ELSEVIER

Computer Networks 34 (2000) 861–872

COMPUTER NETWORKS

www elsevier com/locate/comnet

Submission and repository management of digital libraries, using WWW

Gregory Karvounarakis [a,b,*], Sarantos Kapidakis [c]

[a] *Institute of Computer Science, FORTH, Vassilika Vouton, P O Box 1385, GR 711 10 Heraklion, Greece*
[b] *Department of Computer Science, University of Crete, GR 71409 Heraklion, Greece*
[c] *National Documentation Centre, National Hellenic Research Foundation, 48 Vas Constantinou, 11635 Athens, Greece*

Abstract

Digital library objects can be quite complex, with a lot of metadata and data in different digital forms. Currently, most installations of digital library systems need personnel experienced with computers to administer and maintain them We designed and implemented tools to simplify addition, modification and numerous other maintenance actions in a digital library, for use by content contributors (where appropriate) and librarians without special computer training. Our tools provide an easy way to input or modify the metadata describing a submission and also to upload the digital documents, as well as inspect, commit or modify new submissions (and old objects) to a set of collections, even with different metadata fields. The tools are easy to install, while at the same time they are powerful and configurable. The tools are modular and generic, and cover even rare requirements. The configuration choices allow the customization of the tools for use with different digital library systems using different metadata formats, fields, languages and many optional features, depending on the level of sophistication required for a particular environment.

Keywords Digital libraries; Metadata submission; Repository management; Library administration

1. Introduction

Digital libraries hold material online in a digital form and provide advanced ways (compared to traditional libraries) to access, search and retrieve this material. They support distributed collections of digital data all over the world. Users of such libraries can recall from their computer the data they are interested in, and study digital copies of it without ever having to visit the building of a library. A digital library can contain text, picture, sound, video, and in general, objects in many different formats. Moreover, compared to the usual Internet search tools, digital libraries maintain this huge amount of information under consistent classification schemes, thus helping the users to easily locate the information they require.

A digital library consists of a distributed set of servers, handling the corresponding repositories. Digital library clients may range from specialized client applications to common web browsers; these can be used to forward operations to the digital libraries, as querying or browsing. Queries are performed on a group of objects that form a database or repository – the "physical" storage of the

* Corresponding author
E-mail addresses gregkar@ics forth gr, gregkar@csd uch gr (G Karvounarakis), sarantos@ekt gr (S Kapidakis)

1389-1286/00/$ - see front matter
PII: S1389-1286(00)00157-2

digital library. This repository may be separated into collections, which contain objects with common attributes.

A query in a group of collections is based on the structure of the repository and the objects it contains. Any such object should be "registered" in the digital library server, so that the server knows about its existence and can access it when necessary. Every registered object is available in some digital formats and has some *metadata* associated with it. Metadata (or "data for data") is information about the data (e.g., title, creator, summary), and is stored in a way that is directly associated with the corresponding object. Without such information, it is not possible to query or retrieve the digital formats of the object. Based on this metadata, digital library systems can query the actual data, through indices or other more sophisticated storage facilities. When a query is performed, qualifying objects are reported to the user, and their digital formats become available.

The openness and usefulness of digital libraries and the huge amount of information stored in a digital library intensify the need for contributors to directly add content and tools to simplify the procedure of collecting and managing new objects [4,5]. Moreover, the existing tools are very primitive. Furthermore, there is a need for tools to simplify the administration of the repository of a digital library. Such tools should be as user-friendly and usable as possible to allow contributors to register their work in digital libraries, and librarians to administer and maintain a digital library, even with minimal experience and knowledge of computers.

In addition, digital library systems may handle multilingual objects as well as collections with different metadata fields. Such features have a serious impact on the form of the metadata descriptions that should be created, as they complicate the creation, submission and maintenance of a digital library object even further, therefore making the creation of tools for these functions even more appealing.

This paper describes our work on these issues. During the design and implementation phase, our main aims were to create an easy to use and learn user-interface and to provide all the functionality needed. Moreover, we tried to make these tools as easy to install as possible and, at the same time, as configurable as possible. We tried to write reusable code and to create a set of tools, which can be modified and hopefully used with any digital library system, without extensive effort. In any case, we hope that this work will help as a guideline for similar efforts in the future.

In Section 2 we present an overview of the submission tools, and afterwards we describe the functionality of the tools.

2. Overview of the submission tools

The operations that a set of submission tools for a digital library need to support, can be divided into three functions: submission of a new digital library object, modification of an older submission (metadata modification or submission of additional digital formats to a submitted library object), and the process of approving and committing submitted metadata and digital formats.

A general analysis of the sequence of actions and forms, which are displayed for each of these functions, is shown in Fig. 1. This diagram shows the first two grouped functions as concentric ovals. The entry points to the tools, for each one of them, are displayed as rectangles with rounded corners. The rest of the forms are displayed as rectangles with the arrows between them showing the sequence of the forms during these operations. Finally, especially when there are several alternatives, the bubbles on each arrow illustrate the case or event, which leads to the transition from one form to another.

The operation of submitting a new object to a digital library begins with the submission of the metadata describing that object. Afterwards, the user should submit at least one digital document (through the "file upload form"). After uploading each format, a confirmation message is displayed – if uploading was successful – and this process can be repeated, as long as there are more formats to submit. When all formats have been submitted, a form with an overview of the submission is displayed, describing the submitted digital formats.

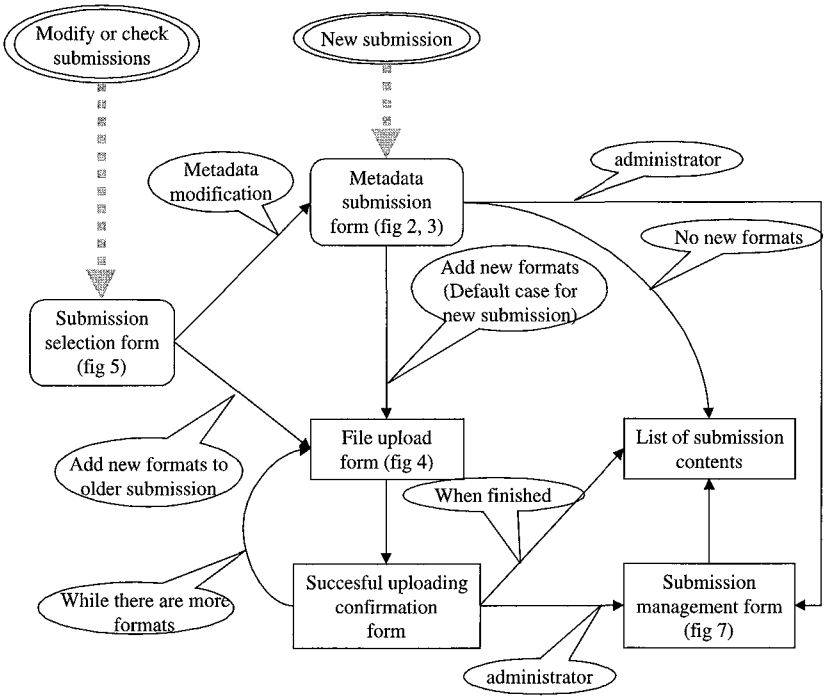

Fig 1 General sequence of forms of the submission tool

In the case of modification or checking, one has to first select which library object should be modified. If they want to add some new formats, without modifying the metadata, then the "file upload form" is used and the sequence is continued as before. Otherwise, a form with the current contents of the metadata file is displayed. After submitting the data of this form, the user can either choose to upload new or modified digital formats, if any, or proceed directly to the final form.

All incoming data are placed in a temporary repository following the same structure as that of the permanent repository. Unauthorized contributors can access and modify objects only from this temporary repository. When someone tries to modify an object in the permanent repository and there is no entry in the temporary repository for it, a temporary entry is created to store the modified information, based on the original entry in the permanent repository. Using the submission management form, the librarian-administrator can later inspect and discard or approve and commit all or part of a new or modified submission to the permanent repository.

These tools are largely configurable (including paths, languages, fields per collection and even messages, colors etc.), although their minimal configuration is very simple, since most of the parameters take default values from their environment and, in most cases, do not need to be modified.

For user-friendliness, we use the tools through a web browser interface. Moreover, HTML forms, which are used as the main tool for entering information, are simple enough and close to the novice user's concept of an interface for entering information. In our implementation, we used the PERL programming language and the common gateway interface (CGI [3]), as CGI-scripts are most suitable to use in co-operation with HTML forms.

The functions of submitting a library object to a digital library and managing the objects that form

the library can be subdivided into three distinct, but complementary, categories: metadata manipulation, data handling and repository management. In the next sections, we will analyze and explain each of the above separately, in order to clarify the issues that were considered during the design phase and the solutions at each stage. We also discuss the way these tools are meant to be used to fulfill the initial aims and consider the level to which they meet our initial usability requirements.

3. Metadata manipulation (submission – modification)

Metadata is a basic part of any submission, because all functions on the digital objects are normally based on it. Consequently, a library object should include the metadata describing it, in order to be used by the digital library server. For example, metadata could conform to the protocol described in RFC 1807 [7], which proposes a generic format for organizing metadata. This format is as follows:

FIELD NAME :: field value

which in the case of bibliographic data is applied more specifically as:

```
CS-TR-version :: value
ID :: collection_id//registration id
FIELD_NAME_1 :: field_value_1
FIELD_NAME_2 :: field_value_2
...
END :: collection_id//registration id
```

This format may look simple enough to understand, but could still be quite difficult for users with minimal computer knowledge, as in fact are the majority of the people that are expected to contribute content to a library. Moreover, it can often be frustrating for administrator-librarians to maintain a digital library by manually editing such configuration files. Furthermore, as a digital library provides www access, the submission of metadata should be specified in a similar manner. Thus, we use the following form shown in Fig. 2.

Several conventions and restrictions can be applied for specific fields. For example, RFC 1807 supports multivalued fields in the form of repeated attribute-value pairs. Thus, if an object has two or more *creators*, multiple entries for this field should be created in the metadata file. In this implementation, such fields can be configured to be filled in as comma-separated (or generally special character separated) lists. We can enter, for example, the value "Greg Karvounarakis, Sarantos Kapidakis" as a value for the field *creator*, if it has been configured to be comma-separated. All these internal transcriptions are transparent to the user who submits an object to the library, requiring only the completion of a form with the appropriate fields for each collection and denoting the languages supported. Moreover, as the code (that depends on the specific protocol in which metadata is stored) forms a distinct module, it can easily be modified to manipulate different metadata interchange formats for different systems.

Fields whose value should have a specific format also complicate the creation of the metadata for a digital library object is that, for example, a *Date* field should be in the format "Month Day, Year". The submission form performs such validation checking, while manual metadata entry does not.

Some fields can, and should be, automatically filled (e.g., "submission date") for others, default values could be proposed, especially if the fields have special significance. For example, the field ID denotes the collection in which the object is going to be placed as well as its identification (as a library object). While the user should choose the collection name, from a list of the existing collections, it is preferable that the object ID is automatically generated, to ensure uniqueness. Thus, we create an ID based on the date of the submission, a serial number and a random code – for security reasons. For instance, for submission number 135 committed on 17 August 1998 we would get something like 1998_09_17-135-26721 (the length of the code is configurable). In the repository management tools, all fields that the

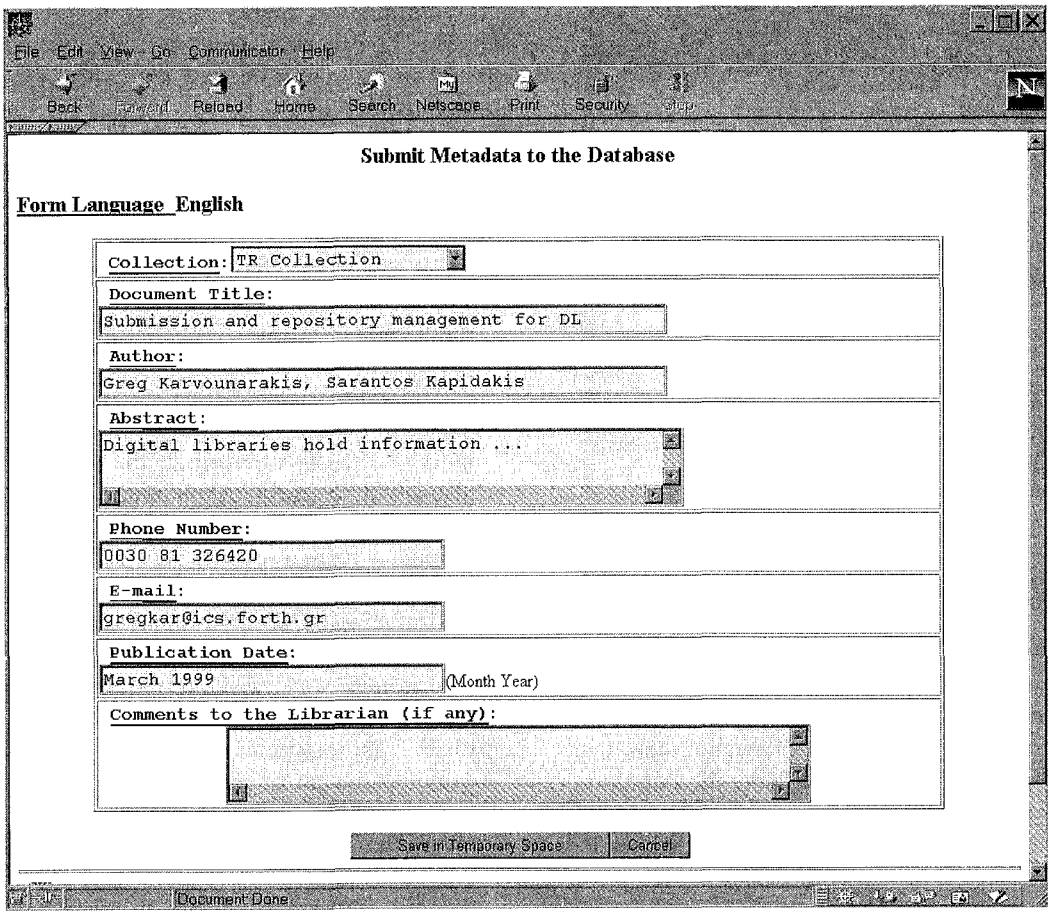

Fig 2 Single language metadata submission form

administrator is allowed to modify are also automatically filled, to allow flexibility in the manipulation of the digital library objects.

Another kind of validation check applies when we declare some fields as mandatory: the metadata is not accepted until the user has filled in a value for all those fields (for at least one of the languages supported by the system).

In the modification of a submission, if the metadata is not found in the temporary repository, it is first copied from the permanent repository, and the metadata in the temporary repository is modified. This way all modifications are performed in the temporary repository and nothing is placed in the digital library repository, before the administrator approves it.

3.1. Multilinguality

Another aspect that should be considered is the extensions of digital libraries in order to manage multilingual objects and handle heterogeneous collections, which require different and more complicated metadata. We may have different metadata fields per collection, and categories of restrictions (e.g., comma-separated, mandatory fields) for some validation checking. The metadata forms and validation checking functions are created dynamically, based on the configuration. Moreover, we provide a multilingual user-interface and offer two facilities for handling multilingual metadata, which suffice in most cases:

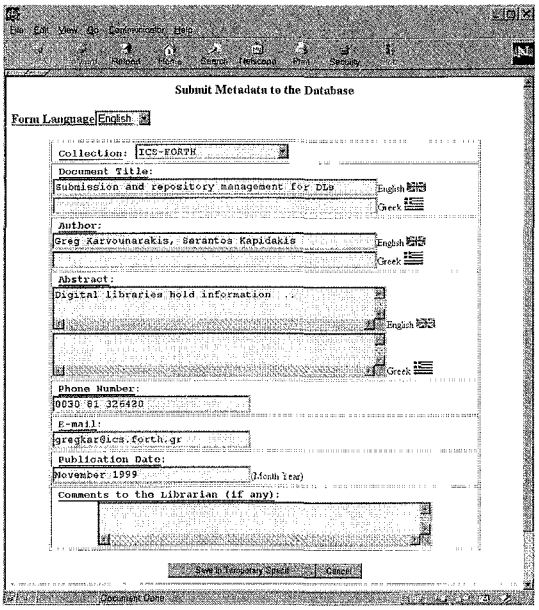

Fig 3 Multilingual metadata submission form

- *Different fields per language.* We use different fields for the translation of each initial field and the digital library uses them all. For example, for the field "Title", which by default should be in English, we can define a field called "Title_Greek". Then, both fields would appear in the metadata submission form. The tools may not even know that the system supports multilinguality.
- *Separated list of values.* We encode the different translations of the metadata field in one value, separated by a special character. Then, if we want to enter, for example, the title in both English and Greek, the corresponding line in the metadata file would look like:

  `TITLE : English_Title#Greek_Title,`

  where # is the special character separating the translations. The form shown in Fig. 3 is used for the submission of such metadata.

4. Data handling

A digital library object consists of metadata describing the object and several digital formats representing the object. Several issues arise regarding the way these formats are uploaded to the library as well as to the way they are organized in the repository of the library so that the digital library system can access and retrieve them. As the aim of the creation of this set of tools was to provide an easy means of submitting an object to a digital library, these issues should be hidden from the user and we provide such functionality in that:

- The contributor does not need to know the digital formats that the digital library supports. Instead, a list of the supported formats should be provided in the submitting interface.
- There is a simple way to upload the digital formats of the object through the www. There is also a way to send more than one digital format, add newer formats some time later or change an uploaded format with a new or corrected version.
- The submission process manages both metadata and digital formats, and stores them in a *self-explanatory* way, without requiring any other contact with the administrator in order to complete a submission. This means that the tools provide a means of submitting bibliographic data and digital formats, as well as storing them in a way that requires the minimum possible effort from the administrator, to inspect, approve and commit each submission to the permanent repository.

In order to provide an intuitive tool to upload digital formats, we use the protocol described in RFC 1867 [8], which is supported by HTML 3.0 and later and by the most commonly used web browsers – that is Netscape Navigator 3.0 and later and Internet Explorer 4.0, with which the tools have been tested. [1]

This protocol allows the uploading of files through web-servers, specifying a special HTML form field for this purpose. This way one can upload a file of any type by just entering its name as

[1] For Internet Explorer, a plug-in must be used but this plug-in is usually preinstalled, although the use of Internet Explorer is not suggested, as it is not totally compatible with the standard specification of JavaScript, which is extensively used in these tools

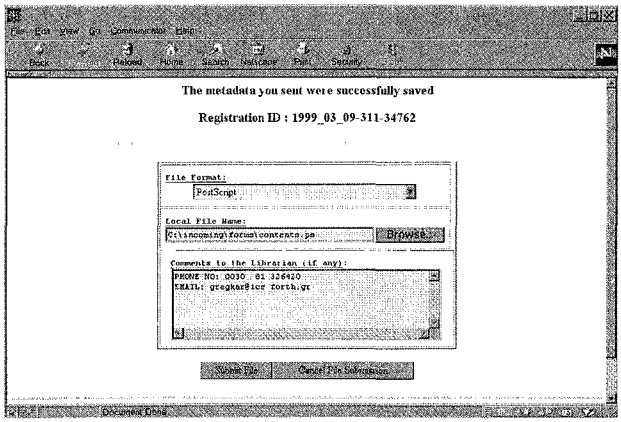

Fig 4 Selection and uploading of digital documents

the value of that field, as shown in Fig. 4. This form also contains a list of all acceptable digital formats, created from the list of supported formats that is specified in the configuration of the tools.

The message at the top of the form is displayed on a new submission, confirming the successful storage of the metadata and providing the registration ID of the submission. Digital formats are stored as a part of the entry for each submission and described by the ID of that submission. In this way, the connection between the metadata and the digital formats of the object it describes is preserved. Moreover, the naming of the files stored in the repository depends on the ID of the submission and on the type selected from the list of acceptable types. This way, there can only be one instance of a specific format of a digital document.

According to the way they should be handled, the acceptable digital formats have been divided into the following categories:

- Formats that consist of one file, for example, PostScript, PDF, Word documents etc.
- Formats that consist of many files, for example, a structured HTML document, which may include several HTML files, images etc.
- Formats that consist of many files, which are also ordered in some human-notion. An example of such a format is a scanned copy of a book, where each page could be an individual image file.

4 1. Simple formats

These formats are just uploaded through the form. When trying to save the uploaded file, if a file of the same type (possibly compressed – using gzip or zip) is found for this submission, an overwrite-question is displayed, so that only one version exists for a specific format.

Apart from the formats that can be selected from the list of supported formats, we also allow the submission of currently unsupported formats, through the selection "Other formats". This way, an administrator can allow, for example, the submission of LaTeX.

4 2 Multifile formats

The most interesting issue applies in digital formats that are physically formed by a set of files. Protocol security reasons prohibit uploading multiple files or a whole directory. Therefore, the procedure described above is inadequate to upload a digital format that consists of many files, as all files should be sequentially specified. For this reason, we accept such formats in the form of an archive (possibly compressed), which includes all the files that form the digital object.

A special such format is HTML, where objects should be uploaded in the form of an archive including all the necessary files and, among them the entry page, a file named *index.html*, and we check

its existence in the archive, in order to accept HTML format.

4.3 Numbered multifile formats

This is a special case of multifile formats in which, apart from the validity of each file separately, one should also check the ability to put these files in a logical order, in order to display them correctly. We designed and applied an algorithm, in order to specify some types of naming of the files that form the digital format, which seem to have an "obvious", unambiguous order.

After the archive has been uploaded, we have to find the order of the files and rename them to a unique and easily parsable scheme, such as:

```
directory_name/0000.suffix
directory_name/0001.suffix
directory_name/ .
```

where `suffix` is the proper suffix for each type of files (e.g., `tif` for scanned images).

This ordering is unambiguous and simple, but, we do not restrict creators to such a strict ordering, as, for instance, one may prefer to order a book in a more structured way, using chapters etc., or partial ordering in terms of a chapter. Moreover, the creator should not have to name their files as 0000, 0001 etc., but be allowed to name them file1, or page1 etc. Thus, we accept a less strict set of ordered names. According to this, the first number of every file name denotes the relative order and should be unique, so that missing numbers in the sequence can be detected. Names like chapter1-page5 would probably lead to the rejection of the submission, since there should probably exist another file with number 1. According to this, we accept two possible formats:

- *Plain format.* We use the list of all file names from the current directory and all of its sub-directories, recursively. The names of the files, ignoring the directory component, should contain numbers and be able to be ordered. Moreover, any number should appear in no more than one file. Furthermore, all files from this list should follow global numbering, regardless of the directory in which they are. In this case, we have global numbering. This numbering should be consecutive and start either from zero, one or from the number that the last global numbering stopped (in chapter format with each chapter separately conforming to the plain format). The simplest case of plain format has global numbering starting from 0, with all files in the same directory.
- *Chapter format.* The current directory should only contain chapters (sub-directories), not simple files. The names of these chapters should be serializable and missing chapters are checked. Each chapter (sub-directory) should be organized with the plain format or the chapter format, recursively, with either continuous numbering or numbering starting from the beginning.
- In some cases, we accept a mixed format, when the human-notion of the ordering is unambiguous.

Some examples should make these formats clearer (in most examples, for simplicity, we use plain numbers instead of longer file names):

1. Global numbering, plain format.

   ```
   dir/xxx/file1
   dir/xxx/foo3
   dir/yyy/test2
   dir/zzz/page4
   ```

2. Chapter format in `dir` and `ch2`, while we have plain format and partial numbering in `ch1`, `ch2/sec1`, `ch2/sec2`, `ch3`.

   ```
   dir/ch1/1
   dir/ch1/2
   dir/ch2/sec1/1
   dir/ch2/sec1/2
   dir/ch2/sec1/3
   dir/ch2/sec2/1
   dir/ch2/sec2/2
   dir/ch3/1
   ```

3. Chapter format in `dir` and plain format with partial numbering in each of the chapters. In `ch2`, we cannot order the sub-directories in

chapter format, but we can order the files separately in plain format.

```
dir/ch1/1
dir/ch1/2
dir/ch2/xxx/1
dir/ch2/xxx/3
dir/ch2/yyy/2
dir/ch3/1
```

4. The directory `dir` follows plain format with global numbering.

```
dir/ch1/1
dir/ch1/3
dir/ch2/2
dir/ch2/4
dir/xxx/yyy/zzz/5
dir/6
```

If the structure of the object is found valid and unambiguous, then the files are globally ordered and restructured.

5. Repository management (submission – modification)

A great deal of this administrative work overlaps with the processes of submission and modification. However, there are some more actions that an administrator should be able to perform. Moreover, the administrative tools for submission and modification of digital library objects should provide greater control and functionality. Keeping these aims in mind, we can distinguish the following actions, which are required for the administration of a digital library:

- Browsing, to see and select a submission from the temporary repository, as in Fig. 5.
- Inspecting and approving a submission: The main task of an administrator-librarian is to inspect contributed objects to see if they should be approved and placed in the digital library. This action actually consists of two parts. First, the administrator has to check – and possibly modify or complete – the metadata describing the object, for completeness and validity. This is

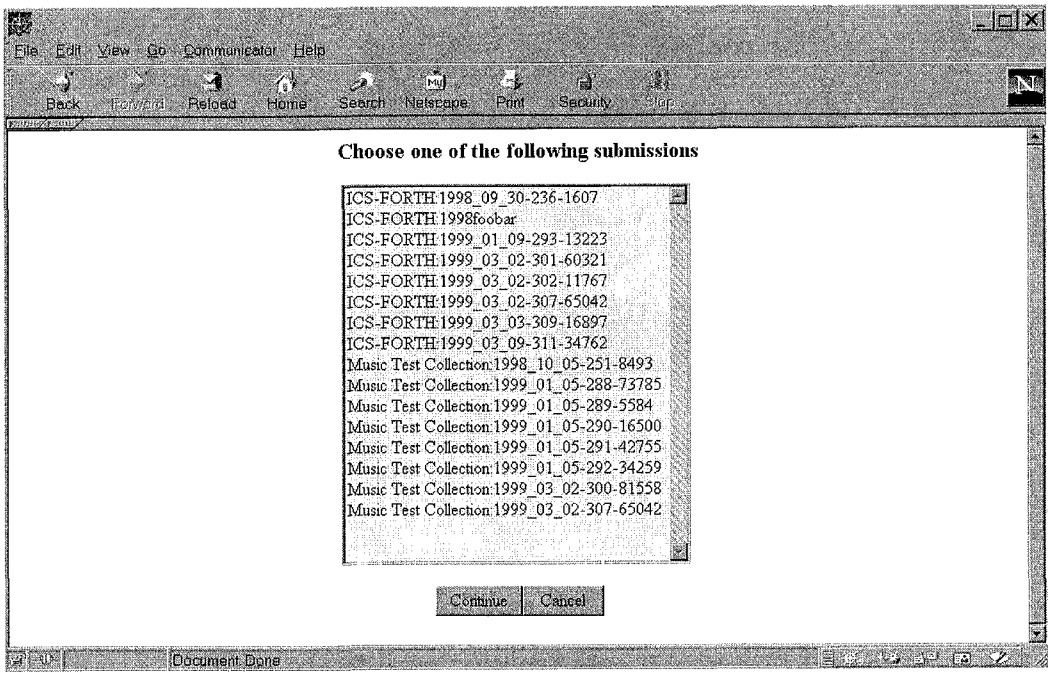

Fig 5 Temporary repository browsing form

similar to the function of the modification of the metadata of a submission, and therefore a form like the one in Fig. 6 is used. Next, the administrator should check what digital formats have been submitted for this object and, if possible, whether they are valid. In the end, the administrator should either approve the submission, and commit it to the digital library, either as a whole or a part of it, or reject it by erasing it from the temporary storage space, or maintain it in the temporary space to be modified or inspected at a later time, as in Fig. 7.

- New submission: This action consists of the same tasks as that for an object contributor. However, the administrator is allowed to intervene in the creation of the submission. For example, an administrator should be able to modify the automatically generated object ID.

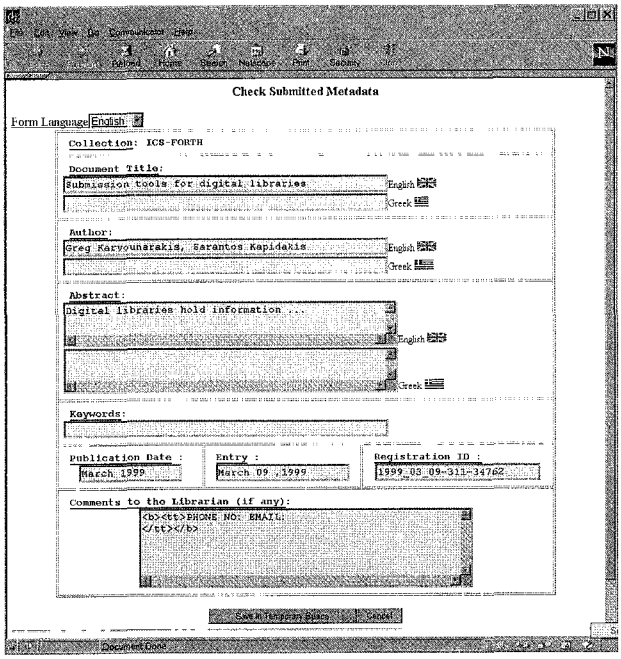

Fig 6 Cross-check of a submission by the collection administrator

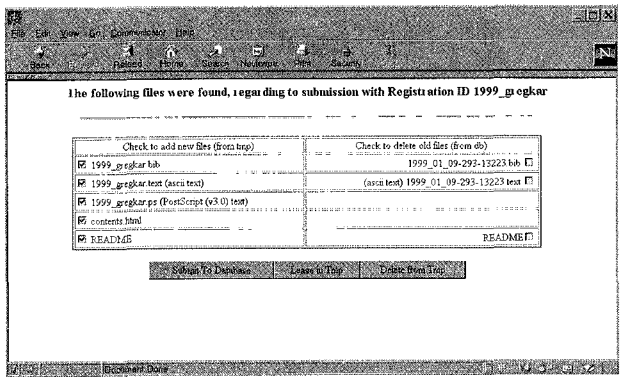

Fig 7 Update of the permanent repository by the administrator.

The administrator can also configure some other fields that are only editable by himself. After the metadata has been stored and the digital formats have been uploaded, the administrator is able to immediately commit the new submission to the permanent repository, as in Fig. 7. This way, inspection and approval can be performed in one stage.

- Modification of permanent digital library objects: If an error is found in a permanent library object, or if the creator of such an object provides a newer version for it, the administrator should be able to modify a permanent object. This action is actually the same as the modification of a submission that appears in the temporary space. For reasons of consistency, this modification also results in the creation of a modified copy of the permanent object in the temporary space. When all changes are over, the administrator can choose those parts from the older version that should be kept, and those that should be replaced by the modified ones, in an atomic action, as seen in Fig. 7.

For most of these functions, the forms of the contributor interface can be used. The format uploading form can be used as it is; the metadata form looks the same as the contributor form, with the addition of possible "administrator" fields, for example, "keywords", and the ability to modify fields such as registration ID or submission date, which are not shown in the creator interface for simplicity and consistency. Therefore, the only additional functionality that should be offered by the administration tools is the ability to manage digital objects, commit them to the permanent repository or remove them, as well as the ability to merge existing objects with submitted modifications and additional formats. A main administrator page leads to these actions.

Finally, we provide utilities to check the validity of the digital formats, through:

- An estimation of the type of each file by invoking the Unix command "file".
- The ability to launch the digital formats through the web browser, using HTML anchors to the repository.

It should be obvious that major security issues emerge, with the introduction of such web-based administrative tools, since they can be used to modify or delete objects both from the temporary and permanent digital library repositories. To cope with this problem, we chose to apply access control to these tools, by using the access control options offered by the web-server [2,9]. Since the whole application is based on server scripts this is a simple way to ensure that only the persons that should use these tools will be allowed to access them.

6. Conclusions

These tools were created to simplify the process of submission and repository management of a digital library. In this context, they have been extensively used with several digital library systems, including the dissertation digital library of the University of Crete [2] and the SKEPSIS system, [3] a digital library holding educational material (such as lecture notes) of lessons taught in Greek universities. These systems use a distributed indexing and retrieval system called DIENST [6], which uses the RFC 1807 [7] metadata format. In all cases, our tools allowed users with minimal computer experience, such as librarians, to easily administer large collections. Moreover, the use of such tools encouraged more, non-expert, users to submit their work in digital libraries; especially in the case of SKEPSIS, professors from various university departments, many with only elementary computer knowledge, were able to submit their work without additional support.

Due to their modular design, our tools were easily adapted to handle more specialized cases. For example, the metadata manipulation code, which forms a distinct module, was easily modified to serve the needs of the Ph.D. Thesis collection of the National Documentation Centre. [4] This collection required the creation of MARC [10] records or XML [1] semi-structured documents in a temporary repository, while storing the permanent objects in a file-system or a database. Further-

[2] http://dlib.libh.uoc.gr
[3] http://skepsis.odl.uoc.gr
[4] http://www.ndc.gr

more, our tools can be abstractly viewed as a simple, one file, graphical ftp-client, that can be used for accumulating any kind of material on the www, they also guide users by providing specific fields or requiring specific file types and are more user-friendly than common, command-line ftp-clients. Generally, they can be used for managing any kind of collection of accumulated material from the www as, for example, for collecting necessary metadata information (Author names, Contact-info, Abstract) and managing the submissions for a conference, requiring minimal effort and computer knowledge and allowing modification-updating of submissions by authors.

References

[1] T Bray, J Paoli, C M Sperberg-McQueen, XML: eXtensible Markup Language 1.0, W3C Recommendation, February 1998, available at: http://www w3 org/TR/REC-xml

[2] CERN httpd: http://www w3 org/Daemon

[3] Common Gateway Interface (CGI): http://hoohoo ncsa uiuc edu/cgi/ or http://www w3 org/CGI

[4] S Kapidakis, Issues in the development and operation of a digital library, Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries, Paris, 22–24 September 1999, pp 363–382

[5] S Kapidakis, G Karvounarakis, Submission and repository management tools for digital libraries, with www interface: a demo proposal, Demonstration at the Third European Conference on Research and Advanced Technology for Digital Libraries, Paris, 22–24 September 1999, p 495

[6] C Lagoze, J. Davis, DIENST: an architecture for distributed documents libraries, Communications of the ACM 38 (4) (1995) 47

[7] R Lasher, D Cohen, RFC 1807: A Format for Bibliographic Records, June 1995

[8] E Nebel, L Masinter, RFC 1867, Form-based File Upload in HTML, Xerox Corp, November 1995, available at http://rfc fh-koeln de/rfc/html/rfc1867 html

[9] NCSA httpd manual: http://hoohoo ncsa uiuc edu/docs/Overview html or NCSA Apache httpd manual: http://www apache org/docs

[10] The Library of Congress: Standards, MARC 21, Specifications for Record Structure, Character Sets, and Exchange Media, available at http://lcweb loc gov/marc/specifications/

Gregory Karvounarakis is an M Sc student at the Computer Science Department of the University of Crete He received his Diploma in Computer Science in 1998 from the same department He has also been working as an undergraduate and postgraduate scholar in the Parallel and Distributed Systems Group and in the Information Systems and Software Technology Group, at the Institute of Computer Science, Foundation of Research and Technology, Hellas, since 1996 His current research interests are web-based information systems, metadata repositories, query languages, digital libraries, parallel and distributed systems and programming languages

Sarantos Kapidakis is head of Information Systems at the National Documentation Centre, part of the National Hellenic Research Foundation, in Greece, coordinating Digital Libraries Activities. He received a Ph D degree in Computer Science from Princeton University in 1990 He also holds an M Sc. from Princeton University and a Diploma in Electrical Engineering from the National Technical University of Athens His current research interests include algorithms, programming languages, operating systems, digital libraries, electronic commerce and parallel and distributed systems.