INTRODUCTION

Digital libraries are systems that contain organized collections of objects, serving in their most basic functions as a mirror of the traditional library that contains paper documents. Most of the information contained in the collections of a digital library consists of documents, which can evolve with time. That is, a document can be modified to obtain a new document, and digital library users may want access to any of those versions. This introduces in digital libraries the problem of versioning, a problem that has also been considered in a related community, the hypertext community (hypermedia in its most extensive acception). Some domains in which document evolution is very important are the legislative domain (Arnold-Moore, 2000; Martínez-González, 2001; Vitali, 1999), the management of errata made to scientific articles (Poworotznek, 2003) and software construction (Conradi & Westfechtel, 1998).

In the legislative domain, rules suffer amendments that result in new versions of the amended rules. 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 and how it should be modified is expressed later. For each modification, its author cites the document fragment he or she wants to change and indicates how the said fragment could be modified (e.g., eliminating it, substituting it). The new version obtained by the application of these changes is virtual, in the sense that the library users know it exists but there is no physical copy of it available.

Figure 1 shows an example. The text that appears in the figure is a fragment of an EU normative document. It is possible to recognise here the 20th article of the document, that modifies another article of a different document (the ‘1968 convention’). The reference in the figure indicates in a precise manner the article (number 41) affected by the modification (substitution) that follows. The legislators leave to the readers the cut-and-paste work needed to obtain an updated version of the modified convention.

Errata to scientific articles are somehow similar. The errata are posterior to the original article and they are published together with the reference to the part of the article to be changed by the modification. How and where the errata are inserted varies among publishers and among publications from the same publisher. One way to insert errata is by listing it at the beginning or at the end of the corrected article.

Software construction is a bit different, as it is the composition of software with several of the program files considered here. The different versions of program files are available at the same time, and the composition of software has to assemble adequate versions in order to obtain the correct version of the software. There is no need to be precise about the internal parts of a document or file affected by changes; the program files used to obtain a software are not fragmented during the composition process.

Next, the issues related with document versioning are revised, the main approaches are proposed, and the issues that each approach privileges are identified. Some of them are more recent than others, but promising. Versioning a document impacts not only the document itself but also other items, as references from and to the versioned document, or the indexes created for information retrieval operation.
BACKGROUND

As for the issues of interest related to document versions, there are seven categories:

1. **What can be versioned**: This question can be considered from two perspectives. The first perspective considers objects stored in the system. This is the typical situation in the Web and hypertext environments. Hypertext nodes (e.g., documents, files) can change, but the hypertext structure can also change (e.g., objects may vary their location, some of them may disappear, others may change their references to other objects). The evolution considered in the second perspective is the one used by digital library users—these documents may or may not match unidirectionally any of the objects stored in the digital library (Arms, Blanchi, & Overly, 1997). Changes in this case affect the content of documents: the text, the internal structure of documents (e.g., this is the case for structured documents), or references (i.e., citations within documents that are part of document content). How the evolution of documents at the user level impacts the system depends on the technical solutions used in the digital library (a digital library can use hypertext models and software, textual document databases, or other types of databases).

2. **Detecting changes**: Sometimes it is necessary to recognize two documents as versions of the same work or to find the changes that have to be applied to a document version to obtain another one. There are two possible ways to do this: extracting references from document content (Martínez-González, de la Fuente, Derniame, & Pedrero, 2003; Thistlewaite, 1997), or comparing versions (Chawathe, Rajaraman, García-Molína, & Widom, 1996; Cobena, Abiteboul & Marian, 2002; Lim & Ng, 2001).

3. **Representing changes**: The information about versions and their differences has to be stored somehow in the system. There are three ways to accomplish it:
   - To store the versions caused by a change or the corresponding differences. This is the solution 1 of versioning management approaches.
   - To represent changes as annotations (attributes) to the affected nodes. This is solution 2 for version management. All the history of a node is described in these annotations.
   - To model modification relationships as links between the modifiers and the target of the modification (Martínez-González et al., 2003). This solution considers the semantic relationship that is behind a change.

4. **Querying changes or querying the history of a document**: This consists of answering questions about a document evolution, such as “What are the documents that modify this one?” or “What are the modifications of this document over the last month?” The way to operate here is dependent on the choice made for representing changes. If versions are first class objects, the only way to query changes is to search for differences between versions. It can be a file or tree (in case of structured documents) comparison. If changes are represented as annotations, to query them is to query annotations. In the case of structured documents, that means to query node attributes. If changes are represented as independent links, to query changes is to query links, which is the same as querying any document.

5. **Accessing any version**: The access to any version can be provided either by storing all of them or by composing them on demand. It depends on the approach chosen for version management.

6. **Dealing with the propagation (effects) of versioning some items on related items**: For example, changes to a document affect references or links that reach the document. This is the well-known problem of dangling links common in the Web or, in a more general definition, the referential integrity issue (Ashman, 2000). Other possible impacts are on indexes used for information retrieval operations.

7. **Inferring the composition rules of new versions**: In certain situations, such as the example in Figure 1, the new versions are virtual (no copy or direct way to obtain it is provided), and the structure of the new version has to be inferred from the information provided about modifications. Humans commonly assume this task, but an automatic inference can also be tackled (Martínez-González, de la Fuente, & Derniame, 2003).

APPROACHES TO THE MANAGEMENT OF VERSIONS

Different approaches can be distinguished. Here they are listed, ordered temporally: The ones with longer tradition appear first and the more recent ones are at the end of the list.

1. **Maintaining simultaneously all versions (or deltas) in digital library collections and linking related versions**: The main problem with this ap-