SeLeNe - Self E-Learning Networks
IST-2001-39045

WP4 Deliverable 4.1

Generation and Syndication of Learning Object Metadata

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. We assume that each LO resides at the local repository of its provider, so all providers' repositories, collectively, can be thought of as a database of LOs spread over the network. Providers willing to share their LOs with other providers in the network must register them through the appropriate SeLeNe services (see deliverable 3). In the present deliverable we focus on the part of the registering, un-registering and querying services that rely on LO content description. In this respect, we distinguish three kinds of description: the provider description, the implied description and the registration description.

Each of these descriptions is actually a set of terms from a taxonomy to which all providers adhere. During registration of a LO, its provider is required to submit a description that we call the *provider description*. If the LO is atomic then the

provider description must be nonempty, whereas if the LO is composite the provider description can be empty. However, if the LO is composite the provider is also required to submit all *parts* of the LO being registered (i.e., all LOs that constitute the LO being registered). Based on the descriptions of these parts, the registration service then automatically determines a description that we call the *implied* description of the composite LO. The LO registration involves the provider 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*. In this context, the main contributions of the present deliverable are:

1. providing appropriate definitions of LO descriptions;
2. providing an algorithm for the automatic computation of implied descriptions;
3. defining the procedures which must be supported by the SeLeNe services to maintain the correct LO descriptions through registrations and un-registrations;
4. defining the procedures which must be supported by the SeLeNe services to syndicate LO metadata when the providers use their own, local taxonomies;
5. providing an articulation mechanism between distinct taxonomies for querying purposes.

We have tested our results in a case study that illustrates some features and functionalities of a SeLeNe in which the LOs are XML documents (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 concerns two activities:

1. Integration of our algorithms into the change propagation module developed by Birkbeck (see Deliverable 4.4 [13]).
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.

LRI, France

The SeLeNe Project

Life-long learning and the knowledge economy have brought about the need to support a broad and diverse community of consumers throughout their lifetimes. These consumers are geographically distributed and highly heterogeneous in their educational backgrounds and learning needs. The number of learning resources available on the Web is continuously increasing, thus indicating the Web's enormous potential as a significant resource of educational material both for consumers and instructors.

The SeLeNe Project aims to elaborate new educational metaphors and tools in order to facilitate the formation of learning communities that require world-wide discovery and assimilation of knowledge. To realize this vision, SeLeNe is relying on semantic metadata describing educational material. SeLeNe offers advanced services for the discovery, sharing, and collaborative creation of learning resources, facilitating a syndicated and personalised access to such resources. These resources may be seen as the modern equivalent of textbooks, comprising rich composition structures, "how to read" prerequisite paths, subject indices, and detailed learning objectives.

The SeLeNe Project (IST-2001-39045) is a one-year Accompanying Measure funded by EU FP5, running from 1st November 2002 to 31st October 2003. The project falls into action line V.1.9 CPA9 of the IST 2002 Work Programme, and is contributing to the objectives of Information and Knowledge Grids by allowing access to widespread information and knowledge, with e-Learning as the test-bed application. The project is conducting a feasibility study of using Semantic Web technology for syndicating knowledge-intensive resources (such as learning objects) and for creating personalized views over such a Knowledge Grid.

Executive Summary

This deliverable (4.1) is part of the SeLeNe Workpackage 4 on Syndication and Personalization of Educational Resources. Workpackage 4 has two main objectives:

- To investigate techniques for syndication and personalization of distributed, autonomous RDF description bases.
- To design language primitives for defining user views over distributed RDF description bases and for deriving composite learning objects' descriptions from those of their constituent learning objects.

Accessing RDF description bases in SeLeNe raises two basic technical challenges: (1) flexible mediation of the different RDF schemas employed by the RDF description bases, and (2) personalization of learning objects' descriptions and schemas according to the educational needs and interests of learning objects' providers (i.e., instructors) and consumers. (i.e., learners).

Concerning problem (1), the IEEE LOM has effectively achieved the integration of the various educational metadata standards, as is reported in Deliverable 2.1 [16] of Workpackage 2. Thus, in the context of a SeLeNe, we are assuming that metadata about learning

objects are represented using an RDF/S binding of the IEEE LOM. However, fine-grained descriptions expressed in domain or topic-specific taxonomies may also be made available by instructors. Hence, this workpackage is investigating a flexible articulation of different domain/topic-specific taxonomies which can be used for e-learning, as well as the automatic generation of semantic descriptions for composite learning objects using the descriptions of their constituent learning objects. The taxonomies used for this purpose and the resulting descriptions can easily represented RDF/S. This work is reported in Deliverable 4.1.

Concerning problem (2) above, two major issues are involved: (a) specification by learners of their educational needs, and (b) adaptation of learning objects to these needs. Issue (a) requires the representation of educational needs in a "learner profile" using the e-learning schemas as well as the domain or topic-specific taxonomies available in SeLeNe. It also requires unstructured, keyword-based querying facilities, which can be translated automatically into the structured RDF/S queries supported by the SeLeNe system. The result of these unstructured queries may be returned in a special form of composite learning object called "trails". To address issue (b) we need methods for dynamically adapting learning material to the preferences of a learner. This requires ranking of query results by matching the descriptions of the returned learning objects against the a learner's profile. These issues are discussed in Deliverable 4.2.

Specifying educational needs or describing educational material according to personalized e-learning RDF/S schemas (for both learners and instructors) requires formalisms for defining declarative views over learning object descriptions and schemas, and this work is reported in Deliverable 4.3. This deliverable also discusses the structured RDF/S querying facilities supported by SeLeNe.

Also needed are techniques for detecting changes in learning objects' descriptions or users' personal profiles, and for notifying users who have subscribed to be notified of such changes. Techniques for the provision of this kind of reactive functionality over RDF descriptions of learning objects and users are reported in Deliverable 4.4.

Revision Information

Revision Date	Version	Changes
September 11, 2003	0.1	First Draft Proposal
October 24, 2003	1.0	First version
November, 5, 2003	1.1	Minor modifications
November, 21, 2003	2.0	Second version, including a revision of Section 4 according to the grid-based architecture of deliverable 3.
November, 26, 2003	2.1	Minor improvements of the previous revision.
January 7, 2004	2.2	New section on articulations with local taxonomies

Table of Contents

Contents

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 [14].

In a SeLeNe, a community of LO *providers* co-operate in the creation of LOs to be used also by other providers and by a community of learners. Each provider is 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 providers have created and are willing to share [6]. In a nutshell, our approach can be described as follows.

We distinguish LOs into *atomic* and *composite*. Intuitively, an atomic LO is any individual piece of learning material (text, image, sound, etc.) that can be identified uniquely and cannot be decomposed further; its nature and granularity are entirely up to its provider. 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 provider, so all providers' repositories, collectively, can be thought of as a database of LOs spread over the network. Typically, a provider 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.

Providers willing to share their LOs with other providers in the network must register them through the appropriate SeLeNe services (see deliverable 3 [4]). In the present deliverable we focus on the part of the registering, un-registering and querying processes that rely on LO content description. In this respect, we distinguish three kinds of description: the provider description, the implied description and the registration description.

Such descriptions are actually sets of terms from a controlled vocabulary, or *taxonomy*, to which all providers adhere. The well known ACM Computing Classification System [1] is an example of such a taxonomy.

During registration of a LO, its provider 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 *provider description* of o; if o is atomic then the provider description must be nonempty, whereas if o is composite then the provider description can be empty.

3. If o is composite, then registration requires, additionally, the submission of all parts of o; using the descriptions of these parts, the registration process 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 registration service uses the provider 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 Update Service provides an interface that maintains the part of the RDF repository (see deliverable 3 [4] and 4.3 [8]) concerning the descriptions of registered LOs, called the *catalogue* hereafter. During registration of a LO with identifier o, these services insert in the catalogue a pair (t, o), for each term t in the registration description of o.

Providers and consumers searching for LOs that match their needs will rely on the catalogue for their subject-based 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 procedures which must be supported by the SeLeNe services to maintain the correct LO descriptions through registrations and un-registrations;
4. defining the procedures which must be supported by the SeLeNe services to syndicate LO metadata when the providers use their own, local taxonomies;
5. providing an articulation mechanism between distinct taxonomies for querying purposes.

This report proposes *generic* solutions to the above issues, i.e., solutions that are valid independently of questions concerning network configuration and the organization of services over this network. We refer to deliverable 3 [4] for a presentation of both the architectural framework and the services that support the functionalities presented in the following.

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 taxonomy acting as a mediator (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 concerns two activities:

1. Integration of our algorithms into the change propagation module developed by Birkbeck (see Deliverable 4.4 [13]).
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 provider, 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.

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 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 providers 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 provider. 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 1.

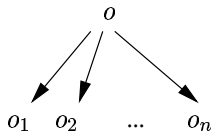

Figure 1: A LO and its parts

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

Definition 2 (Components of a 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:*

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

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.

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 providers adhere. A taxonomy consists of a set of terms together with a subsumption relation between terms. An example of a 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.

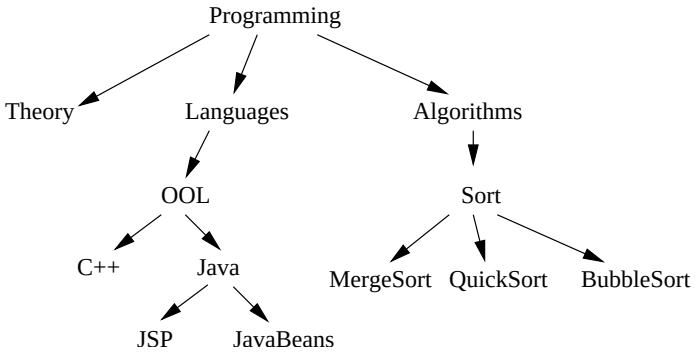

Figure 2: A taxonomy

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 t to term s iff t subsumes s . Figure 2 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 2, 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 providers 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 use the terms `QuickSort` and `Java` to describe its content, and the set of terms {`QuickSort, Java`} is then 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 `QuickSort` 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 LO 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 provider 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 we would like the implied description of $o = o_1 + o_2$ to be $\{\texttt{Sort, OOL}\}$.

We shall come back to this example after the formal definition of implied description. Actually, the main 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 2, `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`} because:

- {`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 either `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. To gain some insight into this ordering, let us see an example. Referring to Figure 2, consider a LO with two parts having 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 in *all* parts (and `Theory` is not common in the two parts of our example).

Clearly, $\sqsubseteq$ is a reflexive and transitive relation over descriptions, but *not* antisymmetric, as the following example shows. Consider $D_1 = \{\texttt{OOL, Java, Sort}\}$ and $D_2 = \{\texttt{Java, Sort, Algorithms}\}$. 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 collecting all v_i's together, i.e., $v =< v_1, v_2, ..., v_n >$. 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 $o = o_1 + o_2 \ldots + o_n$ be a LO 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 $D = \{T_1, \ldots, T_l\}$, where l is the cardinality of P
return $reduce(D)$
end

In this 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 2.

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:

$$P = \begin{cases} L_1 =< \textit{QuickSort, BubbleSort} > \\ L_2 =< \textit{QuickSort, C++} > \\ L_3 =< \textit{Java, BubbleSort} > \\ L_4 =< \textit{Java, C++} > \end{cases}$$

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. $T_1 =$ *Sort*
2. $T_2 =$ *Programming*
3. $T_3 =$ *Programming*
4. $T_4 =$ *OOL*

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

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

Finally we reduce D to obtain the implied description:

$$\textit{Idescr(o)} = \textit{reduce(D)} = \{\textit{OOL, Sort}\}$$

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 the LO $o' = o_1 + o_3$, with the following descriptions of its parts:*

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

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

1. *The cross-product:*

$$P = \begin{cases} L_1 =< \textit{QuickSort, BubbleSort} > \\ L_2 =< \textit{Java, BubbleSort} > \end{cases}$$

2. *The set of least upper bounds $D = \{$Sort, Programming$\}$*
3. *The implied description $IDescr(o) = reduce(D) = \{$Sort$\}$*

Regarding Example 3, 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 both, redundant (as o_1 is part of o') and partially irrelevant (as only part of o' concerns Java).

We now give a last example that illustrates an important aspect of implied Descriptions: the same LO will generate different implied descriptions, depending on its "companion" parts in a composite LO.

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

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

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

1. *The cross-product*
$$P = \begin{cases} L_1 = < \textit{Java, C++} > \\ L_2 = < \textit{QuickSort, C++} > \end{cases}$$

2. *The set of least upper bounds $D = \{\textit{OOL, Programming}\}$*

3. *The implied description $IDescr(o'') = reduce(D) = \{\textit{OOL}\}$*

Note that o_1 participates in the creation of the composite LO in *both* of the last two examples, but each time with a different "companion part": in Example 3, o_1 has o_3 as companion part in creating o', whereas in Example 4 o_1 has o_4 as companion part in creating o''. The interesting point to note here is that, depending on the companion part, a *different* aspect of o_1 appears in the implied decription of the created LO: in the implied decription of o' only the "Sort" aspect of o_1 appears, whereas in the implied decription of o'', only the "OOL" aspect of o_1 appears.

4 Descriptions Management in a SeLeNe

As we mentioned in the introduction, in a SeLeNe, a community of providers co-operate in the creation of LOs to be used by other providers and by a community of learners. Each provider is 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 providers have created and are willing to share. Each LO resides at the local repository of its provider, so all providers' repositories, collectively, can be thought of as a database of LOs spread over the network. Typically, a provider 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 providers - assuming of course that those other providers are willing to share their LOs.

Providers willing to share some or all of their LOs with other providers in the network must register them, and providers that search for LOs matching their needs must be able to issue *term-based queries*, in which the terms are from the taxonomy in use. We refer to such queries as term-based to distinguish from other kinds of queries (see deliverable 2.2 [6]). Each of these functionalities involves the coordination of one or several services from the service-based framework presented in deliverable 3 [4]. In the following we give a high-level description of the term-based querying and LO registration services, *only* with respect to the maintenance of LO descriptions. In particular, we consider:

- support for term-based querying
- registration of a LO description
- un-registration of a LO description
- description modification
- syndication of LO descriptions

Note that these functionalities are all considered *only* from the point of view of LO description management. The registration and search of a LO involve many other eLearning metadata which are ignored here. The reader is referred to deliverable 2.1 [16] in this project. It must also be underlined that the algorithms presented thereafter are architecture-neutral and will be implemented and incorporated as part of the services described in deliverable 3 [4].

During registration of a LO, its provider is required to submit the LO identifier, say o, and a description of o that we call the *provider description* of o, denoted as $PDescr(o)$. If o is atomic then the provider description must be nonempty, whereas if o is composite the provider description can be empty. However, if o is composite the provider is also required to submit all parts of o. Based on the descriptions of the parts, the registration process then computes (automatically) the implied description of o. Finally, to register o, the registration process uses the provider 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* $RDescr(o) = reduce(PDescr(o))$
- *else* $RDescr(o) = reduce(PDescr(o) \cup IDescr(RDescr(o_1) \cup \ldots \cup RDescr(o_n)))$

One may wonder why the provider description is not sufficient for LO registration. The answer is that the provider of a composite LO o may not describe the parts of o in the same way as the providers 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 providers, with the following provider descriptions

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

Assume now that a third provider 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 provider of o submits the following provider description:

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

Although this provider description might be accurate for the provider'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 provider 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 provider of o *and* those given by the providers of its parts.

During LO registration, the registration description of o is what is actually stored in the RDF repository. Conceptually, this part of the repository can be thought of as a set of pairs constructed as follows: during registration of a LO o, one 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 3 shows a catalogue over the taxonomy of Figure 2. The dotted 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. In what follows, we discuss in more detail how to use the catalogue to support the search of LOs through term-based queries, and the maintenance of LO descriptions.

Term-based Query Language

We present a simple query language which specifically addresses the descriptions of LOs. Queries are "term-based" in the sense that they rely on terms chosen from the common taxonomy. Term-based queries can be submitted as annotated keyword queries, as described

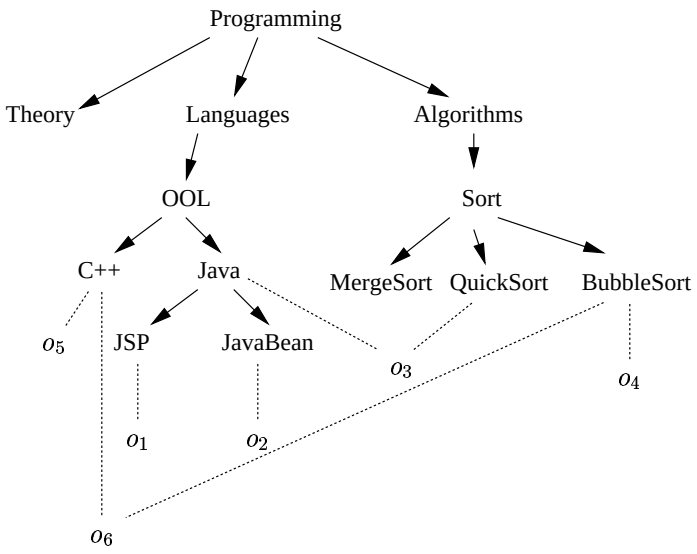

Figure 3: A catalogue

by Deliverable 2.2 [6], and can easily be incorporated in RQL queries over the RDF global medatada, thanks to a simple translation similar to that presented in Deliverable 4.2 [5]. A term-based query is either a single term or a boolean combination of terms, as stated in the following definition.

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

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

As all queries considered in this deliverable are term-based queries, hereafter, we shall write "query" to stand for "term-based query". 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.

Definition 12 (Query Answer) *The* answer *to a query q over a catalogue $\mathcal{C}$, 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., $q = t$*

$ans(t) = \bigcup\{I(s) | s \preceq t\}$, *where* $I(s) = \{o | (s, o) \in \mathcal{C}\}$

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* $ans(q) = \{o_5\}$.

Registration of LO descriptions

A provider wishes to make a LO available to other users in the network.

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

1. the LO identifier, say o;
2. a description (the provider description of o, which must be nonempty for each atomic component of o, including o itself if it is atomic);
3. the identifiers of the parts of o, if o is composite.

The registration process then computes the registration description of o on which the actual registration will be performed. To do this, it uses the following algorithm, whose correctness is an immediate consequence of Definition 9. The algorithm takes as input the above items, and updates the catalogue. The updated catalogue is 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: A LO o, the provider description $Pdescr(o)$, the parts $\{o_1, o_2, \ldots, o_n\}$ of o
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 the catalogue

else
R_i = RDESC (o_i, $Pdescr(o_i)$, $parts(o_i)$) [Recursive call to RDESC]
endif
$\mathcal{D} := \mathcal{D} \cup R_i$
end for
Let $R = reduce(IDescr(o) \cup PDescr(o))$
for each t in R, insert the pair (t, o) in the catalogue
end

Note that the computations performed by this algorithm 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 provider description is required as input, and its registration description is the reduction of its provider 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 term-based 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, one needs at least one term t, in order to insert the pair (t, o) in the catalogue, and make o 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 provider decription be nonempty.

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

If the LO is composite, then the registration constraint implies that either the provider 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 registration process will be able to compute a nonempty implied description, and thus a nonempty registration description, independently on whether the provider 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 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 provider descriptions of all three LOs are shown in the figure. Although the provider description of o_2 is empty, its registration was possible as both its parts have nonempty provider descriptions. Note that the registration descriptions of o_3 and o_4 coincide with their provider descriptions (since both LOs are atomic and their provider descriptions happen to be reduced). The registration description of o_2 is easily seen to be {`OOL`}.

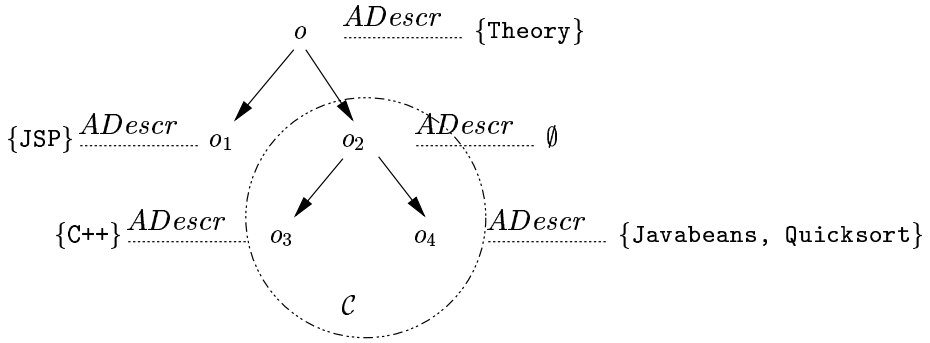

Figure 4: Registration description of a composite LO

Now, suppose that a provider 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 provider's local database. Suppose now that in order to register o, the provider provides descriptions for o and o_1, as shown in the figure. Based on these descriptions, and the registration description of o_2 (computed earlier), one will then compute the registration description of o - which is easily seen to be {`OOL, Theory`}. Finally, using this registration description, one will enter in the catalogue the two pairs (`OOL`, o) and (`Theory`, o).

Unregistration:

A provider wishes to remove from the catalogue one of his registered LOs

To unregister a LO, its provider must submit the LO identifier, say o. The SeLeNe services (see deliverable 3 [4]) are then required to perform 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 SeLeNe keeps track of the "foreign LOs" used by each user). The ECA Services presented in deliverable 4.4 [13] will be able to deal with notifying only those users affected. 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:

A provider wishes to modify the description of one of his registered LOs

To modify the description of a LO, its provider must submit the LO identifier, say o, and the new provider description, say D. The SeLeNe services are then required to perform 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 provider description D, compute the new registration description $RDescr(o)$ from the catalogue;
- Re-compute the registration descriptions of all composite LOs affected by the modification.

As in the case of un-registration, notification can be handled by the ECA rules (deliverable 4.4 [13]). 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.

Syndication of LO descriptions

The basic assumption underlying LO descriptions management has been so far the existence of a network-wide SeLeNe Taxonomy, i.e., a central taxonomy according to which LOs are described and term-based queries are formulated.

However, we can relax this assumption, i.e., we can let each provider, or group of providers have their own (possibly non-standard) taxonomy in order to describe their LOs locally and to formulate term-based queries to their local repositories. In such a scenario, we would like each provider to be able to register his LOs with the SeLeNe registration service and to query the central taxonomy based on the terms of his own local taxonomy.

Clearly, in order to allow this functionality, we need to translate the provider's LO descriptions and queries, into descriptions and queries over the central taxonomy. This requires the establishment of *articulations*, i.e., semantic mappings between the terms of each local taxonomy and those of the central taxonomy.

Definition 13 (Articulation) *Let S and T be two taxonomies. An* articulation *from S to T is a set A of pairs of the form $(s, A(s))$, such that:*

1. *s is a term of S;*
2. *$A(s)$ is a set of terms from T;*
3. *every term s of S appears at most once in A.*

In other words, an articulation from S to T is a partial mapping from S to the powerset of T. The use of articulations in translating LO descriptions and queries over a local taxonomy to descriptions and queries over the central taxonomy of a SeLeNe is described in the following definition.

Definition 14 (Translation) *In a SeLeNe, let S be a local taxonomy, T the cenral taxonomy and A an articulation from S to T. Then LO descriptions and queries over S are translated to descriptions and queries over T as follows:*

1. ***Description***
   Let D be a LO description over S. The translation of D over T, denoted $A(D)$, is defined by: $A(D) = \cup\{A(s)/(s, A(s)) \in A\}$
2. ***Query***
   Let Q be a query over S. The translation of Q over T, denoted $A(Q)$, is obtained by replacing each term s of Q appearing in A by the conjunction of all terms in $A(s)$.

For example, suppose that a provider uses (locally) a taxonomy S articulated to the central taxonomy T of a SeLeNe by the following articulation:

$$A = \{\texttt{(LOO, C++, Java), (Tri, QuickSort, MergeSort), ...}\}$$

Such an articulation might simply reflect the facts that

- this provider's local taxonomy is in French (in French, `LOO` stands for "Langages Orienté-Objet" and means "object-oriented languages", and "Tri" means "Sort");
- the local taxonomy describes LOs at a granularity coarser than that of the central taxonomy T.

Suppose now that this provider wishes to register an atomic LO and, to this effect, he submits the LO identifier and the following LO description:

$$D = \{\texttt{LOO, Tri}\}$$

Then following the previous definition, the translation $A(D)$ is obtained by taking the union of the images of the terms `LOO` and `Tri` under A:

$$A(D) = \{\texttt{C++, Java, QuickSort, MergeSort}\}$$

Next, suppose that this provider issues the following query to the mediator: $Q =$`LOO` $\wedge \neg$ `Tri`. Following the previous definition, the translation $A(Q)$ is obtained by replacing (in Q) `LOO` by the conjunction `C++` $\wedge$ `Java` and `Tri` by the conjunction `QuickSort` $\wedge$ `MergeSort`:

$$A(Q) = \{\texttt{(C++} \wedge \texttt{Java)} \wedge \neg \texttt{(QuickSort} \wedge \texttt{MergeSort)}\}$$

In a SeLeNe, the articulations between local taxonomies and the central taxonomy are stored and used whenever a provider uses his local taxonomy to register a LO or to formulate a query.

Indeed, in a SeLeNe, a provider is able to submit provider descriptions and queries based on his local taxonomy and, subsequently, the SeLeNe registration and querying services translate them into descriptions and queries over the central taxonomy (by accessing the stored articulations).

Once this translation has been performed, querying the central taxonomy, registering a LO description, un-registering a LO description, and modifying a LO description can proceed as described earlier, in the previous subsections. The SeLeNe service that stores and manages articulations (and translations) from local taxonomies to the central taxonomy is called the *Syndication Service*. This service is called upon whenever the need arises to:

1. store/delete an articulation, when a new user (or group of users) connects/disconnects to/from a SeLeNe;

2. translate a LO description when a user registers/modifies/unregisters a LO;

3. translate a query when a user formulates one based on his local taxonomy.

A more detailed discussion on the syndication service and its interactions with other SeLeNe services can be found in Deliverable 3 [4].

We end this section with two remarks concerning articulations and their role in the functioning of a SeLeNe. The first remark concerns the form of articulation introduced here (i.e., local taxonomy terms mapped to sets of central taxonomy terms), which is certainly not the most general one. Articulations can actually have more complex forms, leading though to more complex translation schemes (see [19, 20]) and the ICS-FORTH Semantic Web Integration Middleware (SWIM)[1]. We believe that the form of articulation introduced here is quite flexible without being too complex to handle.

The second remark concerns communication in a SeLeNe through articulations. Roughly, we can distinguish three basic scenarios, that we call *Centralized*, *Local-to-Central* and *Autonomic*.

[1] See `ftp://ftp.research.microsoft.com/pub/debull/A03dec/issue1.htm`.

1. *Centralized*: There is a central taxonomy and each user (provider and/or learner) uses that taxonomy - in other words, there is no local taxonomy. Put it differently, the local taxonomies are each identical to the central taxonomy (and their articulations to the central taxonomy are identities).

2. *Local-to-Central*: There is a central taxonomy but each user may use his own local taxonomy. Each local taxonomy, in turn, is articulated to the central taxonomy.

3. *Autonomic*: There is no central taxonomy. Each user or group of users has his own taxonomy and is free to articulate that taxonomy to one or more other taxonomies (the resulting network of articulated taxonomies is also called a (pure) *peer-to-peer network*).

In this report, we have studied in some detail the first two scenarios (centralized and local-to-central). Some preliminary work on the third scenario (autonomic or peer-to-peer) is reported in ER'03 [18]. Also, recent work at Birkbeck presents a new approach to constructing articulation mappings between heterogeneous schemas in P2P environments [9]. In this approach, mappings consist of a sequence of bidirectional schema transformations. These sequences are easily extensible, making the approach well-suited to the needs to P2P data integration, where the set of schemas managed by peers, and the mappings between schemas, may change at any time.

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. Each LO is also associated with three kinds of description, each of them being a set of terms coming from the SeLeNe taxonomy:

1. The *provider description*, i.e., a set of terms provided explicitly by the provider and coming either from the SeLeNe taxonomy (in the Centralized scenario) or from the provider's local taxonomy (in the local-to-central scenario); in the latter case, the provider description is translated into a set of terms from the SeLeNe taxonomy based on the stored articulations.

2. The *implied description*, i.e., a set of terms coming from the SeLeNe taxonomy and implied by the descriptions of the parts; the implied description is computed during registration.

3. The *registration description*, i.e., a set of terms coming from the SeLeNe taxonomy and computed from the previous two descriptions; the registration description is used to register the LO.

We have also outlined the algorithm which must be triggered by the appropriate SeLeNe services for registering, modifying or unregistering LOs descriptions, notifying users of changes, maintaining the SeLeNe Catalogue and answering term-based queries by providers and/or consumers.

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 [13]).

Our results were presented assuming a centralized scenario, i.e., assuming a network-wide taxonomy according to which LOs are described and queries are formulated. However, we have seen that this assumption can be relaxed through the use of articulations to allow a provider, or a group of providers to use their own, local taxonomies.

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

First, we will investigate more complex forms of articulation between local taxonomies and the central taxonomy, to allow for richer forms of communication between providers and the SeLeNe services. Work in that direction will be based on previous work on mediation [19, 20, 15, 21, 17, 10].

Second, we will assume that the services of LO Registration, Querying and Syndication are available at each peer, based on the local taxonomy. The Syndication Service (see Deliverable 3 [4]) will then translate descriptions and term-based queries using the articulations between peers.

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 SeLeNe>
```

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

[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. Karenos and G. Samaras. A Grid-service Framework for Seld e-Learning networks. Technical report, SeLeNe Consortium, 2003. www.dcs.bbk.ac.uk/selene/.

[5] K. Keenoy, M. Levene, and D. Peterson. Personalisation and Trails in Self e-Learning Networks. Technical report, SeLeNe Consortium, 2003. www.dcs.bbk.ac.uk/selene/.

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

[7] 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.

[8] A. Magkanaraki, V. Christophides, G. Karvounarakis, and D. Plexousakis. Views and Structured Querying in Self e-Learning Networks. Technical report, SeLeNe Consortium, 2003. www.dcs.bbk.ac.uk/selene/.

[9] P. McBrien and A. Poulovassilis. Defining Peer-to-Peer Data Integration using Both as View Rules. In *Proc. Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003. Berlin, september 2003.

[10] 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.

[11] 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.

[12] 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.

[13] G. Papamarkos, A. Poulovassilis, and P.T. Wood. ECA Rule Languages for Active Self e-Learning Networks. Technical report, SeLeNe Consortium, 2003. www.dcs.bbk.ac.uk/selene/.

[14] SeLeNe: Self eLearning Networks. www.dcs.bbk.ac.uk/selene/.

[15] 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.

[16] M. Stratakis, V. Christophides, K. Keenoy, and A. Magkanaraki. E-learning Standard. Technical report, SeLeNe Consortium, 2003. www.dcs.bbk.ac.uk/selene/.

[17] 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.

[18] Y. Tzitzikas, C. Meghini, and N. Spyratos. Taxonomy-based Conceptual Modeling for Peer-to-Peer Networks. In *Proc. Intl. Conf. on Conceptual Modeling (ER'03)*, Chicago, Illinois, 2003.

[19] 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.

[20] 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.

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

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

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

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 5. 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 [22]. 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.

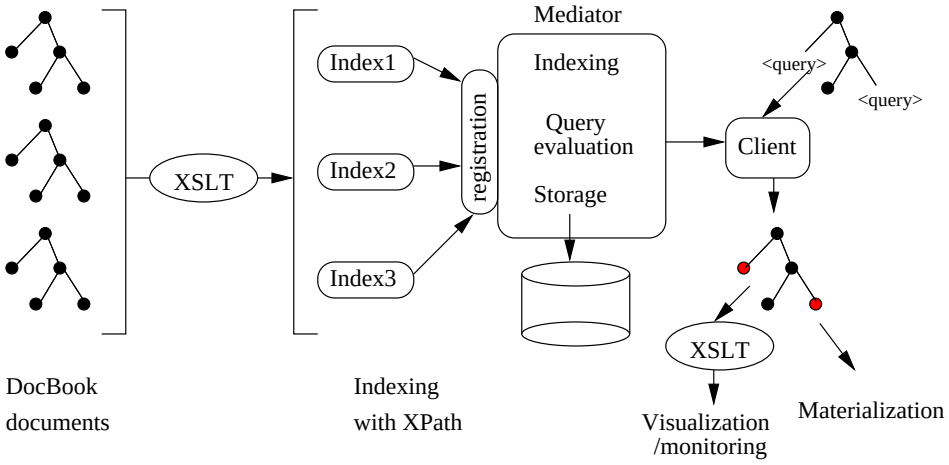

Figure 5: Overview of the system's architecture

From these documents, a program (written with the XML transformation language, XSLT) produces the description for each document. Descriptions are sent to the Update Service 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 and replaced by its result at run-time. ¿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. 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 6.

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, 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:

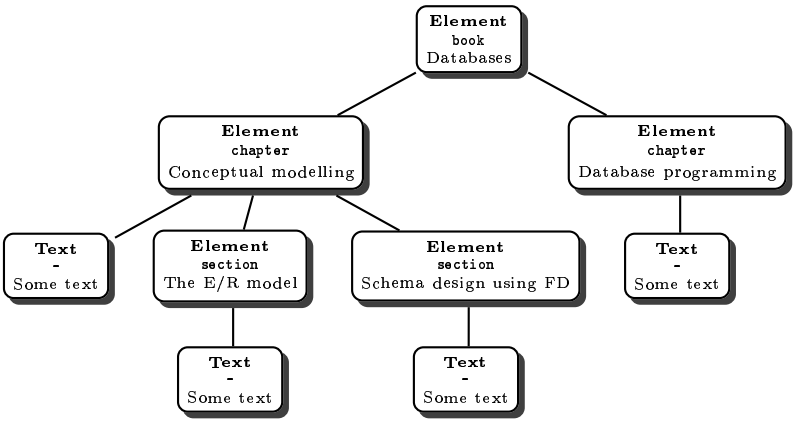

Figure 6: The hierarchical structure of the document

```
<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 7. 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.

Finally the composition graph together with the descriptions 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 [23] 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.

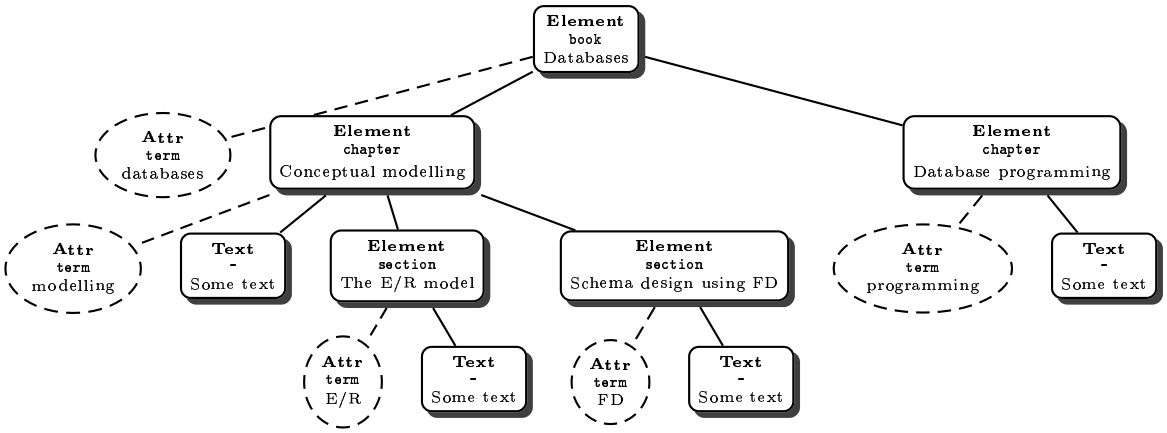

Figure 7: A document enriched with descriptions

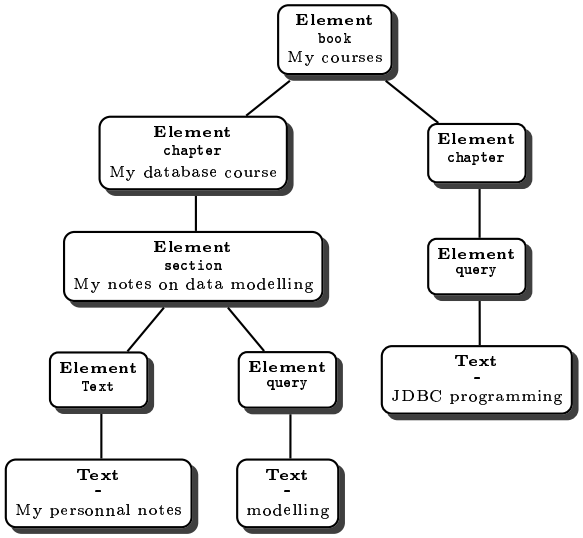

Figure 8: A derived document

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 8 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).

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 maintains the SeLeNe Catalogue and answers queries by providers and/or consumers.

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 are parts of a composite LO under construction.

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