

# SeLeNe Report: Metadata Management and Learning Object Composition in a Self eLearning Network

Philippe Rigaux and Nicolas Spyratos

*Laboratoire de Recherche en Informatique  
Université Paris-Sud Orsay, France  
rigaux@lri.fr, spyratos@lri.fr*

## Abstract

In this report, we propose a simple data model for the composition and metadata management of *learning objects* (LOs) in a distributed setting that we call a *Self-eLearning Network*, or *SeLeNe* for short. The model features an abstract definition of LO, as well as operations that allow to compose new LOs from simpler ones, and to query a database of LOs spread over the network. The querying process is supported by a coordinator, or *mediator* which relies on descriptions of the LO content through *indexes*. The index of an LO is extracted automatically from the indexes of its components.

The model provides support for the specification of a concrete SeLeNe environment in which the authors of LOs are the basic peers and where communication between peers is assisted by the mediator. We illustrate the features and functionalities of such a system with a case study in which the LOs are XML documents.

## 1 Introduction

In this report, we consider the architecture and functioning of a *Self eLearning Network*, or *SeLeNe* for short, in which a community of authors co-operate through a coordinator, or *mediator*, in creating *Learning Objects* (LOs) to be used by a community of learners.

Each author is a “provider” of LOs to the network but also a “consumer”, in the sense that he creates LOs based not only on other LOs that he himself has created but also on LOs that other authors have created and made available to the network. We assume that each author is free to use his own organization, format and tools to manage his LOs locally, independently of all other authors. Thus we view a SeLeNe as a peer network, in which the basic peers are the authors co-operating through the network for serving a community of learners. The present report focuses on the definition of LOs, and their use during the collaborative process enabling the creation of new LOs. We ignore content management issues and focus on those aspects of metadata management that we consider essential to the creation, use and exchange of LOs.

In a nutshell, a SeLeNe is a distributed repository of educational metadata describing LOs available on the Web, collaboratively built and used by anyone who wishes to use existing LOs, or construct new ones in some knowledge domain. A SeLeNe allows authors to create LOs either from scratch or from LOs that other authors have created and made available to the network. To make a LO available to the network its author must register it with a mediator, i.e., a central server that keeps track of all LOs available and answers queries by authors searching for LOs of a specific kind.

Each registered object is accompanied by an *index*, a *composition graph* and a *dependency graph*. The index gives information about the content (thus guiding the search of authors for LOs that match a given

task), the composition graph shows the structure of the LO (thus guiding the automatic derivation of composite LO indexes from those of their components), and the dependency graph shows the possible ways in which the LO can be used in the learning process (thus guiding the choice of a sequence in which the component LOs should be learned by a specific user). In this introductory section, we give an overview of the basic aspects of a SeLeNe, i.e., the creation or authoring of LOs, the mediating process and the learning process. In doing so, we often use the term “network” as a synonym for “Self eLearning Network”.

## Authoring of Learning Objects

The basic building blocks for constructing LOs are the *atomic* LOs. Intuitively, an atomic LO is any piece of information (text, image, sound, etc.) that can be identified uniquely, using an identifier provided by the network. The LO content can be described using a set of *terms* from a commonly agreed, standard *terminology* (e.g., the ACM Computing Classification System); this set of terms is called the *index* of the LO, and we say that each term of this set *indexes* the LO. We shall refer to the commonly agreed, standard terminology as the network terminology, or terminology for short.

The granularity of an atomic object is entirely up to its author, i.e., an atomic object can be a piece of text, such as a whole book, a chapter or a section of a book, a paragraph or even a phrase; similarly, it can be an image or part of an image, a musical score or part thereof, and so on. As we mentioned earlier, atomic LOs are the basic building blocks for constructing new LOs called *composite LOs*.

A composite LO consists of a set of *parts*, i.e., a set of other LOs, each of which can be atomic or composite. For example, in the case study that we shall present in Section 5, a composite object will be an XML document and its parts will be other XML documents or fragments thereof. Like an atomic object, a composite object is associated with an identifier provided by the network and a description giving information about its content. However, this time, the index of a composite LO is derived from the indexes of its parts. For the purposes of this report, we assume the composition graph of a composite LO to be a tree, with the LO as the root and its parts as the roots of its sub-trees. We refer to the LOs appearing in the tree as the *components* of the root LO. If a component is atomic then this component is a leaf. Using the composition graph, we can derive the index of a composite LO (recursively) as follows:

- the index of each atomic LO is provided by its author;
- the index of each composite LO is the union of the indexes of its parts.

For example, consider two atomic LOs and their indexes (as given by their authors):

$$o_1: \{\text{Quick-sort, Java}\} \quad o_2: \{\text{Bubble-sort, Pascal}\}$$

The index of  $o_1$  says that it contains material on the Quick-sort algorithm written in Java, and that of  $o_2$  that it contains material on the Bubble-sort algorithm written in Pascal. Then if one defines a composite LO, say  $o_3$ , with parts  $o_1$  and  $o_2$ , its index will be derived automatically from those of  $o_1$  and  $o_2$ :

$$o_3: \{\text{Quick-sort, Java, Bubble-sort, Pascal}\}$$

Trivially, the composition graph of an atomic LO consists of a single node together with the identifier and the index of the atomic LO. As we shall see, the composition graph of a LO, together with the indexes of its leaves, contains all the information needed by the mediator in order to make the LO available to the network.

Each LO, whether atomic or composite, has a content. As we have already mentioned, the content of an atomic object can be a piece of text, an image, a sound, and so on, while the content of a composite object should be defined based on the contents of its components. However, content management issues lie outside the scope of the present report.

Typically, an author wishing to compose a new LO will use some of the objects in his local database as components and will also query the mediator in search of relevant LOs available over the network. He will

then inspect the answer in order to select those LOs most relevant to the LO being composed, based on their indexes. He may also wish to consult their contents, therefore user-friendly tools for browsing content are especially useful here.

We stress the fact that we view each LO, whether atomic or composite, as an identifier. If the LO is a document in electronic form, then such an identifier could be a URI where the LO content resides; one could then browse its content by “clicking” on that URI. Similarly, if the LO is a book in the traditional report form, then such an identifier could be an ISBN identifier, and so on. For the purposes of this report, however, we take an abstract point of view and we assume that a LO identifier is just a natural number.

### **The Mediating Process**

The mediator is a software module that acts as a central server, keeping track of the objects currently available to the network in order to answer queries asked by authors (and/or learners) [TSC01]. To carry out these tasks, the mediator maintains two pieces of information:

1. A commonly agreed *taxonomy*, i.e., a terminology together with a subsumption relation between terms (to which all authors adhere).
2. For each term  $t$ , the extension of  $t$ , i.e., the set  $E(t)$  of all LOs indexed by  $t$ .

In order to make a LO available to the network, its author must register it with the mediator. Registration of a LO consists in making the composition graph (along with the indexes of its leaves) and the dependency graph of the LO available to the mediator. Upon registration, the mediator updates the extension  $E(t)$  of every term  $t$  appearing in the composition graph, by adding to  $E(t)$  the identifier of each LO indexed by  $t$ . Authors and learners in search of LOs of a specific kind address their queries to the mediator. A query is either a single term or a boolean combination of terms, and its answer is a set of LOs computed in a manner that we shall see shortly.

It is important to note that the answer to a query is a set of LOs and that the issuer of the query can access not only the content of each LO contained in the answer, but also its composition and dependency graph.

### **The Learning Process**

Learners access the LOs available over the network in order to learn from their contents. A learner can either be assisted by an instructor (who might very well be the author of the LO himself) or do self-learning. In either case, the appropriate LOs are accessed by querying the mediator.

Once a LO has been selected, the learning process is assisted by the dependency graph of the LO. Indeed, based on the background of the learner, the dependency graph can assist in selecting the components to be learned and in defining the sequence in which the selected components should be learned. Each such sequence is what we call a learning trail or simply a *trail*.

Clearly, the definition of a learning trail depends on the individual learner and the specific LO used during the learning process. One can even envisage the possibility of combining more than one LO in order to define the desired learning trail for a given learner. In this case, each learning trail can be thought of as a LO and the combined trail as a new, composite LO.

In case where a learning trail concerns not just one learner but a group of learners (e.g., a whole class, or a specialized diploma course) it may be useful to materialize the trail, i.e., to “freeze” its content in time, by printing it out in the form of a book (in the traditional report form). In doing so, an interesting problem is producing automatically the “Table of Contents” of such a book, based on the composition graph of the

trail. Clearly, materializing the same trail at two different points in time may not yield the same book, as in the meantime some of the trail components may have been updated by their authors.

There are several learning scenarios that have been considered in the literature, that take into account the nature of the learning process, for example, whether the trail is chosen by the instructor and “imposed” to the learners, or whether the learner participates in the definition of the trail, or even if the trail is defined “collaboratively” in which case one talks of an “emerging” trail. The study of learning scenarios, however, lies outside the scope of the present report.

In the remaining of this report we present formally the basic assumptions on which a SeLeNe functions, in Section 2, the authoring of learning objects, in Section 3, and the mediating process, in Section 4. We also present a case study, in Section 5, in which the learning objects and their components are XML documents. Finally, we offer some concluding remarks and suggestions for further research, in Section 6.

## 2 The Basic Assumptions

From an abstract point of view, the functioning of a SeLeNe relies on the following basic assumptions:

- The existence of a countably infinite set whose elements are used by all authors for identifying the created LOs that are made available to the network. In fact, we assume that the creation of a LO is tantamount to choosing a (new) element from that set. For notational convenience we assume this set to be the set of positive integers, and the creation of a LO to be tantamount to choosing a (new) integer. Moreover, we assume that each created LO is associated with three entities, its parts, its trails and its index.
- The existence of a taxonomy whose terms are used by all authors for indexing the created LOs. There are several standardized taxonomies that one might use, depending on the knowledge domain, e.g., the ACM Computing Classification System. However, for the purposes of this report, we shall assume an abstract taxonomy, consisting of a terminology  $T$  and a subsumption relation  $\preceq$ . The terminology  $T$  is just an abstract set but it will help thinking of its elements as being keywords of some kind; we shall refer to the elements of  $T$  as *terms*. As for the subsumption relation, it is a reflexive and transitive relation over  $T$  (i.e., a pre-order over  $T$ ). If for terms  $s$  and  $t$  we have  $s \preceq t$ , it will be helpful to think that, conceptually,  $s$  is covered or subsumed by  $t$ , or that  $t$  covers or subsumes  $s$  (e.g., Quicksort  $\preceq$  Algorithm). Finally, two terms  $s$  and  $t$  are called *equivalent*, or *synonym*, denoted  $s \sim t$ , if  $s \preceq t$  and  $t \preceq s$ .
- The existence of a table, or catalogue  $C$ , relating the terminology  $T$  with the set of all LOs currently available in the network. The contents of  $C$  will be described by two functions:
  1. the *extension*  $E$ , mapping each term  $t$  to a set  $E(t)$  of LOs, defined by:  $o \in E(t)$  iff  $(t, o) \in C$ ;
  2. the *index*  $I$ , mapping each LO  $o$  to a set  $I(o)$  of terms, defined by:  $t \in I(o)$  iff  $(t, o) \in C$ .

## 3 Authoring of Learning Objects

As mentioned earlier, each created LO is associated with three entities, its parts, its trails and its index.

**Definition 1 (The Parts of a LO)** *Each learning object  $o$  is associated with a set of learning objects, called the parts of  $o$  and denoted as  $parts(o)$ . If  $parts(o) = \emptyset$  then  $o$  is called atomic, else it is called composite.*

Clearly, what the parts of an object are should be left entirely up to its author. We can represent a LO and its parts graphically, as follows: if  $i$  has  $j$  as a part then we draw an arrow from  $i$  to  $j$ . Thus, if  $parts(o) = \{o_1, o_2, \dots, o_n\}$  then we represent this graphically as in Figure 1.

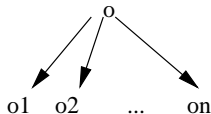


Figure 1: A LO and its parts

Based on the concept of part, we can now define the concept of component.

**Definition 2 (Components of a LO)** *Let  $o$  be a LO and let  $parts(o) = \{o_1, o_2, \dots, o_n\}$ . The set of components of  $o$ , denoted as  $comp(o)$ , is defined recursively as follows:*

*if  $o$  is atomic then  $comp(o) = \emptyset$  else  $comp(o) = parts(o) \cup comp(o_1) \cup comp(o_2) \cup \dots \cup comp(o_n)$ .*

Clearly, a LO  $o$  and its associated set of components can be represented as a directed acyclic graph with  $o$  as the single root. We shall refer to this graph as the *composition graph* of  $o$ . We make two assumptions on the composition graph, both reflecting reasonable, real-life constraints:

1. All nodes of the composition graph are different, i.e., no LO in the composition graph is a component of itself.
2. Every LO in the composition graph (except the root) is part of one and only one LO.

We note that these assumptions say, roughly, that a composite LO should not contain redundancies. An immediate consequence of these assumptions is that the composition graph is a tree.

When an author creates a composite LO, apart from making the components visible (through the composition graph), it is natural to give an indication as to the possible sequences in which the various components should be taught by an instructor, or learned by a self-teaching learner. This can be done by giving a partial order “precedes” on the components of the LO, saying what component precedes, or is pre-requisite for, what other components. Such a partial order can also be represented as a directed acyclic graph, that we shall refer to as the *dependency graph* of the LO.

**Definition 3 (The Dependency Graph of a LO)** *Let  $o$  be a composite LO. The dependency graph of  $o$ , denoted as  $dep(o)$ , is a partial order over the set of components of  $o$ .*

We note that the dependency graph is different than the composition graph. In particular, the dependency graph may have more than one root, each root indicating a possible starting component in teaching (or learning) from the composite LO.

We also note that most of the recent textbooks (at least in Computer Science) do contain such a dependency graph for assisting the instructor in designing his course, or the self-learning student in learning from the textbook.

Indeed, in analogy to what happens with a traditional textbook, a composite LO might be used in several different ways. For example, depending on background, a learner might skip some components (chapters and/or sections), or might decide to follow a specific order in learning the various components. Similarly, an instructor might decide to use a composite learning object in various ways in designing a course, depending on his own background and that of the learners.

Assuming that the author of a composite LO is the one most familiar with its content, any sequence of components that “respects” the dependency graph given by the author is a “good” sequence in which the components should be taught, or learned.

**Definition 4 (Learning Trail)** Let  $o$  be a composite LO. A learning trail, or simply trail in  $o$  is any path in the transitive closure of the dependency graph.

## 4 The Mediating Process

As we mentioned in the introduction, the mediator is a software module that acts as a central server, keeping track of the LOs currently available to the network in order to answer queries asked by authors (and/or learners). To carry out these tasks, the mediator maintains two pieces of information:

1. The network taxonomy, updating its terminology and/or its subsumption relation whenever this is necessary.
2. For each term  $t$ , the extension of  $t$ , i.e., the set  $E(t)$  of all LOs indexed by  $t$ ; as we shall see shortly, the extension is used for the evaluation of queries.

In order to make a LO available to the network, its author must register it with the mediator. Registration of a LO  $o$  by an author consists in providing the following items to the mediator:

- the composition graph of  $o$  and the indexes of its leaves,
- the dependency graph of  $o$ .

Upon registration, the mediator performs two actions:

1. computes the index of every composite LO  $o'$  in the dependency graph of  $o$ , recursively, based on the indexes of the atomic LOs (leaves), as follows:

$$\text{if } parts(o') = \{o_1, o_2, \dots, o_n\} \text{ then } I(o') = I(o_1) \cup I(o_2) \cup \dots \cup I(o_n)$$

2. updates the extension  $E(t)$  of every term  $t$  in  $T$  as follows:

$$\text{for each } o' \text{ in } comp(o), \text{ if } t \text{ is in } I(o') \text{ then } E(t) := E(t) \cup \{o'\}$$

Authors and learners in search of LOs of a specific kind address their queries to the mediator. A query is either a single term or a boolean combination of terms, as stated in the following definition.

**Definition 5 (Query language)** A query over the network taxonomy is any string derived by the following grammar, where  $t$  is a term and  $\epsilon$  is the empty query:

$$q ::= t \mid q \wedge q' \mid q \vee q' \mid q \wedge \neg q' \mid (q) \mid \epsilon$$

(Our use of negation corresponds to domain restricted negation).

The answer to a query  $q$ , denoted by  $ans(q)$ , is a set of LOs defined as follows:

```

ans(t): if tail(t) =  $\emptyset$  then I(t) else  $\bigcup \{ans(s) \mid s \in tail(t)\}$ 
ans(q):
    if q = t then ans(t)
    else
    begin

```

```

if  $q = q_1 \wedge q_2$ ,  $ans(q) = ans(q_1) \cap ans(q_2)$ 
if  $q = q_1 \vee q_2$ ,  $ans(q) = ans(q_1) \cup ans(q_2)$ 
if  $q = q_1 \wedge \neg q_2$ ,  $ans(q) = ans(q_1) \setminus ans(q_2)$ 
end
 $ans(\epsilon) = \emptyset$ 

```

It is important to note that the answer to a query is a set of LOs and that the issuer of the query can access not only the content of each LO contained in the answer, but also its composition and dependency graph.

We conclude this section by stressing the fact that the composition graph, the dependency graph, and the index of a LO constitute only part of the metadata that accompany a LO. Other kinds of metadata are also used when one searches for LOs that match given needs. For example, the format in which a LO is available, the language in which it is written, the date of creation, and so on, are examples of metadata associated to a LO. However, the composition graph, the dependency graph, and the index of a LO are probably the most challenging from a technical point of view.

## 5 A Case Study

We are currently implementing a prototype to experiment in a practical setting the functionalities of the model. In this prototype the LOs and their components are XML documents, and the system relies on XML tools and languages for addressing and transformation tasks.

The architecture of the system is summarized in Figure 2. Here are some comments, before looking into the technical details. First the composite LOs are represented in this specific implementation by XML *documents* which are *valid* with respect to the DocBook DTD [WM99]. The hierarchical nature of XML documents fits well with the composition mechanism of our model, which allows to construct composite LOs from simpler ones. Each fragment of the XML structure (i.e. each subtree) corresponds to a LO, and the leaves are the atomic LOs introduced in the model. When submitting a document to the system, it is required that each of the leaves is labelled with a set of terms from the network terminology.

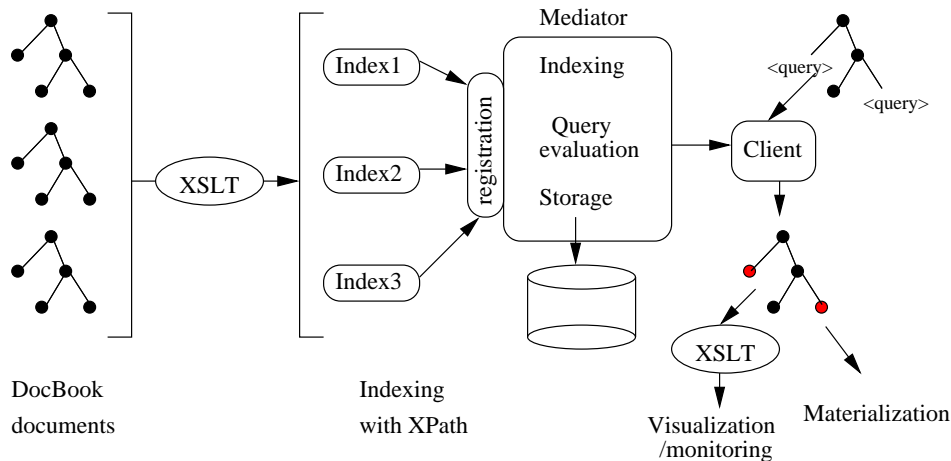


Figure 2: Overview of the system's architecture

From these documents, a program (written with the XML transformation language, XSLT) produces the index for each document. Indexes are sent to the mediator which stores them in a repository, creates indexes on objects, and proposes querying services. Finally users can create composite LOs as DocBook documents

augmented with the <query> element. The content of such an element is a query which is executed by the mediator and replaced by the result of the query. From this the user can:

- either browse through the query result, visualize the fragments coming from atomic documents, and possibly remove some of them,
- or materialize the document, including the result of queries, and store it locally.

We now embark in a detailed description of each part.

## 5.1 The terminology

The terminology used in the system is the ACM Computing Classification System (see <http://www.acm.org/class/>). It is initially designed to classify published works in the field of computer science. We use an XML representation, stored at the mediator and accessible through a Web interface to users and authors who want to pick up terms for, respectively, indexing their documents or expressing queries. Here is a small part of this terminology.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- http://www.wikipedia.org/wiki/Computer_science-->

<term name="Computer_Science">

<term name="Artificial_intelligence">

    <term name="Knowledge_representation"/>

    <term name="Machine_learning">
<term name="analytical_learning"/>
<term name="artificial_neural_networks"/>
    <term name="algorithms_for_pattern_discovery"/>
        (...)
    </term>

    (...)
</term>

<term name="Databases">
    <term name="Database_management_system">

<term name="Closed_source">
    <term name="Oracle"/>
    <term name="Teradata"/>
(...)
</term>

<term name="Open_source">
    <term name="MySQL"/>
    <term name="PostgreSQL"/>
(...)<
</term>
</term>

<term name="SQL">
```

```

        (...)
    </term>

    <term name="Relational_model">
        <term name="Relational_schema">
            <term name="Constraint"/>
            (...)
        </term>
    </term>
</term>
</term>
</term>

```

## 5.2 Documents

DocBook is a DTD for writing structured documents using SGML or XML. It is particularly well-suited to books and papers about computer hardware and software, though it is by no means limited to them. DocBook is an easy-to-understand and widely used DTD: dozens of organizations use DocBook for millions of pages of documentation, in various print and online formats, worldwide. Many publishers use DocBook to represent and exchange their books or parts of their books, and given the wide acceptance of this DTD and its maturity, it seems reasonable to adopt it as a *de facto* standard.

It is worth mentioning however that any other DTD would do, the important assumption here being that *all* authors in the system provide their LO content in a common format. This assumption is mostly motivated by practical considerations. Indeed the exchange of fragments and their integration is greatly facilitated by the homogeneity of the representation. In particular, it is easy with minimal effort to ensure that inserting a DocBook fragment in a DocBook document keeps the whole document valid with respect to the DTD.

We distinguish in a DocBook document the following tags that identify the *structure* of the document: `book`, `chapter`, `section` and `subsection`. Elements of type `subsection` are considered to form the leaves of the composition graph, to which an index must be associated. The inference mechanism described in the model is then used to create the indexes for the upper-level elements `book`, `chapter` and `section`. As an example, here is a (quite simplified) document:

```

<book title="Databases">

    <chapter title="Conceptual modelling">
        Some text ...
        <section title="The Entity-relationship model">
            Some text ...
        </section>
        <section title="Schema design using functional dependencies">
            Some text ...
        </section>
    </chapter>

    <chapter title="Database programming">
        Some text ...
    </chapter>
</book>

```

Beyond the (somehow heavy) syntax of XML, we are interested in the *structure* of the information contained in this document. This structure is defined by the tags and is better represented as a tree, shown in Figure 3.

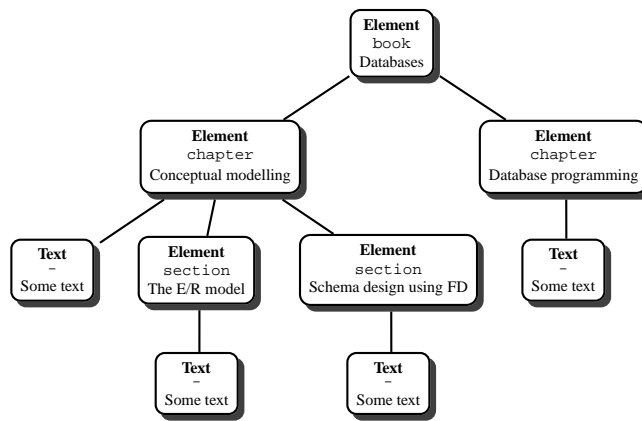


Figure 3: The hierarchical structure of the document

Essentially, nodes of type `Text` represent the content, while nodes of type `Element` represent the structure. The role of the author, before submitting such a document to the mediator, is to index the elements located at the lower level in the structure (here `<section>`) with terms from the terminology. This is simply done by adding an attribute to the `<section>`, as illustrated in the document below:

```

<book title="Databases">

  <chapter title="Conceptual modelling">
    Some text ...
    <section title="The Entity-relationship model" term="E/R">
      Some text ...
    </section>
    <section title="Schema design using functional dependencies"
      term="FD">
      Some text ...
    </section>
  </chapter>

  <chapter title="Database programming">
    Some text ...
  </chapter>
</book>

```

We obtain a new structure derived from the previous one, and illustrated in Figure 4. The document represented in this figure contains indexes for *all* the elements, at any level. The indexes for `<chapter>` and `<book>` elements have been derived automatically from the indexes of the leaves in the way explained earlier.

Finally the composition graph together with the indexes of the leaves is sent to the mediator who stores, with each term of the terminology, the path to the XML subtree(s) that relate(s) to this term. Currently we use the XPath language [XP99] to refer to these subtrees, and complete XPath expressions with the URL of the document. The table below shows the information handled by the mediator to refer to the nodes of the document used so far.

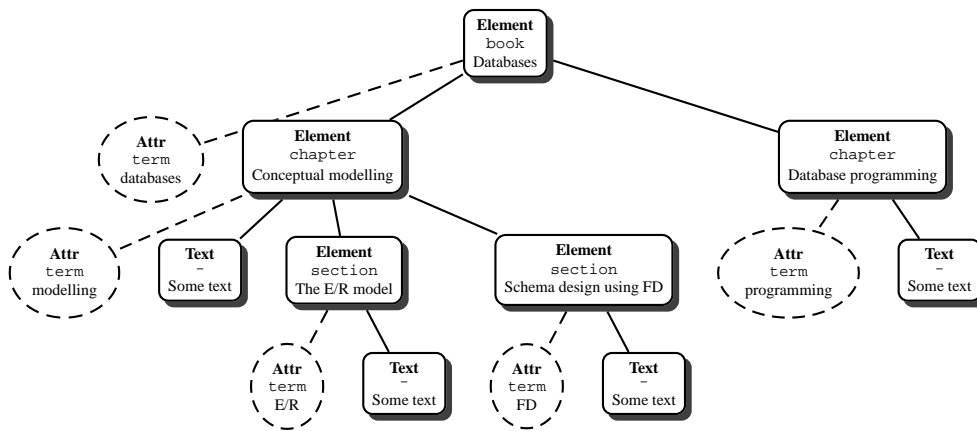


Figure 4: A document enriched with indexes

Term	XPath expression
databases	/book
modelling	/book/chapter[1]
E/R	/book/chapter[1]/section[1]
E/R	/book/chapter[1]/section[2]
programming	/book/chapter[2]

Finally let us illustrate how one can create composite documents by inserting *queries*. The example of Figure 5 shows the structure of a DocBook document, enriched with `<query>` elements that allow to express queries. In this particular example, the document is that of a student who collect course notes, introduces his own course notes, and mixes them with fragments/LO extracted from the set of available sources. When submitted to the mediator via a client/server dialog whose description is omitted here, the `<query>` is replaced by the content of the answer to the query (or, more generally, by the concatenation of the contents of the set of XML subtrees obtained as query results).

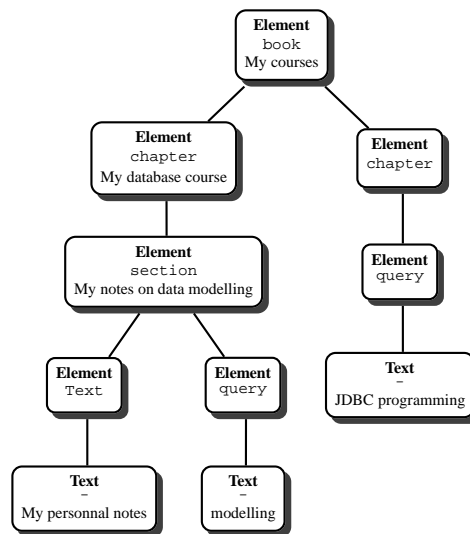


Figure 5: A derived document

## 6 Concluding remarks

Intuitively, a LO should be regarded as a textbook (in its traditional form), and its composition graph as the context in which learning trails can be defined. The dependency graph, in turn, should be regarded as the set of all possible trails that can be followed, at least as far as the author of the LO sees his LO. Of course, the users of the LO, whether learners or instructors, may find other ways of using it (i.e., other trails).

From an abstract point of view, both, the composition graph and the dependency graph, can be seen as two different kinds of relations that one can define over the set of LOs available to the network. If one uses a semantically rich model (e.g., RDF), one can define these and other relations at the level of LOs, and write algorithms for their computation, whenever necessary.

One important issue that can be handled by our mediator is personalized interaction with the network. Indeed, from a conceptual point of view, all one has to do is to let the network user express his needs in terms of a set of named queries, or *views* of the form:

```
<term-name> = <query-to-the mediator>
```

The set of terms thus declared (plus, eventually, a user-defined subsumption relation) will then constitute the user-defined taxonomy, that will serve as the *personalized* interface to the network. Queries to this personalized taxonomy can be answered by simple substitution, based on the user declarations defining the terms of the personalized taxonomy. Work on the personalization aspects is ongoing and will be reported later.

## References

- [ACK<sup>+</sup>01] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *Proc. Intl. Conf. on Semantic Web*, 2001.
- [KK01] J. Kahan and M.-. Koivunen. Annotea: an Open RDF Infrastructure for Shared Web Annotations. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 623–632, 2001.
- [KSV<sup>+</sup>02] B. Kieslinger, B. Simon, G. Vrabic, G. Neumann, J. Quemada, N. Henze, S. Gunnersdottir, S. Brantner, T. Kuechler, W. Siberski, and W. Nejdl. ELENA Creating a Smart Space for Learning. In *Proc. Intl. Semantic Web Conference*, volume 2342 of *LNCS*. Springer Verlag, 2002.
- [NWQ<sup>+</sup>02] W. Nejdl, B. Worlf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. EDUTELLA: a P2P networking Infrastructure Based on RDF. In *Proc. Intl. World Wide Web Conference (WWW)*, page 604:615, 2002.
- [NWS<sup>+</sup>03] W. Neidl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. Super-Peer Based Routing and Clustering Strategies for RDF-based Peer-to-Peer networks. In *Proc. Intl. World Wide Web Conference (WWW)*, 2003.
- [SEL] SeLeNe: Self eLearning Networks. See <http://www.dcs.bbk.ac.uk/ap/projects/selene/>.
- [TSC01] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. Mediators over Ontology-based Information Sources. In *Proc. Intl. Conf. on Web Information Systems Engineering (WISE'01)*, 2001.
- [WM99] N. Walsh and Leonard Mueller. *DocBook, the definitive guide*. O'Reilly, 1999.

[XPa99] The XPath language recommendation (1.0). World Wide Web Consortium, 1999.  
<http://www.w3.org/TR/xpath>.