A method for the dynamic generation of virtual versions of evolving documents

Mercedes Martínez
Departamento de Informática
Universidad de Valladolid
Campus "Miguel Delibes", s/n
47001 Valladolid (Spain)
mercedes@infor.uva.es

Jean-Claude Derniame
LORIA, BP 239
Vandoeuvre Cedex, France
derniame@loria.fr

Pablo de la Fuente
Departamento de Informática
Universidad de Valladolid
Campus "Miguel Delibes", s/n
47001 Valladolid (Spain)
pfuente@infor.uva.es

ABSTRACT

Document evolution is usually performed by creating a new document which explicitly details changes to *specific* paragraphs inside other document content. Obtaining (virtual) document versions corresponding to its state at a specific date is left to document users, who manually extract from library collections, and compose, the pieces of text needed to obtain the desired version. But this can be a very tedious and difficult task when changes are numerous. We propose a solution to dynamically *generate* virtual document 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 relationship graph. In this paper, we focus on the version generation process, consisting in a dynamic document composition based on a graph traversal. This solution has already shown its adequacy with a legislative digital library.

Keywords

Document evolution, document relationships, dynamic virtual version generation, XLink, structured documents

1. INTRODUCTION

Documents are data that evolve in time. One possible source of evolution comes from the modifications made to a document's content by its author(s), which partially modify the content, what results in different historical document versions. This reality gets specially relevant in some environments as the legislative ones (rules suffer frequent modifications) or collaborative authoring environments (each author can propose a series of modifications to the global work). Among the different possibilities to modify a document, it is frequent that authors describe these modifications as a new document or inside another document. For each modification, its author cites the document fragment he/she wants to change and indicates how the said fragment could be modified (eliminating it, substituting it, ...). The new version obtained by the application of these changes is a virtual one, in the sense that the library users[1] 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[2] D and the collection of versions of D (also abstract entities), is it possible to access any version of D in this library?"

We present a proposal to answer to this problem, consisting in the dynamic generation of document versions, on user demand. The idea is that versions are due, in many cases, to modifications made to previous versions (*historical versions* [9]), which in the end is just another type of relationship. So, if it is possible to query relationships and to work with the associated link graph, it must be possible to dynamically generate versions with a traversal of this graph. Moreover, these relationships can be automatically detected by analysing documents content, and stored in a link database. The solution for the dynamic generation of versions focuses the attention of this paper. It relies on the querying of the semantic relationships that expresses modifications, to dynamically infer the composition rules that allow obtaining new documents. The input documents are structured documents (documents that have well defined internal parts) -as are XML documents-. We profit from the expressiveness that characterises XML and its associated standards (XLink, XPath, XPointer) to access document fragments and to model semantic relationships. We test the version generation method in a legislative digital library.

SAC 2002, Madrid, Spain

[1]Library users are people who access the digital library searching documents (intellectual entities) that match their requirements. These users may be specialists in a domain (for example, jurists), with no special knowledge about computers, who just use the library as one more tool for their work.

[2]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. DOCUMENT VERSIONING AND RELATIONSHIPS

2.1 The document versions problem

Most of the effort related to document versioning is concerned with knowing about the fact that two electronic documents are versions of the same abstract document. Three approaches can be distinguished:

1. To link related versions. These links were named *revision links* by Parunak in 1990 [14]. Two databases are kept simultaneously: the document database and the link database. The main problem with this approach is to keep the revision links database [6, 15] up to date.

2. To consider different stamps of the database and to compare them in order to detect changes that reflect the fact that an object has been versioned [4]. This solution is used with object databases, and therefore can be considered when modelling documents as objects [1]. In this approach the link database disappears and document changes are represented indirectly as the difference between two database states [5].

3. A third approach comes from the area of semistructured data. Chawathe et al. [5] model changes to hierarchically structured data (which is the case of structured XML documents) as changes to nodes in the document tree. They represent changes as annotations (attributes) to the affected nodes, facilitating queries about nodes "history". The detection of versions is done by tree comparisons. In contrast to the previous ones, this is the first solution where document structure is considered, thereby associating changes to document fragments instead of to whole documents.

   This idea of annotating document nodes with attributes that contain information about these changes can be found in some public servers [11], where document elements are qualified with attributes that indicate they have been modified.

2.2 A brief visit to document relationships

Relationships between documents can have varied causes and meanings. They can be semantic (documents that share the subject, author, ...), explicit links, or they can be derived from references inside documents. In all cases, relationships can be represented by a graph, where the resources are the related items and the arcs represent the relationships. If there are heterogeneous relationships, arcs can have *type* that represent the nature of the relationship [16]. If working with structured documents, the hierarchical relationships among document elements can make part of the graph, and link vertices can be node sets (the set of document elements involved in the relationships). This is a usual case when relationships are derived from references: the author referencing another document to comment or modify it, first specifies the portion (set of elements) of interest, to continue with the comment or modification.
Relationships can be modelled in the document metadata - this is done, for example, when using metainformation standards as the *Dublin Core* [3]- or as links. Links can be embedded inside document content -as with HTML hypertext-, or they can be stored in independent link databases.

The more general use of the relationship graph has been the creation of hypertext that users can navigate through [7]. But the graph can be used with more purposes; for example, relationships can be queried [10, 17].

3. RELATIONSHIP GRAPHS

We model relationships with a labelled **graph**, whose nodes correspond to the fragments of documents referenced (or which reference). A document *normalisation*[3] step provides what will be the nodes of the relationship graph (see figure 1). The graph is constructed on the forest formed by related documents (we call it *structural graph*). To this initial graph, 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:

- 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 shown, but are drawn in bold.

- Arcs that represent a citation in the origin to the target have the label *citation*. They represent citation links.

- Arcs associated to a modification link are labelled with *modification*.

4. DYNAMIC GENERATION OF VERSIONS

Every version of a document should show the resulting document version after applying to the original version of the document all modifications made to it in the time interval from document creation and requested version date. 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.
Another characteristic is that modifications are commonly included in some other document, but are not available as complete documents, what introduces a complexity degree by comparison with solutions where modifications are isolated [8]. That is, a document is modified by the replacement of some of its fragments by fragments coming from other documents. Our proposal works on structured documents, in which modifications concern document fragments whose limits can be specified in terms of the document structure (for example, XML documents). With structured documents, it is feasible to have a modification graph whose vertices are node sets (sets of nodes inside the document's tree) that allows the version generation to be tackled as a graph problem. MoDiFiCaTions to be applied are filtered from the total set of modifications using request parameters (for historical versions the filter criteria is the date).
Also, it is proposed to "store" modifications in a link database that contains the information about the relationship graph that involves all documents needed. Obtaining document versions from modifications should be done on demand, by applying modifications expressed in links to the original versions.

[3]Normalisation is done on document content vocabulary, that marks the start of each document part (*element*, if talking in XML vocabulary). More details about this process can be found in [12].

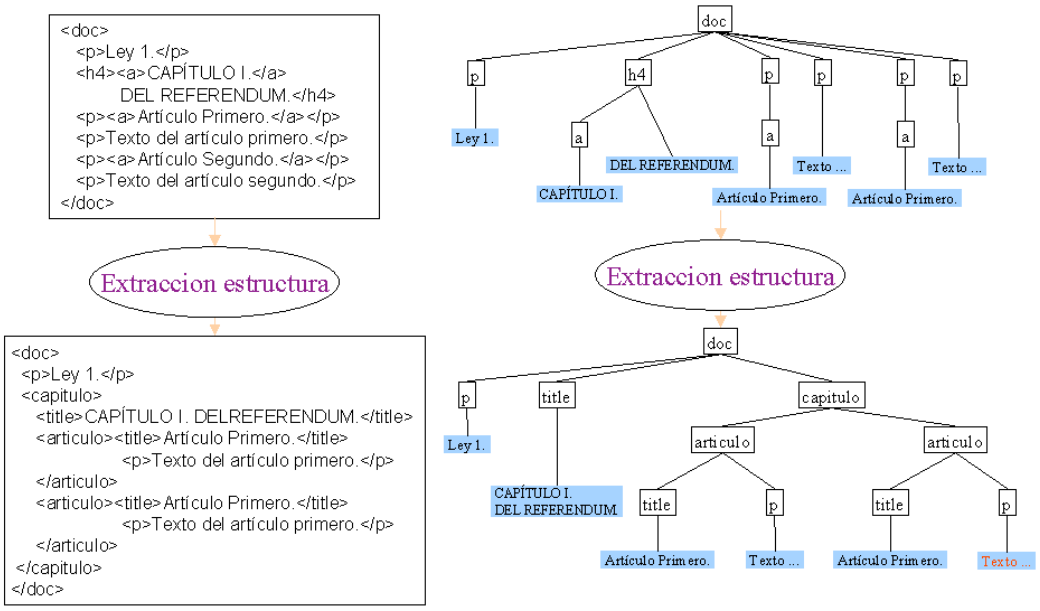

Figure 1: An example of normalised document (the input document is on the top of the figure and the normalised one on the bottom). The logical structure changes, that is, the document tree changes.

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 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 those found when a node is affected by more than one modification (see part 4.2).

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 basically 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 any link) remain untouched in the output version. Structural links (links that relate document elements) and modification links are used to produce the versioning graph, as explained in 4.1.
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 until 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 target). The modification link is *resolved*.
- It is not affected by modifications (there are no modification links reaching it).

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 is limited to resolve l, that is, to substitute the node by l's origin.
- *Transitive modifications.*
  There is a sequence of historical modifications: the

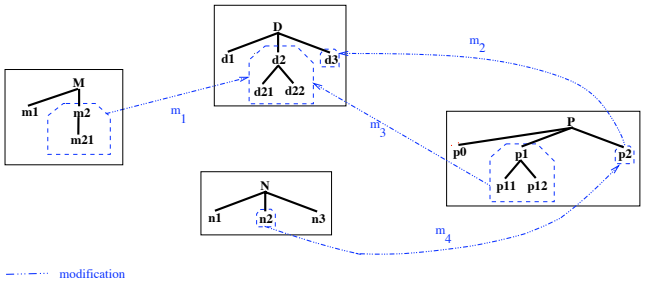

Figure 2: Versioning graph.

origin node of some modification is itself the target of other modifications (it is modified somewhere else).

- *Modifications overlapping.*
  In this case the conflict is due to the fact that a node set is the target of several modifications. When all modifications apply exactly to the same node set the conflict is resolved by applying the *priority* criteria to the modifications: only the most recent is applied. In other cases, there can be further criteria to consider.

4.3 An example

Document D in figure 2 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 has three elements (p_0, p_1, p_2); element p_1 is itself formed by two other elements (p_{11},p_{12}). Similarly, documents M and N are composed by elements (m_1,m_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 shown. m_1 and m_3 affect the same document fragment (d_2), and so there is a conflict -only one of them has to be applied- that is resolved using the date criteria (the most recent one is applied). On the other side, the resolution of m_2 has to take into account 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 M is more recent than document P, what means that the modification expressed in M (m_1) gets relevant over the modification expressed in P (m_3): element m_2 (with its descendant m_{21}) replaces element d_2 in D.

- 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 any of 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 m_2 and its descendants while d_3 is replaced by n_2, thereby completing the recursion. The resulting document can be seen in figure 3.

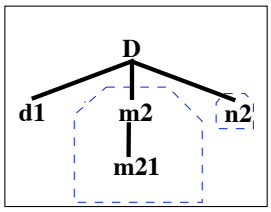

Figure 3: Output version obtained by resolving the versioning graph in figure 2.

4.4 Using XLink to model the relationships

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

1. Links must be able to address internal document fragments.
2. The link database should accept queries about linking information.
3. Links can be queried multidirectionally. A citation or modification can be exploited in both link 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. For 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.

XML has associated standards that allow relationships between documents to be modelled. Linking with XML includes rules to link resources (XLink) [19] and to address internal fragments inside linking resources (XPointer) [18]. XLink allows traditional unidirectional links to be embedded inside a document (as HTML links are), but also more complex links.
Xlinks are a means to model graph information: vertices are resources (documents, images, etc.), arcs in the graph are arcs inside an xlink, arc labels are modelled with `role` attributes and other metadata about arcs or vertices can be joined as (resources or arc) attributes.
Figures 5 and 4 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 expresses 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 of one ENLACE element, containing 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

...
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.

...

Figure 6: A fragment of modifier rule.

l13-1986.xml. The ARCO element tells about the link type: *substitution*. More semantic about the relationship can be found inside the link elements: the document dates and the document classes.

5. THE LEGISLATIVE DIGITAL LIBRARY CASE: RULES VERSIONING

Legislative documents are intensively related and rules suffer amendments that result in new versions of the amended rules. Besides, they are highly structured (their component elements can be accurately addressed). Moreover, access to all versions of a document is an important facility for their users; for example, to understand a tribunal sentence it is necessary to get access to the text of involved rules, as they were valid at the moment the sentence was made. In legislative documents, modifications are embedded inside other documents, so that the document to be modified is cited while how it should be modified is expressed later. A document can modify several other documents, and modifications to a given document can come from various sources.
We can see in figure 6 a fragment of the document where the modification expressed by link in figure 4 can be found. As it is stated by the xlink, the replacement text is the one of the first *Artículo* inside the first *DISPOSICION*. It will be copied during the versioning process as the first *Artículo* of the new version of the document *Ley Orgánica 2/1980* (`lo2/1980.xml`).

6. RELATED WORK

Document evolution has been considered from different perspectives and related to other problems. Most of the times, the perspective taken is to study its impact on the systems that suffer this evolution: maintaining the documents database [6], obtaining the revision links this evolution provokes [6, 5], representing this evolution [5], or its influence on link evolution -and validity- [15, 13]. 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 them, versions are supposed to be manually generated. But 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: storing all versions caused by a change and linking them [6], and representing changes as annotations [5]. The solution of maintaining all versions simultaneously has shown to have its main difficulty in links maintenance [6, 15]. 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 [6, 15] 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.
Finally, we are only aware of one proposal to automatically generate document versions; it was recently proposed in [2]. They keep rules to automatically generate document versions. The version generation algorithm presented in this chapter does not use rules that indicate how to generate versions, but the information it uses is the relationship information stored in links. It is the link graph traversal that allows versions to be generated, with variable parameters (such as the version date).

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 to future integration of other dynamic methods of treating relationships.
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. This means that elements involved in a modification relationship are not first-class entities (they are not documents most of the times, and they are not even files, but fragments). The second interesting quality is that it is possible to extract the relationship from a document text. Node-sets involved in a relationship can be addressed by the position of their root node in the document tree. We profit from these characteristics to allow for dynamic document generation.
The approach we have chosen is to maintain links separately from document content, taking profit from XLink capabilities. This has several benefits: the integrity of documents

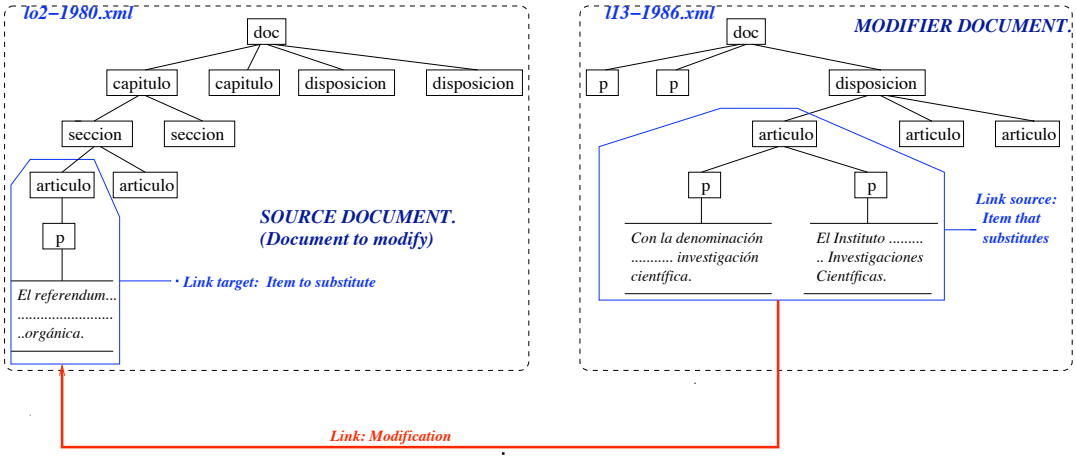

Figure 4: "Modification" link. The ORIGIN will replace the TARGET when generating a new version of the source document. The TARGET is a subtree of the source document made up of the element `articulo` and all its descendants. The ORIGIN is the subtree in the modifier document tree whose root is the first element `articulo` inside the first `disposicion`, as can be seen in figure 5.

```
<ENLACE>
<ORIGEN  xlink:href= "l13-1986.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>
```

Figure 5: Text for the example link.

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 is simplified (it can be based on a document tree, considering modifications as links that reach this tree).
Future work will consider to test the adequacy of the version generation solution in other environments where modifications do not come from document references, and to explore methods that allow for an advanced intelligent resolution of conflicts during version generation.

8. REFERENCES

[1] Abiteboul, S., Cluet, S., Christophides, V., Milo, T., Moerkotte, G., and Simeon, J. Querying documents in object databases. *International Journal on Digital Libraries 1*, 1 (1997).

[2] 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).

[3] 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).

[4] Cellary, W., and Jomier, G. *Building an object-oriented databse system. The story of O_2*. No. 19 in The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1992, ch. Consistency of Versions in Object-Oriented Databases.

[5] 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).

[6] 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.

[7] Conklin, J. Hypertext: An introduction and survey. *IEEE Computer 20*, 9 (1987), 17–41.

[8] *CVS. Concurrent Versions System.* http://www.cvshome.org/docs/manual/.

[9] 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.

[10] DeRose, S. J., Seperberg-McQueen, C., and Smith, B. Queries on Links and Hierarchies. In *Proceedings of QL'98 - The Query Languages Workshop* (Boston, december 1998).

[11] Noticias Jurídicas. http://www.noticiasjuridicas.es/.

[12] Martínez González, M. *Dynamic exploitation of relationships in digital libraries*. PhD thesis, Dpto. de Informática (U. Valladolid, Spain), Apr. 2001.

[13] 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.

[14] Parunak, H. V. D. *Hypertext/Hyermedia Handbook*. Intertext Publications, 1991, ch. Ordering the information graph, pp. 299–325.

[15] Sjöberg, C. M. DTD development for the legal domain. In *Swedis SGML 97* (1997). http://info.admin.kth.se/SGML/.

[16] Verbyla, J. Unlinking the Link. *ACM Computing Surveys 31*, 4 (Dec. 1999), 34–.

[17] Vercoustre, A.-M., and Paradis, F. Reuse of Linked Documents through Virtual Document Prescriptions. *Lecture Notes in Computer Science. Lecture Notes in Artificial Intelligence* (May 1998).

[18] W3C, the World Wide Web Consortium. *XML Pointer Language (XPointer)*, Dec. 1999. W3C Working Draft. http://www.w3.org/TR/xptr.

[19] W3C, the World Wide Web Consortium. *XML Linking Language (XLink)*, Feb. 2001. W3C Working Draft 21-February-2000. http://www.w3.org/TR/2000/WD-xlink-20000221.

9. BIOGRAPHIES

Mercedes Martínez obtained her PhD in Computing Science in 2001. Her main research interest are in the area of digital libraries. She is a reader at the University of Valladolid (Spain). She has done her PhD Thesis in collaboration with the French laboratoy LORIA (*Laboratoire Lorraine pour la Recherche en Informatique et ses Applications*). In march 2002 she starts a postdoctoral stage at the University of Esbjerg (Denmark), to work in the digital libraries and hypertext domain.

Jean-Claude Derniame is currently Professor at *Institut National Polytechnique de Lorraine* and researcher at LORIA, leading the LORIASI (*LORIA dans la Société de l'Information*) team. He has coordinated several European research projects related with software process technology and organised various congresses, workshops and summer schools. Since 1998 he developed an new activity of technology transfer and assistance near administration and public institutions for professional web sites and intranets.

Pablo de la Fuente is an Associate Professor at the University of Valladolid (Spain). He leads the "Digital Libraries and Information Retrieval" (*GRINBD*) and the "Architecture and Concurrency" (*ARCO*) research teams in this university. He has participated in the coordination of several research projects related with Spanish manuscripts and legislative digital libraries. He has organised conferences related to digital libraries and information retrieval areas.