RAPPORT DE RECHERCHE

L R I

GENERATION AND SYNDICATION OF LEARNING OBJECT METADATA

RIGAUX P / SPYRATOS N

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud-LRI

10/2003

Rapport de Recherche N° 1371

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 650
91405 ORSAY Cedex (France)

Generation and Syndication of Learning Object Metadata

Philippe Rigaux and Nicolas Spyratos

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

October 24, 2003

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. We assume that each LO resides at the local repository of its author, so all authors' repositories, collectively, can be thought of as a database of LOs spread over the network. Authors willing to share their LOs with other authors in the network must register them with a coordinator, or *syndicator*, and authors that search for LOs matching their needs must address their queries to the syndicator.

The process of registering (or un-registering) a LO, formulating a query to the syndicator, or answering a query by the syndicator, all rely on LO content description. In this report, we focus only on one dimension of content description, namely subject area.

During registration of a LO o, if o is atomic its author is required to submit a description - the author description - whereas if the LO is composite the author description is optional but the author is required to submit all the LOs that are parts of o; based on the parts, the syndicator then computes the *implied* description of o. In this respect, the main contributions of the present deliverable are:

1. providing an appropriate definition of LO description;
2. providing an algorithm for the computation of the implied description;
3. defining the main functionalities that the syndicator should support, in particular, a querying facilities.

This report also contains a case study that illustrates some features and functionalities of a SeLeNe in which the LOs are XML documents, and the network is served by a single syndicator (see Appendix A). This case study has been conducted in the context of a Master thesis, within our research group.

Integration of our results with those of our SeLeNe partners has started since end July 2003, and concerns two activities:

1. Integration of our algorithms into the change propagation module developed by Birkbeck (see Deliverable 4.4).
2. Embedding of our model in the RDF suite developed by ICS-FORTH (see Appendix B).

We stress the fact that, in this deliverable, we do *not* deal with the management of LO content, but only with the management of content description, and in particular with subject area description. Therefore our model and query language capture *only* one part of content description, and further work is needed to extend the model to other kinds of LO metadata and hence to other modes of querying.

Chapter 1

Introduction

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 [9].

In a SeLeNe, a community of LO *authors* co-operate in the creation of LOs to be used also by other authors and 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 are willing to share [4]. In a nutshell, our approach can be described as follows.

We distinguish LOs into *atomic* and *composite*. Intuitively, an atomic LO is any piece of learning material (text, image, sound, etc.) that can be identified uniquely; its nature and granularity are entirely up to its author. A composite LO consists of a set of *parts*, i.e., a set of other LOs that can be either atomic or composite. We assume that each LO resides at the local repository of its author, so all authors' repositories, collectively, can be thought of as a database of LOs spread over the network. Typically, an author wishing to create a new LO will use some of the objects in his local database as components and will also search for relevant LOs available over the network.

Authors willing to share their LOs with other authors in the network must register them with a coordinator, or *syndicator*, and authors that search for LOs matching their needs must address their queries to the syndicator. The process of registering (or un-registering) a LO, formulating a query to the syndicator, or answering a query by the syndicator, all rely on LO content *description*.

Such descriptions are actually sets of terms from a controlled vocabulary, or *taxonomy*, to which all authors adhere. The well known ACM Computing Classification System [1] is an example of such a taxonomy. In this respect, we distinguish three kinds of description: the author description, the implied description and the registration description.

During registration of a LO at the syndicator, its author is required to submit the following items:

1. The LO identifier, say o; this can be a URI allowing to access the LO.
2. A description of the LO content, that we call the *author description* of o; if o is atomic then the author description must be nonempty, whereas if o is composite then the author description can be empty.
3. If o is composite, then registration requires, additionally, the submission of all parts of o (i.e., all LOs that constitute the LO being registered); using the descriptions of these parts, the syndicator then computes *automatically* a description that "summarizes" the descriptions of the parts, and that we call the *implied description* of o.

To register a LO the syndicator uses the author description augmented by the implied description, after removing all redundant terms (i.e., terms that are subsumed by other terms). The final set of terms used for registration is what we call the *registration description*.

The syndicator is actually a software module that maintains a *catalogue* of registered LOs: during registration of a LO with identifier o, the syndicator inserts in the catalogue a pair (t, o), for each term t in the registration description of o. Authors and learners searching for LOs that match their needs address their queries to the syndicator. In turn, the syndicator uses the catalogue to answer such queries.

The main issues addressed in this report are:

1. providing appropriate definition of LO description;
2. providing an algorithm for the computation of implied descriptions;
3. defining the main services that the syndicator should provide.

This report proposes *generic* solutions to the above issues, i.e., solutions that are valid independently of questions concerning network configuration. In other words, the solutions that we provide are still valid whether the network is configured around a central syndicator (as in Napster), or whether it is organized in clusters, each cluster being served by a separate syndicator, or even whether there is no syndicator but each node plays the role of a syndicator for all its connected nodes (as in pure peer-to-peer network).

We have tested our results in a case study that illustrates the features and functionalities of a SeLeNe in which the LOs are XML documents, and the network is served by a single syndicator (see Appendix A). This case study has been conducted in the context of a Master thesis, within our research group. Integration of our results with those of our SeLeNe partners has started since August 2003, and concerns two activities:

1. Integration of our algorithms into the change propagation module developed by Birkbeck (see Deliverable 4.4).
2. Embedding of our model in the RDF Suite developed by ICS-FORTH (see Appendix B).

We stress the fact that, in this deliverable, we do *not* deal with the management of LO content, but only with the management of content description, and in particular with subject area description. We are aware that, apart from subject area, there are several other dimensions of content description such as the format of the LO, its date of creation, its author, the language in which the LO content is written (if there is text involved), and so on. However, in this report, we focus only on the subject area dimension, and when we talk of content description we actually mean subject area description.

Therefore our model and query language capture *only* one dimension of content description, and further work is needed to extend the model to other kinds of LO metadata and hence to other modes of querying.

Chapter 2

The Representation of a Learning Object

As mentioned earlier, in our model, a LO is represented by an identifier together with a composition graph showing how the LO is constructed from other, simpler LOs. We do not consider the LO content itself, but focus only on its representation by an identifier and a composition graph, as this is sufficient for our metadata management and syndication purposes. Therefore, hereafter, when we talk of a LO we shall actually mean its representation by an identifier and a composition graph.

In order to define a LO formally, we assume the existence of a countably infinite set *Obj* whose elements are used by all authors for identifying the created LOs. For example, the set *Obj* could be the set of all URIs. In fact, we assume that the creation of a LO is tantamount to choosing a (new) element from *Obj* and associating it with a set of other LOs that we call its parts.

Definition 1 (The Representation of a Learning Object) *A LO consists of an identifier o together with a (possibly empty) set of LOs, 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, the choice of parts of a composite object and their arrangement to form a composition graph should be left entirely up to its author. For notational convenience, we shall write $o = o_1 + o_2 \ldots + o_n$ to stand for $parts(o) = \{o_1, o_2, \ldots, o_n\}$. We can represent a LO and its parts graphically, as follows: if o has o_i as a part then we draw an arrow from o to o_i. Thus, if $o = o_1 + o_2 \ldots + o_n$ then we represent this graphically as in Figure 2.1.

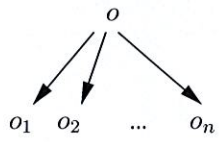

Figure 2.1: A LO and its parts

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

Definition 2 (Components of a Learning Object) *Let $o = o_1 + o_2 \ldots + o_n$. The set of components of o, denoted as $comp(o)$, is defined recursively as follows:*

$$\text{if } o \text{ is atomic then } comp(o) = \emptyset$$
$$\text{else } comp(o) = parts(o) \cup comp(o_1) \cup comp(o_2) \cup \ldots \cup comp(o_n).$$

In this report, we assume that a LO o and its associated set of components can be represented as a directed acyclic graph (*dag*) with o as the only root. We shall refer to this graph as the

composition graph of o. The composition graph of an atomic LO consists of just one node, the LO identifier itself. We note that the absence of cycles in the composition graph simply reflects the reasonable assumption that a LO cannot be a component of itself. Clearly, this does not prevent a LO from being a component of two or more distinct LOs belonging to the same composition graph, or to two different composition graphs.

It is important to note that in our model the ordering of parts in a composite LO is ignored because it is not relevant to our purposes. Many different composite LOs, with different arrangements of the same set of component LOs, have the same representation in the model. As we shall see shortly, deriving the description of a composite LO from the descriptions of its parts does not depend on any ordering of the parts. Therefore, we could see no reason for imposing an ordering on the parts.

Chapter 3

Descriptions of Learning Objects

As we mentioned in the introduction, LO content descriptions are built based on a controlled vocabulary, or *taxonomy*, to which all authors adhere. A taxonomy consists of a set of terms together with a subsumption relation between terms. An example of taxonomy is the well known ACM Computing Classification System [1].

Definition 3 (Taxonomy) *A taxonomy is a pair* $(T, \preceq)$ *where* T *is a* terminology, *i.e., a finite and non-empty set of names, or* terms, *and* $\preceq$ *is a reflexive and transitive relation over* T, *called* subsumption.

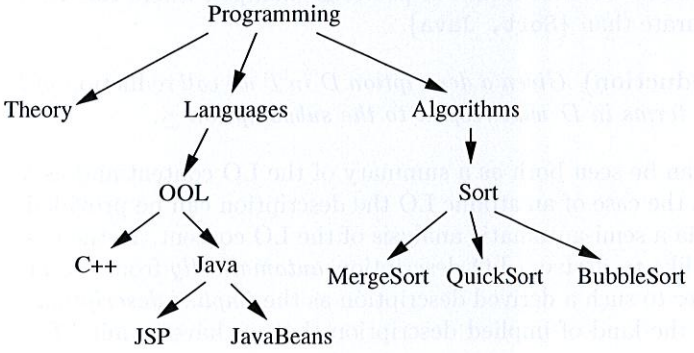

Figure 3.1: A taxonomy

If $s \preceq t$ then we say that s is *subsumed* by t, or that t *subsumes* s. A taxonomy is usually represented as a graph, where the nodes are the terms and there is an arrow from term s to term t iff s subsumes t . Figure 3.1 shows an example of a taxonomy, in which the term `Languages` subsumes the term `OOL`, the term `Java` subsumes the term `JavaBeans`, and so on. We note that the subsumption relation is *not* antisymmetric, i.e., $(s \preceq t)$ and $(t \preceq s)$ does not necessarily imply $s = t$. Therefore, we define two terms s and t to be *synonyms* iff $s \preceq t$ and $t \preceq s$. However, in this report, we shall not consider synonyms. From a technical point of view, this means that we work with classes of synonym terms, rather than individual terms. Put it differently, we work with just one representative from each class of synonyms. For example, referring to Figure 3.1, the term `OOL` is the representative of a class of synonyms in which one can also find terms such as `Object-Oriented Languages`, `O-O Languages`, and so on, that are synonyms of `OOL`.

However, even if we work only with classes of synonyms, a taxonomy is not necessarily a tree. Nevertheless, most taxonomies used in practice (including the ACM Computing Classification System mentioned earlier) are in fact trees. In this report, we shall assume that the taxonomy used by the authors of a SeLeNe to describe the contents of their LOs is in fact a tree. We shall refer to this tree-taxonomy as "the SeLeNe taxonomy", or simply "the taxonomy", for short.

Now, in order to make a LO sharable, we must provide a description of the content, so that users can judge whether the LO in question matches their needs. We define such a description to be just a set of terms from the taxonomy. For example, if the LO contains the quick sort algorithm written in java then we can choose the terms `QuickSort` and `Java` to describe its content. The set of terms {`QuickSort, Java`} can be used as a description of the LO.

Definition 4 (Description) *Given a taxonomy* $(T, \preceq)$ *we call* description *in* T *any set of terms from* T.

However, a problem arises with descriptions: a description can be redundant if some of the terms it contains are subsumed by other terms. For example, the description {`QuickSort, Java, Sort`} is redundant, as `QuickSort` is subsumed by `Sort`. If we remove either `Sort` or `Java` then we obtain a non-redundant description: either {`QuickSort, Java`} or {`Sort, Java`}, respectively. As we shall see later, redundant descriptions are undesirable as they can lead to redundant computations during query evaluation. We shall therefore limit our attention to non-redundant, or *reduced descriptions*, defined as follows:

Definition 5 (Reduced Description) *A description* D *in* T *is called* reduced *if for any terms* s *and* t *in* D, $s \not\preceq t$ *and* $t \not\preceq s$.

Following the above definition one can reduce a description in (at least) two ways: removing all but the minimal terms, or removing all but the maximal terms. In this report we adopt the first approach, i.e., we reduce a description by removing all but its minimal terms. The reason for our choice lies in the fact that by removing all but minimal terms we obtain a more accurate description. This should be clear from our previous example, where the description {`QuickSort, Java`} is more accurate than {`Sort, Java`}.

Definition 6 (Reduction) *Given a description* D *in* T *we call* reduction *of* D, *denoted* $reduce(D)$, *the set of minimal terms in* D *with respect to the subsumption* $\preceq$.

A description can be seen both as a summary of the LO content and as a support to find and retrieve the LO. In the case of an atomic LO the description can be provided either by the author or by the system via a semi-automatic analysis of the LO content. In the case of a composite LO, though, we would like to derive a LO description *automatically* from the descriptions of the LO parts. We shall refer to such a derived description as the *implied description* of the composite LO. To get a feeling of the kind of implied description that we have in mind for a composite LO, let us see an example.

Example 1 *Let* $o = o_1 + o_2$ *be a composite LO with the following descriptions of its parts:*

$$Descr(o_1) = \{\texttt{QuickSort, Java}\} \qquad Descr(o_2) = \{\texttt{BubbleSort, C++}\}$$

Then the implied description of $o = o_1 + o_2$ *will be* {`Sort, OOL`}, *that summarizes what the two parts have in common.*

We shall come back to this example after the formal definition of implied description. Now, the question is: how can one define the implied description of a composite LO so as to best reflect the contents of its parts. Roughly speaking, what we propose in this report is that the implied description of a LO should satisfy the following criteria:

- it should be reduced, for the reasons explained earlier;
- it should summarize what the parts have in common;
- it should be minimal.

To illustrate points 2 and 3 above, suppose that a composite LO has two parts with descriptions {QuickSort} and {BubbleSort}. The term Sort is a good candidate for being the implied description, as it describes what the two parts have in common. Moreover, as we can see in Figure 3.1, Sort is the minimal term with these properties. On the other hand, the term Algorithm is not a good candidate because, although it describes what the two parts have in common, it is not minimal (as it subsumes the term Sort).

Coming back to Example 1, following the above intuitions, we would like the implied description of o to be {Sort, OOL}. Indeed,

- {Sort, OOL} is a reduced description;
- the term Sort summarizes what QuickSort and BubbleSort have in common, and OOL summarizes what Java and C++ have in common;
- it is minimal, as any other description with the above properties will have terms subsuming Sort or OOL.

In order to formalize these intuitions, we introduce the following relation on descriptions.

Definition 7 (Refinement Relation on Descriptions) *Let D and D' be two descriptions. We say that D is* finer *than D', denoted $D \sqsubseteq D'$, iff for each $t' \in D'$, there exists $t \in D$ such that $t \preceq t'$*

In other words, D is finer than D' if every term of D' subsumes some term of D. For example, the description D ={QuickSort, Java, BubbleSort} is finer than D' = {Sort, OOL}, whereas D' is not finer than D. To gain some more insight into this ordering, let us see another example. Referring to Figure 3.1, consider the following reduced descriptions:

$$D = \{\texttt{JSP, QuickSort, BubbleSort}\} \qquad D' = \{\texttt{Java, Sort}\}$$

Then $D \sqsubseteq D'$, as each term t' of D' subsumes some term t of D. Indeed, Java subsumes JSP and Sort subsumes QuickSort (of course, Sort also subsumes BubbleSort, but the existence of one term in D subsumed by Sort is sufficient).

Note that, according to this ordering, once we have verified that $D \sqsubseteq D'$ we may add to D as many extra terms as we wish, *without* destroying the ordering. Thus, in our previous example, if we add to D the term Theory, D still remains finer than D'. This is consistent with our objective that the implied description should summarize what is common to *all* parts (and Theory is not common to all parts).

Clearly, $\sqsubseteq$ is a reflexive and transitive relation, thus a pre-ordering over the set of all descripions. However, $\sqsubseteq$ is *not* antisymmetric, as the following example shows. Consider D_1 = {OOL, Java, Sort} and D_2 = {Java, Sort, Algorithms}. It is easy to see that $D_1 \sqsubseteq D_2$ and $D_2 \sqsubseteq D_1$, although $D_1 \neq D_2$. However, as we have explained earlier, for the purposes of this report, we restrict our attention to reduced descriptions only; and, as stated in the following proposition, for reduced descriptions, the relation $\sqsubseteq$ becomes also antisymmetric, thus a partial order.

Proposition 1 *The relation $\sqsubseteq$ is a partial order over the set of all reduced descriptions.*

Proof. Indeed, assume $D \sqsubseteq D'$ and $D' \sqsubseteq D$, and consider a term t' of D'. Then there is a term t in D such that $t \preceq t'$. We claim that $t' \preceq t$ as well, and therefore that $t = t'$. Otherwise, as $D' \sqsubseteq D$ and t is in D, there is a term t'' (different than t') such that $t'' \preceq t$, and thus $t'' \preceq t'$. Assuming $t'' \neq t'$, we have a contradiction to the fact that D' is a reduced description. □

Now, using this ordering, we can define formally the implied description of a composite LO so as to satisfy the criteria for a "good" implied desription, given earlier. First, we need the following result:

Theorem 2 (Least Upper Bound of a Set of Reduced Descriptions) *Let $\mathcal{D} = \{D_1, .., D_n\}$ be any set of reduced descriptions. Let $\mathcal{U}$ be the set of all reduced descriptions S such that $D_i \sqsubseteq S, i = 1, 2, ..., n$, i.e., $\mathcal{U} = \{S | D_i \sqsubseteq S, i = 1, \ldots, n\}$. Then $\mathcal{U}$ has a least upper bound, that we shall denote as $lub(\mathcal{D}, \sqsubseteq)$.*

Proof. Let $P = D_1 \times D_2 \times \ldots \times D_n$ be the cartesian product of the descriptions in $\mathcal{D}$, and suppose that there are k tuples in this product, say $P = \{L_1, L_2, ..., L_k\}$. Let $D = \{lub_{\preceq}(L_1), lub_{\preceq}(L_2), \ldots, lub_{\preceq}(L_k)\}$, where $lub_{\preceq}(L_i)$ denotes the least upper bound of the terms in L_i, with respect to $\preceq$. As $(T, \preceq)$ is a tree, this least upper bound exists, for all $i = 1, 2, ..., n$. Now, let R be the reduction of D, i.e., $R = reduce(D)$. We shall show that R is the smallest element of $\mathcal{U}$.

Indeed, it follows from the definition of R that $D_i \sqsubseteq R$, for $i = 1, 2, ..., n$. Moreover, let S be any description in $\mathcal{U}$, and let t be a term in S. It follows from the definition of $\mathcal{U}$ that there is a term v_i in each description D_i such that $v_i \preceq t$. Consider now the tuple $v =< v_1, v_2, ..., v_n >$. By the definition of least upper bound, $lub_{\preceq}(v) \preceq t$, and as $lub_{\preceq}(v)$ is in R, it follows that $R \sqsubseteq S$, and this completes the proof. □

With this theorem at hand, we can now give the formal definition of the implied description of a composite LO.

Definition 8 (Implied Description) *Let $o = o_1 + o_2 \ldots + o_n$ and let $D_1, .., D_n$ be the descriptions of its parts, respectively. We call* implied description *of o, denoted $IDescr(o)$, the least upper bound of $\{D_1, .., D_n\}$ in $\sqsubseteq$, i.e., $IDescr(o) = lub(\{D_1, .., D_n\}, \sqsubseteq)$*

Note that, in this definition, the descriptions of the parts are assumed to be known. In Section 4 we shall describe the mechanism by which we can associate a description to each part of a LO, prior to the computation of its implied description.

Theorem 2 suggests the following algorithm for the computation of the implied description of a set of reduced descriptions. Its proof of correctness follows directly from the theorem.

Algorithm IMPLIEDDESCRIPTION
Input: A composite LO $o = o_1 + \ldots + o_n$
The descriptions of the parts, $D_1, D_2, \ldots, D_n$
Output: The implied description $IDescr(o)$
begin
Compute $P = D_1 \times D_2 \times \ldots \times D_n$
for each tuple $L_k = [t_1^k, t_2^k, \ldots, t_n^k]$ in P, compute $T_k = lub_{\preceq}(t_1^k, t_2^k, \ldots, t_n^k)$
Let $D = \{T_1, \ldots, T_l\}$
return $reduce(D)$
end

In the algorithm, the function $lub_{\preceq}(t_1, \ldots, t_n)$ returns the least upper bound of the set of terms $t_1, \ldots, t_n$ with respect to $\preceq$. We end this section by working out a few examples illustrating how this algorithm works, referring to the taxonomy of Figure 3.1.

Example 2 *Consider the LO $o = o_1 + o_2$, composed of two parts with the following descriptions:*

$$Descr(o_1) = \{\texttt{QuickSort, Java}\} \qquad Descr(o_2) = \{\texttt{BubbleSort, C++}\}$$

In order to compute the implied description, first we compute the cross-product $P = Descr(o_1) \times Descr(o_2)$. We find the following set of tuples:

$$P = \begin{cases} L_1 = \text{¡}\texttt{QuickSort, BubbleSort}\text{¿} \\ L_2 = \text{¡}\texttt{QuickSort, C++}\text{¿} \\ L_3 = \text{¡}\texttt{Java, BubbleSort}\text{¿} \\ L_4 = \text{¡}\texttt{Java, C++}\text{¿} \end{cases}$$

Next, for each tuple L_i, $i = 1, \ldots, 4$, we compute the least upper bound of the set of terms in L_i:

1. $T_1 = \texttt{Sort}$
2. $T_2 = \texttt{Programming}$
3. $T_3 = \texttt{Programming}$
4. $T_4 = \texttt{OOL}$

We then collect together these least upper bounds to form the set

$$D = \{\texttt{Sort, Programming, OOL}\}$$

Finally we reduce D to obtain the implied description:

$$\textit{Implied Description} = \{\texttt{OOL, Sort}\}$$

In view of our discussions so far, this result can be interpreted as follows: each part of the LO concerns both, sorting and object-oriented languages.

Example 3 *Consider now the composite LO $o' = o_1 + o_3$, with the following descriptions of its parts:*

$$\textit{Descr}(o_1) = \{\texttt{QuickSort, Java}\} \qquad \textit{Descr}(o_3) = \{\texttt{BubbleSort}\}$$

Proceeding similarly, as in Example 2, we find successively:

1. *The cross-product:*
$$P = \begin{cases} L_1 = \texttt{¡QuickSort, BubbleSort¿} \\ L_2 = \texttt{¡Java, BubbleSort¿} \end{cases}$$
2. *The set of least upper bounds $D = \{\texttt{Sort, Programming}\}$*
3. *The implied description $IDescr(o) = reduce(D) = \{\texttt{Sort}\}$*

The following comments are noteworthy:

1. The term `Java` is *not* reflected in the implied description of Example 3, as it is not something that both parts share.
2. The fact that `Java` has disappeared in the implied description means no loss of information: if a user searches for documents related to java, o_1 will be in the answer and o' will not, which is consistent.
3. If we had put `Java` in the implied description of o', this would give rise to the following problem: when one searches for documents related to java, the system will return both o_1 and o'. Clearly, this answer is at the same time redundant (because o_1 is part of o'), and partially irrelevant as only a part of o' concerns java.

The same LO will generate different implied descriptions, depending on what the "companion" parts are. This is illustrated by our last example.

Example 4 *Consider the composite LO $o'' = o_1 + o_4$, with the following descriptions of its parts:*

$$\textit{Descr}(o_1) = \{\texttt{Java, QuickSort}\} \qquad \textit{Descr}(o_4) = \{\texttt{C++}\}$$

Proceeding similarly, as in Example 2, we find successively:

1. *The cross-product*
$$P = \begin{cases} L_1 = \texttt{¡Java, C++¿} \\ L_2 = \texttt{¡QuickSort, C++¿} \end{cases}$$

2. $D = \{\texttt{OOL}, \texttt{Programming}\}$

3. $reduce(D) = \{\texttt{OOL}\}$

Note that, in the two previous examples, o_1 is part of the composite LO, but each time with a different "companion part": first with o_3 in o', then with o_4 in o''. It is interesting to note that, depending on the companion part, either the "Sort-aspect" of o_1 or the "OOL-aspect" appears in the implied decription.

Chapter 4

The Syndicator

As we mentioned in the introduction, in a SeLeNe, a community of authors co-operate in the creation of LOs to be used by other authors and 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 are willing to share. Each LO resides at the local repository of its author, so all authors' repositories, collectively, can be thought of as a database of LOs spread over the network. Typically, an author wishing to create a new LO will use as components some of the LOs from his local database, and will also search for relevant LOs that reside at the local databases of other authors - provided that those other authors are willing to share them.

Authors willing to share their LOs with other authors in the network must register them with a coordinator, or *syndicator*, and authors that search for LOs matching their needs must address their queries to the syndicator. The syndicator is actually a software module at the heart of a SeLeNe. It provides a set of services, among which the following basic services:

- query evaluation
- registration of a LO
- un-registration of a LO
- description modification

In this section, we discuss these basic services and outline their interconnections.

The implementation of all the basic services relies on LO descriptions that are provided to the syndicator during LO registration. Indeed, during registration of a LO, its author is required to submit the LO identifier, say o, and a description of o that we call the *author description* of o, denoted as $ADescr(o)$. If o is atomic then the author description must be nonempty, whereas if o is composite the author description can be empty. However, if o is composite the author is also required to submit all parts of o. Based on the descriptions of the parts, the syndicator then computes (automatically) the implied description of o. Finally, to register o, the syndicator uses the author description augmented by the implied description, after removing all redundant terms. The final set of terms used for registration is what we call the *registration description* of o.

Definition 9 (Registration Description) *The Registration Description of a LO* $o = o_1 + \ldots + o_n$*, denoted* $RDescr(o)$*, is defined recursively as follows:*

- *if* o *is atomic, then* $RDescr(o) = reduce(ADescr(o))$
- *else* $RDescr(o) = reduce(ADescr(o) \cup RDescr(o_1) \cup \ldots \cup RDescr(o_n))$

One may wonder why the author description is not sufficient for LO registration. The answer is that the author of a composite LO o may not describe the parts of o in the same way as the authors of these parts have done. Let us see an example. Suppose that two LOs, o_1 and o_2, have been created by two different authors, with the following author descriptions

$$ADescr(o_1) = \{\texttt{QuickSort, Java}\} \qquad ADescr(o_2) = \{\texttt{BubbleSort, C++}\}$$

Assume now that a third author considers these LOs as examples of good programming style, and decides to use them as parts of a new, composite LO $o = o_1 + o_2$. Consequently, the author of o provides the following author description:

$$ADescr(o) = \{\texttt{GoodProgrammingStyle}\}$$

Although this author description might be accurate for the author's own purposes, the LO o still can serve to teach (or learn) java and sorting algorithms. This information will certainly be of interest to SeLeNe users searching for LOs containing material on java and sorting algorithms. Therefore, before registration, the author description should be completed, or augmented by the implied description, i.e., {OOL, Sort}, to obtain the following registration description:

$$\{\texttt{GoodProgrammingStyle, OOL, Sort}\}$$

This description contains *all* descriptions, i.e., the one given by the author of o *and* those given by the authors of its parts.

During LO registration, the registration description of o is what is actually stored by the syndicator in a repository. Conceptually, the repository can be thought of as a set of pairs constructed as follows: during registration of a LO o, the syndicator stores a pair (t, o) for each term t appearing in the registration description of o. The set of all such pairs (t, o), for all LOs that are (currently) registered is what we call the *SeLeNe Catalogue*, or simply the catalogue.

Definition 10 (Catalogue) *A* catalogue $\mathcal{C}$ *over* $(T, \preceq)$ *is a set of pairs* (t, o)*, where* t *is a term of* T *and* o *is a LO.*

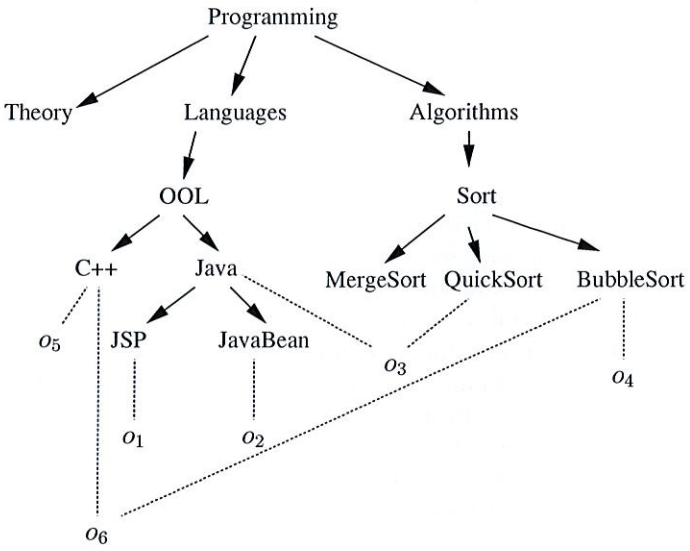

Figure 4.1: A catalogue

Figure 4.1 shows a catalogue over the taxonomy of Figure 3.1. The doted lines indicate the pairs (t, o) of the catalogue, relating terms with LOs. Roughly speaking, the catalogue is a "shopping list" in which SeLeNe users look for LOs that match their needs. As such, the catalogue is the main conceptual tool for syndicating LOs in a SeLeNe. In what follows, we discuss in more detail how the syndicator uses the catalogue to support the basic services listed earlier.

Query Language

In our approach, a query is either a single term or a boolean combination of terms, as stated in the following definition.

Definition 11 (Query Language) *A* query *over the SeLeNe catalogue is any string derived by the following grammar, where t is a term and ϵ is the empty query:*

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

Roughly speaking, the answer to a query is computed as follows. If the query is a single term, then the answer is the set of all LOs related either to t or to a term subsumed by t. If the query is not a single term then we proceed as follows. First, for each term appearing in the query, replace the term by the set of all LOs computed as explained above; then replace each boolean combinator appearing in the query by the corresponding set-theoretic operator; finally, perform the set-theoretic operations to find the answer. These intuitions are reflected in the following definition of answer, where the symbol $tail(t)$ stands for the set of all terms in the taxonomy strictly subsumed by t, i.e., $tail(t) = \{s \backslash s \prec t\}$.

Definition 12 (Query Answer) *The* answer *to a query q over a catalog C, denoted by $ans(q)$, is a set of LOs defined as follows, depending on the form of q (refer to Definition 11):*

Case 1: *q is a single term t from T, i.e., $q = t$*
$ans(t)$ = ***if*** $tail(t) = \emptyset$ ***then*** $\{o \backslash (t, o) \in C\}$ ***else*** $\bigcup\{ans(s)|s \in tail(t)\}$
Case 2: *q is a general query*
$ans(q)$ =
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 $q = q_1 \wedge \neg q_2$, $ans(q) = ans(q_1) \backslash ans(q_2)$
end
Case 3: *q is the empty query*
$ans(\epsilon) = \emptyset$

Example 5 *Consider the query $q =$* `C++` $\vee$ `Sort`*. Applying the above definition we find $ans(q) = \{o_5, o_6\} \cup \{o_3, o_4, o_6\}\{o_3, o_4, o_5, o_6\}$. Similarly, for the query $q =$* `C++` $\wedge \neg$ `BubbleSort` *we find $\{o_5\}$.*

Registration

An author wishes to make a LO available to other users in the network.

To make a LO available to other users in the network, its author must submit the following three items to the syndicator:

1. the LO identifier, say o;
2. a description (the author description of o, which must be nonempty if o is atomic);
3. the identifiers of the parts of o, if o is composite.

The syndicator then computes the registration description of o on which the actual registration will be performed. To do this, the syndicator uses the following algorithm, whose correctness is an immediate consequence of Definition 9. The algorithm takes as input the above items, and returns the updated catalogue (i.e., the old catalogue augmented by a set of pairs (t, o), one for each term t in the registration description of o).

Algorithm RDESCR
Input: The current catalogue $\mathcal{C}$, a LO o, the author description $ADescr$, the parts $\{o_1, o_2, \ldots, o_n\}$ of o
Output: The updated catalogue $\mathcal{C}$
begin
  $\mathcal{D} = \emptyset$
  for each $o_i \in parts(o)$ **do**
    if (o_i is already registered)
      Take the registration description R_i from $\mathcal{C}$
    else
      R_i = RDESC ($\mathcal{C}$, o_i, $ADescr$, $parts(o_i)$) [Recursive call to RDESC]
    endif
    $\mathcal{D} := \mathcal{D} \cup R_i$
  end for
  Let $R = reduce(IDescr(o) \cup ADescr(o))$
  for each t in R, insert the pair (t, o) in the catalogue $\mathcal{C}$
end

Note that, if o is atomic then its registration description reduces to the reduction of its author description. From a practical point of view, the following scenarios can be envisaged for providing the inputs to the algorithm RDESCR; they depend on the nature of the parts of o, as well as on whether these parts have been registered beforehand or not:

- if a part o_i of o has already been registered then its registration description is taken from the catalogue, independently of whether o_i is atomic or composite.
- else if o_i is composite and not yet registered, then its registration description is recursively computed from the registration descriptions of the parts of o_i; in this case, the full composition graph of o_i is required as input.
- else if o_i is atomic then its author description is required as input, and its registration description is the reduction of its author description.

We assume that a LO o, whether atomic or composite, can be registered only if its registration description is nonempty. This assumption is justified by the fact that search for LOs of interest by SeLeNe users is based on descriptions. As a consequence, if we allow registration of a LO with an empty description, then such a LO would be inaccessible by SeLeNe users. Therefore, the syndicator needs at least one term t, in order to insert the pair (t, o) in the catalogue, and make it accessible by SeLeNe users. This is ensured by the following constraint.

Constraint 1 (Registration Constraint) *A LO can be registered only if its registration description is nonempty*

For atomic LOs this is tantamount to requiring that the author decription be nonepmty.

Constraint 2 (Registration Constraint for Atomic LOs) *An atomic LO can be registered only if its author description is nonempty*

If the LO is composite, then the registration constraint implies that either the author description must be nonempty or the implied description must be nonempty. A sufficient condition for the implied description to be nonempty (and thus for the registration constraint to be satisfied)

is that *all* parts of the LO be registered, or (reasoning recursively) that all atomic components of the LO be registered. Indeed, if all atomic components have already been registered, then the syndicator will be able to compute a nonempty implied description, and thus a nonempty registration description, independently on whether the author descriptions of one or more components are missing. Therefore the following sufficient condition for the registration of a composite LO:

Constraint 3 (Sufficient Condition for Composite LO Registration) *If every atomic component of a composite LO is registered then the LO can be registered*

Figure 4.2 shows an example of composite LO registration. As shown in the figure, two atomic LOs, o_3 and o_4 have already been registered in the catalogue $\mathcal{C}$, and so has the composite LO o_2, whose parts are o_3 and o_4. The author descriptions of all three LOs are shown in the figure. Although the author description of o_2 is empty, its registration was possible as both its parts have nonempty author descriptions. Note that the registration descriptions of o_3 and o_4 coincide with their author descriptions (since both LOs are atomic and their author descriptions happen to be reduced). The registration description of o_2 is easily seen to be {OOL}.

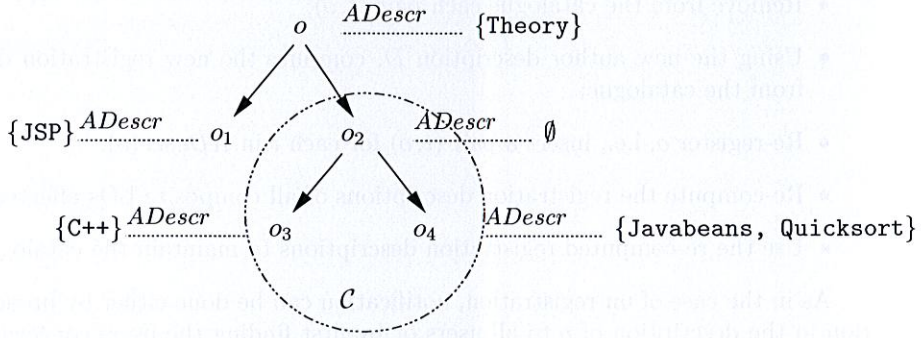

Figure 4.2: Registration description of a composite LO

Now, suppose that an author wishes to reuse o_2 (and its parts) in order to create a new LO o, composed of two parts: o_1 and o_2, where o_1 is an atomic LO from the author's local database. Suppose now that in order to register o, the author provides to the syndicator author descriptions for o and o_1, as shown in the figure. Based on these descriptions, and the registration description of o_2 (computed earlier), the syndicator will then compute the registration description of o - which is easily seen to be {OOL, Theory}. Finally, the syndicator will enter in the catalogue the two pairs (OOL, o) and (Theory, o).

Unregistration:

An author wishes to remove from the catalogue one of his registered LOs

To unregister a LO, its author must submit to the syndicator the LO identifier, say o. The syndicator then performs the following tasks:

- Notify all users using o as a component in their composite objects
- Remove from the catalogue each pair (t, o);
- Re-compute the registration descriptions of all composite LOs affected by the removal of o;
- Use the re-computed registration descriptions to maintain the catalogue.

We note that notification can be done either by broadcasting the removal of o to all users, or by first finding the users concerned and then notifying only those concerned. The first solution is

probably cheaper but may create inconvenience to those users not concerned, whereas the second avoids inconvenience but requires searching the catalogue for finding the users concerned (assuming that the syndicator keeps track of the "foreign LOs" used by each user). In any case, once notified of the (pending) unregistration of o, the users concerned have the option of first creating (in their local database) a copy of o and then proceeding to re-register all composite LOs in which o appears as a component. Otherwise, the registration description of such LOs might become empty.

Description modification:

An author wishes to modify the description of one of his registered LOs

To modify the description of a LO, its author must submit to the syndicator the LO identifier, say o, and the new author description, say D. The syndicator then performs the following tasks:

- Notify all users using o as a component in some of their composite objects;
- Remove from the catalogue each pair (t, o);
- Using the new author description D, compute the new registration description $RDescr(o)$ from the catalogue;
- Re-register o, i.e., insert a pair (t, o) for each t in $RDescr(o)$.
- Re-compute the registration descriptions of all composite LOs affected by the modification
- Use the re-computed registration descriptions to maintain the catalogue

As in the case of un-registration, notification can be done either by broadcasting the modification in the description of o to all users or by first finding the users concerned and then notifying only those concerned. However, now, there is no need for any action on the part of the user: all modified descriptions can be obtained by querying the catalogue.

There are several other services that a SeLeNe syndicator should support that lie outside the scope of the present deliverable. Some of these Services are discussed in separate deliverables.

Chapter 5

Concluding Remarks

We have presented a model for composing LOs from other simpler LOs and we have seen an algorithm for computing implied descriptions of composite LOs based on the descriptions of their parts.

In our model, a LO is represented by an identifier together with a composition graph which shows how the LO is composed from other, simpler LOs. The description of each LO is a set of terms from the SeLeNe Taxonomy. We have distinguished three kinds of description:

1. the author description, i.e., the description provided to the syndicator explicitly by the author;
2. the implied description, i.e., the description implied by the descriptions of the parts (and computed by the syndicator during registration);
3. the registration description, i.e., the description computed from the previous two descriptions and used by the syndicator to register the LO.

We have also outlined the main functionalities of the syndicator, a software module that acts as a central server, registering or unregistering sharable LOs, notifying users of changes, maintaining the SeLeNe Catalogue and answering queries by authors and/or learners.

Work in progress aims at:

- validating our model in the context of a prototype, in which the LOs are XML documents (see Appendix A as well as a Master's Thesis by B. Gueye, in French, contained in this deliverable as a separate document);
- embedding our model in the RDF Suite developed by ICS-FORTH, a SeLeNe partner (see Appendix B);
- integrating our description generating algorithms into the change propagation module developed by Birkbeck, a SeLeNe partner (see Deliverable 4.4).

The basic assumption underlying our work is the existence of a network-wide SeLeNe Taxonomy according to which LOs are described and queries are formulated. As a result, a SeLeNe functions as a super-peer network served by a central catalogue – the SeLeNe Catalogue.

Future work aims at relaxing this assumption, in order to arrive at a pure peer-to-peer network. This will be done in two steps, as follows.

First, we will assume each author, or group of authors to have their own (possibly non-standard) taxonomy, for describing their LOs locally and for formulating their queries to the syndicator. This will require the establishment of articulations, i.e., semantic mappings between each local taxonomy and the SeLeNe Taxonomy. Work in that direction will be based on previous work on mediation [12, 13, 10, 14, 11, 6].

Second, we will assume that the role of the syndicator can be played by any local taxonomy. Indeed, in principle, any local taxonomy can play the role of a syndicator for all other local taxonomies that are articulated to it.

Another line of future research concerns a 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 syndicator>
```

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.

Bibliography

[1] The ACM Computing Classification System. ACM, 1999. http://www.acm.org/class/.

[2] 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.

[3] 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.

[4] K. Keenoy, G. Papamarkos, A. Poulovassilis, D. Peterson, and G. Loizou. Self e-Learning Networks – Functionality and User Requirements. Technical report, SeLeNe Consortium, 2003. www.dcs.bbk.ac.uk/ ap/projects/selene/.

[5] 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.

[6] C. Meghini, Y. Tzitzikas, and N. Spyratos. An Abduction-based Method for Index Relaxation in Taxonomy-based Sources. In *Proc. Intl. Symp. on Mathematical Foundations of Computer Science (MFCS'03)*, Bratislava, Slovak Republic, 2003.

[7] 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.

[8] W. Nejdl, B. Worlf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. EDUTELLA: a P2P networking Infrastruture Based on RDF. In *Proc. Intl. World Wide Web Conference (WWW)*, page 604:615, 2002.

[9] SeLeNe: Self eLearning Networks. www.dcs.bbk.ac.uk/ ap/projects/selene/.

[10] N. Spyratos, V. Christophides, and Y. Tzitzikas. On Personalizing the Catalogs of Web Portals. In *Proc. Intl. FLAIRS Conf, special track on semantic Web*, Pensacola, Floride, 2002.

[11] Y. Tzitzikas, A. Analyti, N. Spyratos, and P. Constantopoulos. An Algebra for Specifying Compound Terms in Faceted Taxonomies. In *Proc. European-Japanese Conference on Information Modelling and Knowledge Base*, Kitakyushu, Japan, 2003.

[12] 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.

[13] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. Query Evaluation for Mediators over Web Catalogs. In *Proc. Intl. Conf. on Information and Communication Technologies and Programming*, Primorsko, Bulgaria, 2002.

[14] Y. Tzitzikas, N. Spyratos, P. Constantopoulos, and A. Analyti. Extended Faceted Ontologies (short paper). In *Proc. Advanced Information Systems Engineering*, Toronto, Canada, 2002.

[15] N. Walsh and Leonard Muellner. *DocBook, the definitive guide.* O'Reilly, 1999.

[16] The XPath language recommandation (1.0). World Wide Web Consortium, 1999. http://www.w3.org/TR/xpath.

Appendix A

A Case Study

Authors: B. Gueye, Ph. Rigaux, N. Spyratos

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 A.1. 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 [15]. 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 form the network terminology.

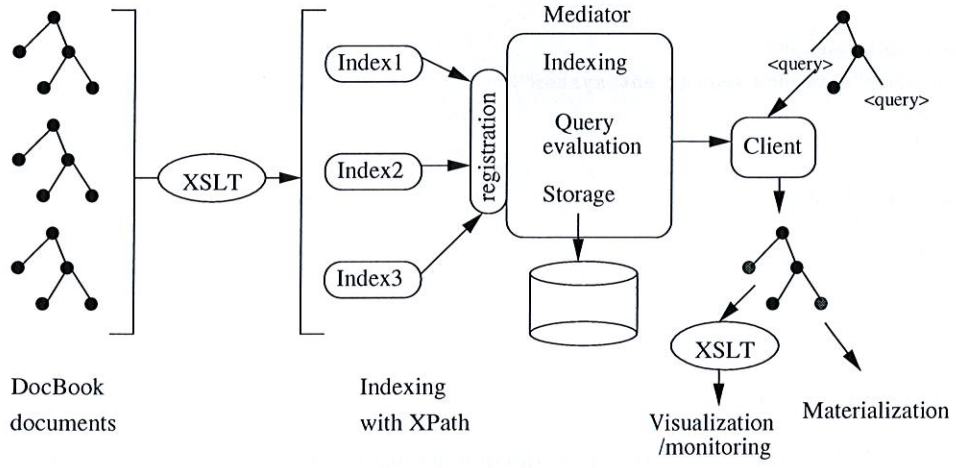

Figure A.1: Overview of the system's architecture

From these documents, a program (written with the XML transformation language, XSLT) produces the description for each document. Descriptions are sent to the syndicator which stores them in a repository, creates description 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 syndicator 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.

A.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 syndicator and accessible through a Web interface to users and authors who want to pick up terms for, respectively, describing their documents or expressing queries. Here is a small part of this terminology.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<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>

    <term name="SQL">
          (...)
    </term>
  </term>
</term>
```

A.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 identifiy 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 a description must be associated. The inference mechanism described in the model is then used to create the descriptions 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">
        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 A.2.

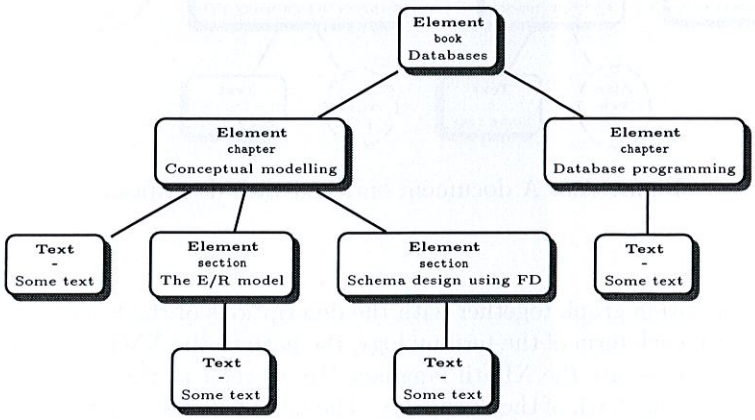

Figure A.2: The hierarchical structure of the document

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 syndicator, is to describe 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 E/R model" term="E/R">
        Some text ...
       </section>
      <section title="Schema design"
                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 A.3. The document represented in this figure contains descriptions for *all* the elements, at any level. The descriptions for `<chapter>` and `<book>` elements have been derived automatically from the descriptions of the leaves in the way explained earlier.

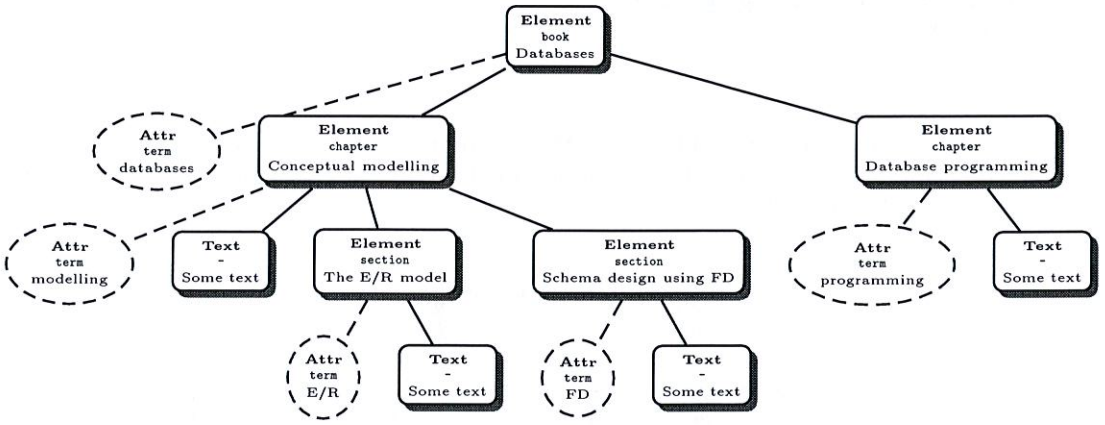

Figure A.3: A document enriched with descriptions

Finally the composition graph together with the descriptions of the leaves is sent to the syndicator 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 [16] to refer to these subtrees, and complete XPath expressions with the URL of the document. The table below shows the information handled by the syndicator to refer to the nodes of the document used so far.

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]`

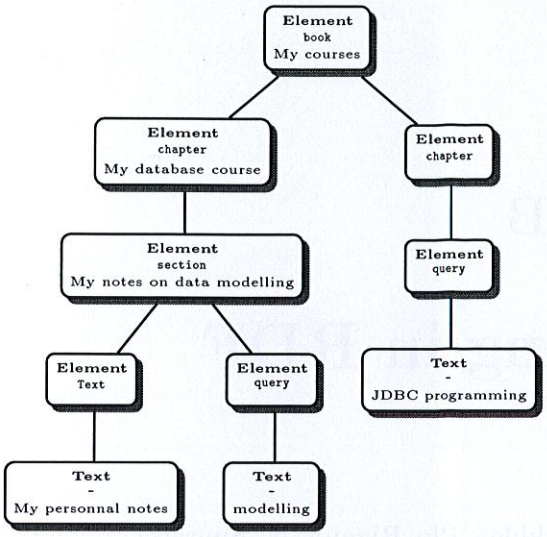

Figure A.4: A derived document

Finally let us illustrate how one can create composite documents by inserting *queries*. The example of Figure A.4 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 syndicator 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).

Appendix B

Embedding in RDF

Authors: V. Christophides, Ph. Rigaux, N. Spyratos

In this appendix we describe briefly integration activities in progress between LRI and ICS-FORTH, aiming at embedding the model proposed by LRI in Deliverable 4.1 into the RDF Suite developed by ICS-FORTH [2].

In a nutshell, Deliverable 4.1 proposes a model for composing LOs from other simpler LOs and an algorithm for generating descriptions of composite LOs based on the descriptions of their parts.

A LO is represented by an identifier together with a composition graph which shows the structure of the LO. The description of a LO is seen as a set of terms from a network-wide taxonomy, the SeLeNe Taxonomy, and consists of two parts, a part provided by the author and a part implied by the LO structure. A central server, the syndicator, maintains the SeLeNe Catalogue and answers queries by authors and/or learners.

Below, we summarize the guidelines that we have agreed upon, for the embedding of this model into the RDF Suite. These guidelines have already been used while conducting the case study (see Appendix A).

1. A SeLeNe Taxonomy will be represented as a RDF scheme:
   - each term of the SeLeNe Taxonomy will be represented as a class name;
   - each subsumption relationship as a ISA link between the corresponding classes.
2. The SeLeNe Catalogue will be represented as a RDF database:
   - each LO will be represented as a RDF resource;
   - each LO (resource) will be classified under each of the terms (class names) appearing in its description.
3. RQL facilities will be used for browsing, querying, and LO composition.

In this respect, we note that RQL facilities include browsing and querying facilities that cover the SeLeNe requirements identified so far, as well as primitives for expressing that a set of resources constitute parts of a given resource. This last feature is essential for expressing that the LOs appearing in the answer of a query to the syndicator are parts of a composite LO under construction.

Moreover, the RQL facilities provide graphic interfaces for user-friendly interaction.

RAPPORTS INTERNES AU LRI - ANNEE 2003

N°	Nom	Titre	Nbre de pages	Date parution
1345	FLANDRIN E LI H WEI B	A SUFFICIENT CONDITION FOR PANCYCLABILITY OF GRAPHS	16 PAGES	01/2003
1346	BARTH D BERTHOME P LAFOREST C VIAL S	SOME EULERIAN PARAMETERS ABOUT PERFORMANCES OF A CONVERGENCE ROUTING IN A 2D-MESH NETWORK	30 PAGES	01/2003
1347	FLANDRIN E LI H MARCZYK A WOZNIAK M	A CHVATAL-ERDOS TYPE CONDITION FOR PANCYCLABILITY	12 PAGES	01/2003
1348	AMAR D FLANDRIN E GANCARZEWICZ G WOJDA A P	BIPARTITE GRAPHS WITH EVERY MATCHING IN A CYCLE	26 PAGES	01/2003
1349	FRAIGNIAUD P GAURON P	THE CONTENT-ADDRESSABLE NETWORK D2B	26 PAGES	01/2003
1350	FAIK T SACLE J F	SOME b-CONTINUOUS CLASSES OF GRAPH	14 PAGES	01/2003
1351	FAVARON O HENNING M A	TOTAL DOMINATION IN CLAW-FREE GRAPHS WITH MINIMUM DEGREE TWO	14 PAGES	01/2003
1352	HU Z LI H	WEAK CYCLE PARTITION INVOLVING DEGREE SUM CONDITIONS	14 PAGES	02/2003
1353	JOHNEN C TIXEUIL S	ROUTE PRESERVING STABILIZATION	28 PAGES	03/2003
1354	PETITJEAN E	DESIGNING TIMED TEST CASES FROM REGION GRAPHS	14 PAGES	03/2003
1355	BERTHOME P DIALLO M FERREIRA A	GENERALIZED PARAMETRIC MULTI-TERMINAL FLOW PROBLEM	18 PAGES	03/2003
1356	FAVARON O HENNING M A	PAIRED DOMINATION IN CLAW-FREE CUBIC GRAPHS	16 PAGES	03/2003
1357	JOHNEN C PETIT F TIXEUIL S	AUTO-STABILISATION ET PROTOCOLES RESEAU	26 PAGES	03/2003
1358	FRANOVA M	LA "FOLIE" DE BRUNELLESCHI ET LA CONCEPTION DES SYSTEMES COMPLEXES	26 PAGES	04/2003
1359	HERAULT T LASSAIGNE R MAGNIETTE F PEYRONNET S	APPROXIMATE PROBABILISTIC MODEL CHECKING	18 PAGES	01/2003
1360	HU Z LI H	A NOTE ON ORE CONDITION AND CYCLE STRUCTURE	10 PAGES	04/2003
1361	DELAET S DUCOURTHIAL B TIXEUIL S	SELF-STABILIZATION WITH r-OPERATORS IN UNRELIABLE DIRECTED NETWORKS	24 PAGES	04/2003
1362	YAO J Y	RAPPORT SCIENTIFIQUE PRESENTE POUR L'OBTENTION D'UNE HABILITATION A DIRIGER DES RECHERCHES	72 PAGES	07/2003
1363	ROUSSEL N EVANS H HANSEN H	MIRRORSPACE : USING PROXIMITY AS AN INTERFACE TO VIDEO-MEDIATED COMMUNICATION	10 PAGES	07/2003

RAPPORTS INTERNES AU LRI - ANNEE 2003

N°	Nom	Titre	Nbre de pages	Date parution
1364	GOURAUD S D	GENERATION DE TESTS A L'AIDE D'OUTILS COMBINATOIRES : PREMIERS RESULTATS EXPERIMENTAUX	24 PAGES	07/2003
1365	BADIS H AL AGHA K	DISTRIBUTED ALGORITHMS FOR SINGLE AND MULTIPLE-METRIC LINK STATE QoS ROUTING	22 PAGES	07/2003
1366	FILLIATRE J C	WHY : A MULTI-LANGUAGE MULTI-PROVER VERIFICATION TOOL	20 PAGES	09/2003
1367	FILLIATRE J C	A THEORY OF MONADS PARAMETERIZED BY EFFECTS	18 PAGES	09/2003
1368	FILLIATRE J C	HASH CONSING IN AN ML FRAMEWORK	14 PAGES	09/2003
1369	FILLIATRE J C	DESIGN OF A PROOF ASSISTANT : COQ VERSION 7	16 PAGES	09/2003
1370	HERMAN T TIXEUIL S	A DISTRIBUTED TDMA SLOT ASSIGNMENT ALGORITHM FOR WIRELESS SENSOR NETWORKS	32 PAGES	09/2003