed links–, and to generate versions exploiting the relationship graph. If changes are represented by independent links, to query changes is to query links, which is the same as querying any document.

We are only aware of one proposal to automatically generate rule versions [5, 6] (the third option presented in section 2). They keep rules ("*tables of contents*") that describe the composition of document versions, in order to do an efficient automatic generation of document versions. The version generation algorithm presented in this chapter has an important difference in the working principle: it *deduces* the composition rules of document versions from the relationship information, modelled in a relationship graph, and stored in links. It is a *link graph traversal* that allows versions to be generated, with variable parameters (such as the version date); there are no "*tables of contents*" or rules that express the composition rules of the desired version. A specific set of rules for each version has the advantage of its precision and efficiency, but the disadvantage that every version's set of rules has to be specified independently. Versions cannot be generated unless the rules have already been specified, even if all the necessary pieces are in the library. A more general method, such as the one presented in section 4, to automatically infer the rules for generation of new versions allows this to be done, taking advantage of the relationship graph. We have primed respecting documents integrity and functional extensibility of our model (querying relationships and hypertext composition) over efficiency.

Finally, the update of structured documents (concretely, XML documents) has also been considered as the expression of a series of operations in a query language. In [16] a language using logic programming is proposed, on the basis of the advantages of declarative expression of update operations as tree paths (as XPath, which we also use). Not being query languages our main concern, but the resolution of the graph, we find these languages more appropriate in cases where the update of documents is interactively made by users (for example, when users edit the documents to apply changes). The situation with legal documents happens to be different because of the peculiarities explained in the Introduction.

7 Conclusion and future work

An original solution for the dynamic intelligent assembly of new information items has been presented. It is not an isolated solution; the ability to combine the exploitation of relationships from several perspectives is one of the main characteristics of the presented solution: information retrieval (querying relationships), hypertext-oriented exploitation, and dynamic information assembly are softly combined. Dynamic aspects are a main value of the proposal, as it allows us to believe that the solution is open for future integration of other dynamic methods of treating relationships.

The conceptual model used –the relationship graph– has a great advantage: working at the same abstraction level that library's users. This feature allowed us to translate in the most accurate manner the processes users utilise aside from the digital library system to obtain information (and that provoke their demand of new functionalities in digital libraries) to dynamic information assembly algorithms.

Versioning is not a problem exclusive to textual documents, but these do have an interesting peculiarity: modifications are mostly expressed inside documents that are themselves a semantic unit that should not be fragmented. The approach we have chosen is to maintain links separately from document content. This has several benefits: the integrity of documents that contain modifications is untouched (they are never fragmented to obtain individual entities that could be directly inserted in a new document, neither their content, structure or attributes is touched), documents that participate in the relationship are not touched when relationships are detected (the structural forest graph and document contents are not modified by representing relationships, as would an insertion of linking elements inside documents), the problem of maintaining revision links disappears, and the process of version generation can be tackled as an intelligent deduction of composition rules.

The prototype used to test this proposal contains a database of Spanish rules (1665 documents), from which a subset –yet in expansion– is used to test the versioning algorithm. The robustness of the versioning depends in a high level in the accuracy of the expressed relationships. Because of this we still work in the improvement of the detection of relations. To the moment, on a set of 50 documents, 90 % of the references by document identifier were correctly detected (context-based references escaped the detection process). As for the versioning itself, it happens to fail in cases of imprecision in references or overlapping mistmaches, which, within the selected documents, we estimate about 10 % of the cases; however, we do not consider this set of documents large enough to do final statements on this subject. Future work will consider to explore methods that allow a more intelligent resolution of conflicts during version generation, learning from user indications.

References

1. Agosti, M., and Allan, J. Methods and tools for the construction of hypertext. *Information Processing and Management 33*, 2 (1997), 129–271.
2. Agosti, M., Colotti, R., and Gradenigo, G. A two-level hypertext retrieval model for legal data. In *14th ACM-SIGIR International Conference on Research and Development in Information Retrieval* (Dipartamento di Elettronica e Informatica, Universita'di Padova, Oct. 1991), Chicago, IL USA, pp. 316–325.
3. André, J., Furuta, R., and Quint, V. Structured documents: What and why? In *Structured Documents* (1989), J. André, R. Furuta, and V. Quint, Eds., Cambridge University Press.
4. Arms, W. Y., Blanchi, C., and Overly, E. A. An Architecture for Information in Digital Libraries. *D-Lib Magazine* (Feb. 1997).
5. Arnold-Moore, T., Anderson, P., and Sacks-Davis, R. Managing a digital library of legislation. In *2nd ACM International Conference on Digital Libraries, ACM DL 1997* (Philadelphia, PA, USA, July 1997), ACM Press, pp. 175–183.
6. Arnold-Moore, T., Fuller, M., Kent, A., Sacks-Davis, R., and Sharman, N. Architecture of a content management server for XML document applications. In *1st International Conference on Web Information Systems Engineering (WISE 2000)* (Hong Kong, June 2000).
7. Biagioni, S., Carlesi, C., and Castelli, D. Supporting retrieval by 'relation amog documents' in the ERCIM Technical Reference Digital Library. In *11th ERCIM Database Research Group Workshop on Metadata for Web Databases* (May 1998).

8. Chawathe, S. S., Rajaraman, A., Garcia-Molina, H., and Widom, J. Change detection in hierarchically structured information. *SIGMOD Record (ACM Special Interest Group on Management of Data) 25*, 2 (1996).
9. Choquette, M., Poulin, D., and Bratley, P. Compiling Legal Hypertexts. In *Database and Expert Systems Applications, 6th International Conference, DEXA'95* (Sept. 1995), N. Revell and A. M. Tjoa, Eds., vol. 978 of *Lecture Notes in Computer Science*, Springer, pp. 449–458.
10. Conklin, J. Hypertext: An introduction and survey. *IEEE Computer 20*, 9 (1987), 17–41.
11. DeRose, S. J. Expanding the Notion of Links. In *Proceedings of Hypertext'89* (Pittsburgh, PA Baltimore, 1989), N. Meyorwitz, Ed., Association for Computing Machinery Press, pp. 249–255.
12. Finke, N. TEI Extensions for Legal Text. In *Text Encoding Initiative Tenth Anniversary User Conference* (Providence, Rhode Island, USA, Nov. 1997).
13. Furuta, R. Concepts and models for structured documents. In *Structured Documents* (1989), J. André, R. Furuta, and V. Quint, Eds., Cambridge University Press, pp. 7–38.
14. Haider, G., Sjöberg, C. M., Quirchmay, G., and Sebald, V. The Comparative Part of the Corpus Legis Project - Using SGML for Intelligent Information Retrieval of Legal Documents. In *EXPERSYS-96, Artificial Intelligence Applications.* (1996), A. Niku-Lari., Ed., Technology Transfer Series, pp. 181–186.
15. Noticias jurídicas. http://noticias.juridicas.com/.
16. Liu, P., and Hsu, L. H. A Logic Approach to XML Document Update Query Specifications. In *XML Europe 2001* (Berlin, Germany, May 2001).
17. Martínez González, M. *Dynamic exploitation of relationships between documents in digital libraries: application to legal documents.* PhD thesis, Institut National Polytechnique de Lorraine / LORIA, Sept. 2001.
18. Niederée, C., ans Joachim W. Schimdt, U. S., and Matthes, F. Aging Links. In *Research and Advanced Technology for Digital Libraries, 4th European Conference, ECDL 2000* (Lisbon, Portugal, Sept. 2000), J. L. Borbinha and T. Baker, Eds., vol. 1923 of *Lecture Notes in Computer Science*, Springer, pp. 269–279.
19. Paskin, N. DOI: Current Status and Outlook May 1999. *D-Lib Magazine* (May 1999). http://www.dlib.org/dlib/may99/05paskin.html.
20. Sjöberg, C. M. DTD development for the legal domain. In *Swedis SGML 97* (1997). http://info.admin.kth.se/SGML/.
21. W3C, the World Wide Web Consortium. *XML Linking Language (XLink) Version 1.0*, June 2001. W3C Recommendation 27 June 2001. http://www.w3.org/TR/2000/REC-xlink-20000627.
22. W3C, the World Wide Web Consortium. *XML Pointer Language (XPointer) Version 1.0*, Sept. 2001. W3C Candidate Recommendation 11 September 2001. http://www.w3.org/TR/2001/WD-xptr-20010108.
23. Wilson, E. Links and structures in hypertext databases for law. In *European Conference on Hypertext, ECHT'90* (Paris (France), 1990), A. Rizk, N. A. Streitz, and J. André, Eds., The Cambridge Series on Electronic Publishing, Cambridge University Press, pp. 194–211.