Supporting retrieval by "relation among documents" in the ERCIM Technical Reference Digital Library

Stefania BIAGIONI, Carlo CARLESI, Donatella CASTELLI

IEI, Consiglio Nazionale delle Ricerche,
Via S. Maria, 46 - 56126 Pisa, Italy
Telephone: +39 50 593111. Fax: +39 50 554342
E-mail {biagioni,carlesi,castelli}@iei.pi.cnr.it

Abstract: This paper illustrates the results of an investigation that aimed at clarifying to what extent the system that is used for implementing the ERCIM Technical Reference Digital Library can support the new metadata "relation with other documents". Particular solutions for this extension are discussed.

1. Introduction

In a very next future, the technical reports(TR) of all the ERCIM[1] member research centres and labs will be available through Web[4]. The aim of this service is to assist ERCIM scientists to make their research result immediately available world-wide and provide them with appropriate on-line facilities to access the technical documentation of others working in the same field. This service relies on a database of TR managed by a set of interoperating servers, based on the Dienst architecture[3,7]. Dienst provides services to store both TR and their metadata, and to retrieve and access them. The retrieval is based on the registered metadata. The configuration of the Dienst system that is used for handling the ERCIM TR database accepts a particular set of metadata. Dienst, however, can be configured to handle also other additional metadata.

At IEI-CNR, in parallel to carry out the start up of the above service, we are designing an extension of the current configuration of the Dienst system that aims at enlarging the set of handled metadata elements. In particular, this extention should comprise the "relation" element as defined by the Dublin Core(DC) metadescription standard[1]. This paper reports the results of an investigation that has been done in order to understand to what extent the Dienst system could support this new element. In particular, the differences between relation and the other metadata elements are outlined. These differences impose new requirements on the handling system. Suggestions on how this requirements could be fulfilled are given.

The presentation is organised as follows: Section 2 introduces briefly the Dienst system and the present version of the ERCIM Technical Reference Digital Library system; Section 3 highlights the peculiaties of the relation element with respect to the other currently supported metadata. Section 4 makes proposals on how this element could be handled within the current version of the Dienst system. Finally, Section 5 contains conclusions.

[1] European Research Consortium for Informatics and Matematics.

2. The ERCIM Technical Reference Digital Library

The ERCIM Technical Reference Digital Library(ETRDL) was developed by the DELOS Working Group[4] [2]. The aim of DELOS is to design, implement and test a prototype infrastructure for networked access to a distributed multi-format collection of technical documents contained in the research libraries of the ERCIM institutes. So far, pilot server sites have been set up at nearly half of the ERCIM national labs. Servers are expected to be installed at the other centres in the near future. Public access to this reference service will be provided through Internet by a common interface for end user (both information provider and seekers).

The ERCIM collection is managed by a set of interoperating servers, based on the Dienst system[3]. Dienst is a protocol and a reference implementation that provides access to distributed, decentralised, multi-format document collections over the World-Wide Web. It was originally developed in 1992-95 for the ARPA-funded Computer Science Technical Reports project in the USA, and currently forms the technological basis of the Networked Computer Science Technical Reference Library (NCSTRL). The protocol of the Dienst system, however, in no way depends on, or is specialised for, this particular collection.

The Dienst system has three logical components:

1. the *document database*: a structured collection of digital documents. Each document in the database is, at least, represented by a bibliographic file (containing meta-information about the document) and may also contain representations of document in multiple formats as Postscript, TIFF, GIF, etc..
2. the *Dienst server*: a standalone process that provides submit, searching and access to the local database; each server can also interoperate with other Dienst servers, providing search and access to documents over the distributed environments.
3. the *World Wide Web server*: the Dienst protocol is embedded in HTTP, the WWW protocol. The two protocols communicate via Common Gateway Interface (CGI).

The above three logical components are implemented by an architecture which consists of four modules. Each of these modules provides specific digital library services:

- *Repository*: it provides services for deposit, storage, access and deliver digital documents, each of which has a unique name and may exist in several different formats;
- *Index*: it supplies the mechanisms for indexing meta-information on documents and the search engines over these indexes which return a list of documents that match the search;
- *Meta*: it furnishes services (also called “contact services”) that provide information for interoperability among servers.
- *User interface*: it implements the service that provide a human front-end to the three other services.

[2] DELOS is an ESPRIT funded Working Group that comprises the followingERCIM institutions:CNR, CWI, GMD, INRIA, INESC, SICS, SZTAKI, and FORTH.

Multiple Dienst servers interoperate to provide a logically uniform collection to the user, even though the physical collection is distributed. Each individual Dienst server "knows" about other sites, and visa-versa, by periodically polling a central meta-server site.

3. Extending the metadata set

The current version of ETRDL is based on the following metadata: *Author*, *Title*, *Subject*, *Description*, *Resource Type*, *Publisher*, *Date.Creation_of_intellectual_content*, *Format*, *Resource Identifier* and *Language*. This set of metadata is close to that suggested by Dublin Core metadescription standard. Actually, only few elements are missing. One of these is "DC.relation". This element is frequently used in the bibliographic area since it permits to specify relations between documents that cannot be derived from the other metadata. At IEI-CNR, as part of the DELOS activity, it was decided to explore which would be the effects of introducing relation as an additional metadata element in ETRDL. What follows describes our experience.

3.1 The relation element

The element relation is, perhaps, the less understood of the fifteen elements proposed by the DC standard. At the present its use is considered as experimental. In the "DC User Guide Draft" this element is defined as follows[2]:

The relationship of this resource to other resources. The intent of this element is to provide a mean to express relationships among resources that have formal relationships to others, but exists as discrete resources themselves. For example, images in a document, chapters in a book, or items in a collection.

A list of types which accommodates most expected relationships has been identified as part of the standard[2]. These are: *IsPartOf*, *HasPart*, *IsVersionOf*, *HasVersion*, *IsFormatOf*, *HasFormat*, *References*, *IsReferencedBy*, *IsBasedOn*, *IsBasisFor*, *Requires*, *IsRequiredBy*.

We decided to consider all of these relation types in our tentative extension since each of them is meaningful for TR. Moreover, as we were not able to envision if other types could also be useful, we decided to design extensions to ETRDL that were parametric with respect to the relation types. This would allow to add novel relation types dynamically as soon as they become relevant.

An accurate analysis of the relation element highlighted that it exhibits distinguishing peculiarities with respect to the other metadata. In particular the followings:

- *Semantics*. The problem of assign a uniform meaning to a relation type is harder than for the other metadata elements. This problem originates especially for the relation types that are added dynamically. However, our experience has outlined that also the interpretation of the standard types can raise difficulties. While carrying out the analysis of the relation types listed above, we found that, for example, *IsVersionOf* was interpreted differently from different people. The definition of *IsVersionOf* given by the standard is: "Version relations are those in which a resource is an historical state or edition of another resource by the same creator"[2]. After having read this definition, some people concluded that *IsVersionOf* is a symmetric relation i.e., if a document x is a version of a document y then also y is a version of x. Others, instead, concluded that *IsVersionOf* is a an order relation among documents. A document dated later could be a version of a document produced

earlier, but not viceversa. Other people was unable to assign a meaning to this definition since they do not understand the meaning of the term "historical state". Finally, others found the definition inappropriate since, according their understanding of the informal meaning of *IsVersionOf*, the related document should not have necessarily the same creator.

- *Dependency between relation and the other metadata elements*. Currently, there is no dependency among the element values in an ETRDL metatada record and among metadata records. These two properties do not hold anymore if the relation element is introduced. Consider, for example, *IsVersionOf*. Its definition states that if *x* is a version of *y* then the two technical reports must have the same value for the *Author* element. Moreover, if the above relation holds, then also the inverse relation *HasVersion* between y and *x* must hold. This dependency among metadata records imposes that the handling system be able to ensure the consistency among the metadata records.
- *Relations can be dynamically added and modified*. A relation between TR is not necessarily stated when the metadata record is created. Often it is stated later. For example, when a version y of a TR x is created, a new link *HasVersion* has to be added in the metadata record of x. If, after a while, y is removed, the added link has to be cancelled.
- *Queries have a different structure*. A standard query to the ETRDL collection is made by specifying a value for one or more selected metadata as in, for example, "getTR with Carlesi=*Author*". All the TR which have that value for the given metadata are expected as a result. A query by relation has a different structure. For example, a typical query by relation is "getTR that are *IsVersionOf* TR1". In this case, a TR in the domain of the relation is specified and all the related TR are expected as result.

The above peculiarities impose a specific treatment for the relation element. The following section discusses this point.

4. Handling the relation element

This section discusses how relation can be supported within the current version of Dienst. First, a specification for the semantics is proposed. Then a possible solution to the requirements that are imposed by the introduction of relation is presented.

4.1 The semantics of the relation element

In order to achieve an acceptable level of interoperability, the intended meaning of the relational types has to be specified precisely. As an attempt to solve this problem, we associated a set of first order formulas with each type. These formulas rely upon the basic predicates that are derivable from the RDF semantics of the DC metadata element set[5,6]. They express the conditions, derivable from the informal semantics, that hold among the metadata of the related objects. For example, the semantics of *IsVersionOf* was expressed as:

- $\forall x \neg IsVersionOf(x,x)$
- $\forall x,y\ IsVersionOf(x,y) \Rightarrow \neg\ IsVersionOf(y,x)$
- $\forall x,y,z\ IsVersionOf(x,y) \wedge IsVersionOf(y,z) \Rightarrow IsVersionOf(x,z)$
- $\forall x,y\ IsVersionOf(x,y) \Rightarrow HasVersion(y,x)$
- $\forall x,y\ IsVersionOf(x,y) \Rightarrow Author(x) = Author(y)$
- $\forall x,y\ IsVersionOf(x,y) \Rightarrow Subject(x) \cap Subject(y) \neq \varnothing$
- $\forall x,y\ IsVersionOf(x,y) \Rightarrow Date.Creation\_of\_intellectual\_content(x) \geq Date.Creation\_of\_intellectual\_content(y)$

A similar semantic was given for each of the other relation types.

We also assumed that a semantic like the one given above could be specified each time a new dynamically added type is introduced for the first time. For example, the semantics of the relation *IsATranslationOf*, that relates a technical report to a translation of it, could be expressed as:

- $\forall x \neg IsATranslationOf(x,x)$
- $\forall x,y\ IsATranslationOf(x,y) \Rightarrow \neg IsATranslationOf(y,x)$
- $\forall x,y\ IsATranslationOf(x,y) \Rightarrow Author(x) = Author(y)$
- $\forall x,y\ IsATranslationOf(x,y) \Rightarrow Subject(x) = Subject(y)$
- $\forall x,y\ IsATranslationOf(x,y) \Rightarrow ResourceIdentifier(x) \neq ResourceIdentifier(y)$
- $\forall x,y\ IsATranslationOf(x,y) \Rightarrow Language(x) \neq Language(y)$

4.2 How to support relations within Dienst

This section discusses which services are required to handle the relation element and gives suggestion on how they could be implemented within the current version of Dienst. The presentation is organised in three subsections, one for each module of Dienst that has to be modified.

4.2.1 Repository

In order to handle the metadata relation, both the structures and the services that are provided by the Repository module of the Dienst system have to be modified.

First of all, the structure that maintains the metadata record has to be extended to maintain also the new metadata element. This can be done easily by exploiting the extension capabilities of Dienst.

Moreover, the information that regards the relation types has to be stored. This could be done, for example, by introducing, for each relation type, an appropriate structure. This structure (called hereafter *RelCont*) could, for example, consists of a sequence of references. The first could be a reference to a text that lists the properties of the relation type. The other references could be used to point to procedures which implement the relation element specific services. For example, there could be a reference to a procedure that is invoked when a new metadata that contains the corresponding relation type is created.

Note that this structure can be easily implemented and permits an easy dynamic addition of novel types.

In order to handle the element relation some of the existing Repository services has to be modified and additional services has to be added.

The service that deposits TR and their metadata has to include the check on the consistency between the value of the relation field and the rest the registered metadata values. This new functionality could, for example, be obtained by invoking, for each of the relations specified in the metadata record, the checking procedures stored in the appropriate *RelCont*.

Note that, this consistency check hides an inherent difficulty. The checking process involves different TR. These servers could be not responding at the time the check is issued. Until now we have not reached a clear understanding of which behaviour should the system exhibit when this situation occurs.

As a further effect of the introduction of relations, a new Repository service for updating relations has to be introduced. As Dienst does not support updates, the only possible solution is to simulate it by first removing a metadata record and then inserting a new one with the modified values. This solution, however, is quite heavy since, given the characteristics

of the current services of Dienst, not only the metadata has to be removed and added again, but also the document.

4.2.2 Index

In order to handle relations, for each of the relation types that are permitted, a new index has to be introduced. Dienst allows to make these extensions quite easy.

Moreover, specific services for retrieving TR through the relation links has to be provided. A possible solution for implementing such retrieval is to simulate the queries by exploiting the inverse relations. The query "getTR with *Author* = Carlesi" could be equivalently expressed as: "getTR with TR1 in *HasVersion*", where "in" stand for the set inclusion operator. The translation of the query in the appropriate form can be implemented by a procedure stored in *RelCont*.

Note that this solution is possible only if each time the direct relation is registered, the inverse is registered too. This last registration could be done automatically by modifying appropriately the creation service. The implementation of this service, however, it is not trivial since the metadata record to be modified might reside on a different server. In the Dienst architecture there is no protocol for either requiring remote updates and for being ensured that a request for a remote update is served.

4.2.3 User interface

Both the registering and querying interface have to be modified to handle the relation metadata.

In order to guide the administrator, a pull down menu with the list of relation types that can be specified should be available on the registering interface. Moreover, an help facility that visualises the semantic of each relation type should be provided on the same interface.

The querying interface should permit queries of the type "GetTR that are *R-related* to *x*" where R is a relation type and *x* is the identifier of a technical report.

New interfaces should also to be developed. In particular: 1) an interface for retrieving and then updating metadata records and 2) an interface for registering information about a new relation type, i.e., the semantics of new relation types and the associated procedures.

5. Conclusions

This paper has reported the results of a preliminary investigation on how the ETRDL system could be extended to cover also the retrieval by relation among documents.

Several topics emerged from this investigation:

1. An informal description of the semantics of the relational types is not sufficient for supporting the required interoperability. An appropriate form for expressing this semantics must be defined.
2. A semantics like the one that we have proposed is strictly dependent from the set metadata elements that has been chosen and by the set of their possible values. This means that there is no "general" definition of, for example, *IsVersionOf*, but the definition depends on the interaction with the chosen metadata. This may cause problems if the metadata element set is changed and if alternative metadata element sets are used.
3. Relation, differently from other metadata, is not necessarily independent from the other metadata. This raises the problem of the maintaining the consistency among the metadata values. This problem is made hard by the distribution of the documents that may reside on different servers.

The updates are not necessarily guaranteed, while the checks are not guaranteed to be completed.

4. Relations can be dynamically added. A mechanisms for updating a metadata record has to be provided.
5. The retrieval by relation has a different form than that of the other metadata elements. Given a document, the interest is in finding all the related documents. The existing query language needs to be appropriately extended.

The paper has proposed a solution to be implemented within the Dienst framework for some of the above items. Not all the solutions are, however, satisfying enough. A more powerful architecture seems to be required for better supporting the metadata relation.

References

[1] *Metadata Resources - Dublin Core*,
http://www.ukoln.ac.uk/metadata/resources/dc.html

[2]*A User Guide for Simple Dublin Core*,
http://128.253.70.110/DC5/UserGuide3.html

[3] C. Lagoze, E. Shaw, J. R. Davis and D.B. Krafft, *Dienst: Implementation Reference manual*, Cornell Computer ScienceTechnical Report TR95-1514,
http://cs-tr.cscornell.edu:80/Dienst/UI/2.0/Describe/ncstrl.cornell/tr96-1595.

[4] S. Biagioni, ERCIM Technical Reference Digital Library, *ERCIM News*, n.33, April 1998.

[5] Resource Description Framework(RDF) Model and Syntax,
http://www.w3.org/TR/WD-rdf-syntax/

[6] Resource Description Framework(RDF)Schemas,
http://www.w3.org/TR/WD-rdf-schema/

[7] C. Lagoze, J. R. Davis *Dienst: an Architecture for Distributed Document Libraries*, Communications of the, 38 (4) April 1995, page 45.