Personalisation Services for Self e-Learning Networks

Kevin KEENOY [a], Vassilis CHRISTOPHIDES [b], George PAPAMARKOS [a], Alexandra POULOVASSILIS [a], Dimitris KOTZINOS[b], Philippe RIGAUX [c], Nicolas SPYRATOS [c],, and Peter WOOD [a]
[a] *School of Computer Science and Information Systems, Birkbeck, University of London*
[b] *Institute of Computer Science, Foundation for Research and Technology - Hellas*
[c] *Laboratoire de Recherche en Informatique, Universite Paris-Sud*

Abstract. This chapter describes personalisation services for *self e-learning networks.* A self e-learning network consists of web-based learning objects that have been made available to the network by its users, along with metadata descriptions of these learning objects and of the network's users. The proposed personalisation facilities include: querying learning object descriptions to return results tailored towards users' individual goals and preferences; the ability to define views over the learning object metadata; facilities for defining new composite learning objects and automatically deriving their descriptions; and facilities for subscribing to personalised event and change notification services. The personalisation facilities are realised using a combination of Semantic Web technologies including RDF/S, RQL, RVL, and RDF ECA rules.

Keywords. E-Learning, personalisation, metadata management, Semantic Web

1. Introduction

Life-long learning and the knowledge economy have brought about the need to support diverse communities of learners throughout their lifetimes. These learners are geographically distributed and may have heterogeneous educational backgrounds and learning needs. In the most general case, learners organise themselves into communities according to their own criteria, such as having common interests. We envisage a *self e-learning network* (SeLeNe) as existing to support a specific community of providers and learners. Learning communities can occur in an educational institution, in a workplace, or on the web.

The SeLeNe EU FP5 [1] project ran from November 2002 to January 2004 and investigated the feasibility and design of tools to support learning communities by matching learners' needs with educational resources potentially available on the Web. Our proposed personalisation services relied on the availability of semantic metadata describing educational material and learners, and included services for the discovery, sharing, and collaborative creation of learning objects (LOs), aiming to facilitate syndicated and personalised access to such resources.

LOs capture any chunk of learning resource regardless of its form, granularity and functionality. By definition, LOs encapsulate both learning content and appropriate descriptive information (i.e. metadata). LOs aim to provide self-describing learning material that once developed can subsequently be exchanged, retrieved and reused. LOs are uniquely identified via their Uniform Resource Identifier (URI) [2] while their metadata are *submitted* to the *SeLeNe* with which they are registered.

The key factor for supporting large scale interoperability, portability and reusability of LOs is the quality of the semantic description of LOs i.e., its metadata specification. The LO metadata and the schemas to which these conform form the SeLeNe *LO information space* (see Figure 1). Users need to be able to query the LO information space in order to locate LOs appropriate for their specific learning or teaching needs, and also need to be able to define personalised views over this potentially large number of heterogeneous resources.

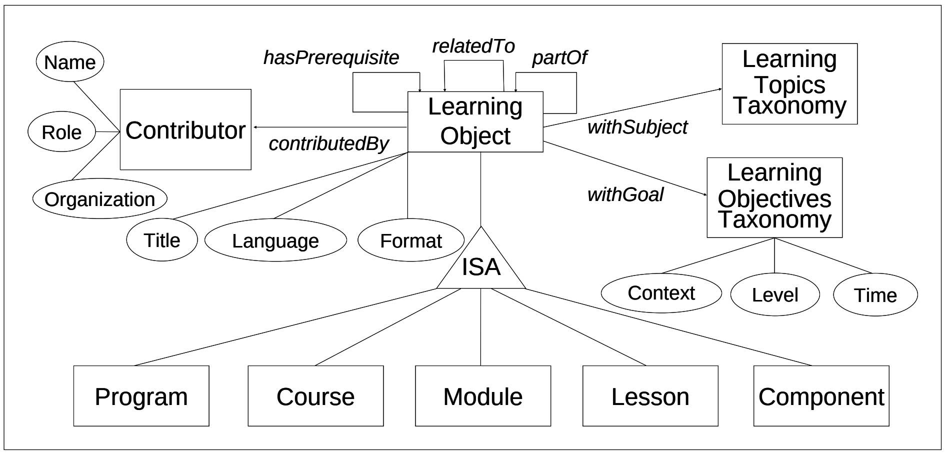

Figure 1. Overview of the SeLeNe Information Space

Following the open tradition of the Web, LOs may be physically stored in the web site of an organisation (educational or corporate) or in the web pages of individual users. A SeLeNe will not manage the LOs themselves, but will instead facilitate access to them by managing their metadata descriptions. In order to enable effective search for LOs in a SeLeNe, we assume that LO descriptions conform to e-learning standards such as IEEE/LOM (Learning Object Metadata) [3], and also employ topic-specific taxonomies of scientific domains such as ACM/CCS (Computing Classification System) [4] or taxonomies of detailed learning objectives. LO schemas and descriptions are represented in the Resource Description Framework/Schema Language (RDF/S) [5], which offers appropriate modelling primitives for the SeLeNe information space.

Figure 1 illustrates the main concepts and properties of the RDFS schemas we employed in the SeLeNe project to capture the semantics of existing e-learning standards. The information content of a LO can be described using attributes such as *title, language, format,* etc., or one or more terms from a topic-specific taxonomy like ACM/CCS, e.g. a LO about C++ could have a *withSubject* property referencing the ACM/CCS classification *D.3.2.11*, which is the taxonomic path representing *Software.ProgrammingLanguages.LanguageClassifications.ObjectOrientedLanguages*.

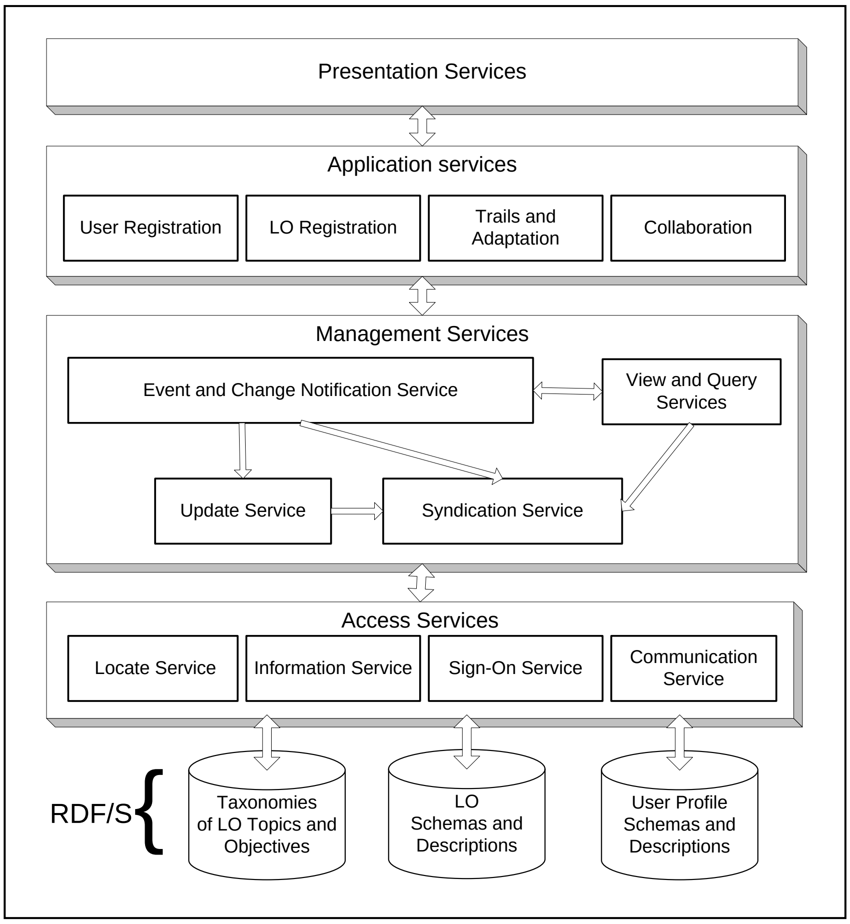

Figure 2. SeLeNe Service Architecture

The schema also allows description of the granularity of a LO (e.g., *Course*, *Lesson*), its relationships with other LOs (e.g., *hasPrerequisite*, *partOf*), and its relationships with other classes of resource (e.g., *contributedBy*). A taxonomy such as Bloom's *Taxonomy of Educational Objectives*, combined with a topics taxonomy, can be used to describe the desired learning outcomes of a LO. For example, a LO *withGoal Application.Use* (from Bloom's taxonomy) [6] and *withSubject D.3.2.11* (from ACM/CCS) is thereby described as having the educational objective "to be able to use an Object-Oriented language".

The diversity and heterogeneity of the learning communities we envisage using SeLeNes means that no single architectural design is suitable to support all of them. We thus defined a service-based architecture that can be deployed in a peer-to-peer, mediation-based or centralised fashion (which is a special case of peer-to-peer); so the deployment option best addressing the needs of any particular learning community can be chosen. We envisage each SeLeNe network as consisting of a number of peers, each of which will support some subset of the full set of SeLeNe services. Figure 2 illustrates the architecture of a SeLeNe. The facilities that are the focus of this chapter are provided by the User Registration, LO Registration, Trails and Adaptation, Event and Change Notification, and View services. We refer the reader to [7] for further discussion of the architecture and other services.

The users of a SeLeNe will include *instructors*, *learners* and *providers* of LOs – a single person could play each of these roles at different times. New users will be able to join a SeLeNe by contacting any peer of the SeLeNe that provides the User Registration service. When registering, users will supply information about themselves and their educational objectives in using this SeLeNe. This information is stored in their *personal profile*.

Providers of LOs will maintain control of the content they create and will be free to use any tools they wish to create their LO content before registering it. We call such LOs, created externally to SeLeNe, *atomic LOs*. Users will also be able to register new *composite LOs* – LOs that have been created as assemblies of LOs already registered with the SeLeNe, for example a course LO which has been created by assembling several module LOs. The SeLeNe will be able to automatically derive the taxonomical description of a composite LO from the taxonomical descriptions of its constituent LOs. The LO registration process and automatic derivation of taxonomical descriptions are discussed in Section 2.

The SeLeNe will provide facilities for defining personalised *views* over the LO information space, which we discuss in Section 3. The SeLeNe will also provide browsing and searching facilities over the LO information space, which return results tailored towards users' individual goals and preferences. This is discussed in Section 4. A user may wish to be notified of changes made to a LO's description. SeLeNe can provide automatic change detection and notification facilities by comparing the new and old descriptions of a LO. More generally, SeLeNe can support personalised notification services depending on users' profiles. Provision of such facilities is discussed in Section 5. Finally, our conclusions are presented in Section 6.

2. Registration of LOs

Registration of a LO with a SeLeNe consists of providing a metadata description including the URI of the LO. Here we focus on the *taxonomical* part of a LO's description.

A taxonomy (T, $\preccurlyeq$) consists of a set of terms *T* together with a subsumption relation $\preccurlyeq$ between terms. It can be 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 shows a taxonomy, which is used by all examples in this section.

A *taxonomical description* is a set of terms from a taxonomy. For example, if a LO contains the Quicksort algorithm written in Java then the terms `QuickSort` and `Java` can be chosen by the LO's provider to describe its content. We call the set of terms `{QuickSort,Java}` the *publisher taxonomical description* (PTD) of the LO.

An important feature of the registration process will be the ability to automatically infer a taxonomical description for a composite LO *o*, from the taxonomical descriptions of the LOs $o_1,\ldots, o_n$ it has been assembled from. This description, which 'summarises' the taxonomical descriptions of the parts of *o*, is called the *implied taxonomical description* (ITD) of *o*.

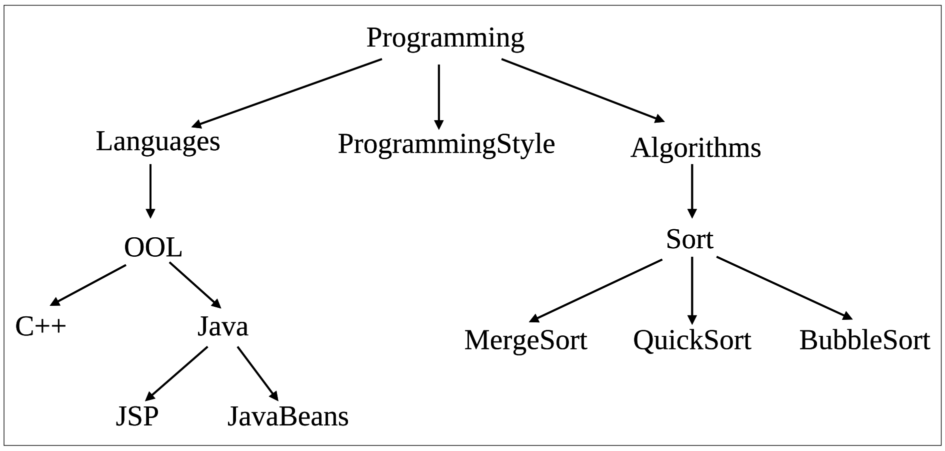

Figure 3. An example Taxonomy showing a set of terms that could be used when describing the topics of LOs in the area of computer programming. The arrows show the subsumption relationship that exists between the terms, for example the terms *MergeSort*, *QuickSort* and *BubbleSort* are subsumed by the term *Sort*, and *Sort* is subsumed by *Algorithms*.

A SeLeNe will automatically derive the taxonomical description D of a composite LO from its PTD augmented by its ITD, removing any redundant terms (i.e., terms that are subsumed by other terms). Consider LOs $o_1,\ldots, o_4$ with the following taxonomical descriptions:

$D(o_1)$ = `{QuickSort,Java}`; $D(o_2)$ = `{BubbleSort}`;
$D(o_3)$ = `{BubbleSort, C++}`; $D(o_4)$ = `{C++}`.

Intuitively, the ITD of a composite LO o expresses what its parts have in common – a composite LO o composed from o_1 and o_2 will have the ITD `{Sort}`. Intuitively, both o_1 and o_2 concern sorting algorithms, and the fact that one of them is written in Java is considered to be irrelevant as far as the composite LO is concerned. Thus, the term `Java` is not reflected in o's ITD as it is not something that both parts share. However, this does not mean a loss of information: if a user searches the SeLeNe for objects related to Java, o_1 will be in the answer set and o will not.

The ITD of a composite LO retains what its parts have in common, while subsuming the taxonomical descriptions of each part. Consider the composite LO o' with parts o_1 and o_3 – in this case the common part would be `{Sort,OOL}`. This result for the ITD should be interpreted as meaning that each part of o' concerns both sorting and object-oriented languages.

Note that a LO may generate different ITDs depending on what its 'companion' parts are. Consider the composite LO o'' with parts o_1 and o_4.The ITD of o'' is `{OOL}` – o_1 is part of each of the composite LOs o, o' and o'', but each time with a different 'companion' part. It is interesting to note that, depending on the companion part, either the 'sort-aspect' of o_1 or its 'OOL-aspect', or both, appear in the ITD.

One may wonder why the PTD of a composite LO is not sufficient and why we need to augment it by its ITD. 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. For example, suppose that the two LOs, o_1 and o_3, have been created by two different providers, with the PTDs as above. Assume now that a third provider considers these LOs as examples of good programming style, and decides to use them as parts of the new composite LO o'. The provider of o' provides the PTD `{ProgrammingStyle}`. Although this PTD 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 users searching for LOs containing material on Java and sorting algorithms. Therefore, the PTD `{ProgrammingStyle}` is augmented by the ITD `{OOL, Sort}` to obtain `{ProgrammingStyle,OOL,Sort}` as the overall taxonomical description for o'.

The ITD of a composite LO o composed of parts $o_1,\dots, o_n$ with descriptions $D_1,\dots, D_n$ is computed by a simple algorithm, which takes the Cartesian product of $D_1,\dots, D_n$, computes the least upper bound of each n-tuple and then 'reduces' the resulting set of terms by removing all but the minimal terms according to the subsumption relation $\preccurlyeq$. The overall taxonomical description of a LO o is computed by another simple algorithm: if o is atomic then its taxonomical description is just its PTD. Otherwise its taxonomical description is recursively computed from its PTD and the taxonomical descriptions of its constituent parts. Readers are referred to [8] for more details of both algorithms.

3. Declarative Queries and Views

One of the novel features of SeLeNe is its ability to exploit the available semantic relationships (see Figure 1) defined among the RDF/S classes of LOs in order to implement conceptual navigation and retrieval. We believe that the semantic relationships of LOs play a crucial role especially in peer-to-peer learning environments, where users have the ability to freely insert or modify LOs and their descriptions while at the same time they are striving for a contextual access to learning material available to a SeLeNe.

Finding LOs and exploring their semantic relationships relies on the RDF/S Query Language (RQL) [10], a declarative language for browsing and querying RDF/S schemas and descriptions. For instance, consider the following RQL query, which retrieves all *Course*s about Object Oriented Languages (*OOL*) that have been *contributedBy* someone having a *name* attribute:

```
SELECT Y, X, W
FROM{Y;ns1:Course}ns1:contributedBy{X}.ns1:name{W},ns2:OOL{Y}
USING NAMESPACE ns1=&www.ieee.org/lom.rdfs#
                ns2=&www.acm.org/class/1998ccs.rdfs#
```

An RQL `FROM` clause consists of path expressions, which provide a navigation through schemas and description bases and bind the introduced variables. The first RQL path expression

```
{Y;ns1:Course}ns1:contributedBy{X}.ns1:name{W}
```

will match instances of class *Course* and their associated *contributedBy* properties, which link them to some instance of *Contributor* and its *name* value. For each such match, we get a binding that maps Y to the *Course* resource, X to the *Contributor* and W to the *name* value. The second path expression

```
ns2:OOL{Y}
```

is evaluated for each binding of Y, and filters *Course* instances by checking whether they are classified under the topic *OOL*. Note that in this query several schemas are used, identified by the corresponding namespaces `n1` and `n2` defined in the `USING NAMESPACE` clause.

Using RQL, more complex scenarios of conceptual navigation and retrieval can be implemented using appropriate Graphical User Interfaces (GUI). In particular, RQL queries can be automatically generated from the browsing actions of learners both at the level of LO schemas and their descriptions in the SeLeNe information space [11]. For example, starting from a *Course* a learner can navigate through the *prerequisite* relationship and retrieve LOs belonging to the target course(s). Alternatively, a learner can choose to visit a *Lecture*, where (s)he can discover whether another *Lecture* or a *Module* of a *Course* is required to fully understand the content of this lecture. Additionaly, (s)he can also find out whether this *Lecture* is *required* for any *Assignment*s that (s)he has to fulfill in order to successfully assimilate *Course* material. In this context, we can consider *prerequisite* and *requires* as “strong” relationships, literally forcing a learner to follow a specific learning path while *partOf* and *SeeAlso* are rather “weaker” relationships suggesting a path but not forcing learners to follow it.

These are few examples of **query-based personalization** techniques, which are based on the RDF/S schemas employed to semantically describe LOs. However, despite our efforts for a modular, extensible and intuitive RDF/S schema design, the SeLeNe information space one has to master still remains incredibly rich. Hence, LOs’ providers may encounter difficulties when querying or updating LOs’ descriptions due to a misleading or even erroneous understanding of the LOs’ semantics. Moreover, LOs’ consumers may also need to consider LO descriptions according to their educational level and current subject of study rather than the RDF/S schemas employed to register LOs in a SeLeNe. For these reasons, we need additional support to specify and visualise appropriate **semantic views** of LO descriptions and schemas. The RDF/S View Language (RVL*)* [9] provides this ability by introducing virtual RDF/S schemas and descriptions on top of the SeLeNe information space.

To illustrate the RVL functionality we will consider a simple virtual schema (view) for instructors, which represents only OOL course material and its authors. This schema can be specified by a set of RVL statements whose output is an RDF/S virtual schema and resource descriptions. In RDF/S the uniqueness of (meta) schema labels and the ability to describe resources using labels from several schemas is ensured by the XML namespace facility. In our example this RVL statement defines a unique namespace:

```
CREATE NAMESPACE myview=&http://www.selene.org/LO.rdf#
```

The following RVL statement specifies the virtual classes `Author` and `OOLCourse` and the virtual properties `creates` and `name`:

```
VIEW rdfs:Class("OOLCourse"),rdfs:Class("Author"),
     rdf:Property("creates", Author, OOLCourse),
     rdf:Property("name", Author, xsd:string);
```

where *rdfs:Class* and *rdf:Property* are two core metaclasses provided in the default RDF/S namespaces. As in RQL, the `USING NAMESPACE` clause declares the namespaces used in view statements. The following statement 'populates' the virtual classes and properties defined in the view:

```
VIEW OOLCourse(Y),Author(X),creates(X,Y),name(X,W)
FROM {Y;ns1:Course}ns1:contributedBy{X;ns1:Author}.
      ns1:name{W}, ns2:OOL{Y}
USING NAMESPACE ns1=&www.ieee.org/lom.rdfs#
                ns2=&www.acm.org/class/1998ccs.rdfs#
```

It is worth mentioning that although the input of both RQL and RVL in the `FROM` clause is an RDF/S graph, only the latter produces virtual schemas and resource descriptions rather than the simple variable bindings returned by the former. This functionality is ensured by the `VIEW` clause, where appropriate population functions are used, taking as parameters the variable bindings produced by the `FROM` clause (and optionally a `WHERE` clause). For instance, the virtual class `OOLCourse` is populated with instances (bound to variable Y) of the base class *Course*, also classified under the topic *OOL*. The virtual class `Author` is populated with instances (bound to variable X) of the base class *Contributor*, which are the range values of the property *contributedBy* applied to *Course* resources. In other words, `Author` is populated with all the contributors who have created an OOL course. Virtual properties are populated with pairs of resources (e.g., `creates` is populated with authors having created OOL courses) or resource-value pairs (e.g., `name` is populated with the names of OOL course authors).

One of the most significant features of RVL is its ability to create virtual schemas by simply populating the two core RDF/S metaclasses *Class* (e.g., with schema classes `Author` and `OOLCourse`) and *Property* (e.g., with schema properties `creates` and `name`). A SeLeNe user could then easily formulate queries *on the view* (using a user-friendly graphical interface), such as the following RQL query retrieving the OOL courses created by the author named "Christophides":

```
SELECT Y
FROM {X}myview:creates{Y}, {X}myview:name{Z}
WHERE Z = "Christophides"
USING NAMESPACE myview=&http://www.selene.org/LO.rdf#
```

A concrete implementation of the personalization techniques presented in this section is available at the Online Curriculum Portal of the Computer Science Department of the Univeristy of Crete [12].

4. Trails and Query Adaptation

Personalisation of query results relies on the *personal profile*, which is an RDF description of the user conforming to a number of RDFS schemas, created when a user registers with a SeLeNe. Figure 4 shows a simplified version of SeLeNe's personal profile schema.

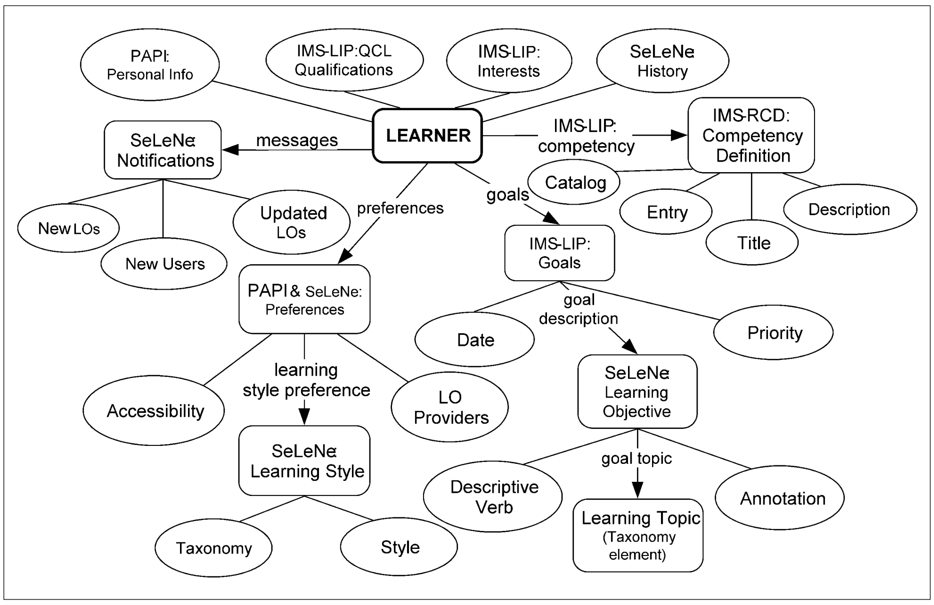

Figure 4. SeLeNe's Personal Profile Schema

To avoid proposing yet another schema for demographic and other information that is already adequately catered for by existing profile schemes, we included elements from the IEEE LTSC's Personal and Private Information (PAPI) Standard [13] and the IMS Learner Information Package (LIP) [14], extending the data model with our own elements where existing specifications fail to be expressive enough.

The shortcomings of existing learner profile specifications were generally in the recording of competencies, learning goals and preferred learning styles, and we developed RDF schemas that allow the expression of all three of these. For competencies and learning goals, we adopted the same topic and learning objectives taxonomies as those used for learning object descriptions, (i.e. the taxonomies shown in Figure 1). Other customised sections of the SeLeNe profile are related to the active functionality of SeLeNe – a history of user activity (accesses to LOs) is included, which allows the profile to adapt to take account of the user's behaviour over time, for example by automatically updating the information in it by means of Event-Condition-Action rules (see Section 5). The learner also has a *messages* property with a *Notifications* class as its target. This can store personal notifications (of new users and new or updated LOs) for the user.

Although the underlying query mechanism in SeLeNe is RQL, we envisage that users will generally search for LOs using simple keyword-based queries (possibly augmented with attribute information). Search results will then be personalised by *filtering* and *ranking* the LOs returned according to the information contained in the user's personal profile.

Filtering will take place before a query is evaluated: all keyword-based queries submitted to the SeLeNe will be routed through the Trails and Adaptation service, which will construct corresponding personalised queries. The user's original query will then be re-formulated and/or annotated to reflect elements of their personal profile. For example, if the profile records that the user only speaks English, all searches they enter can be augmented with the annotation *lang:en*, ensuring that all returned LOs will be in English. The augmented keyword query will then be translated into an appropriate RQL query and executed by the Query service. The reader is referred to [15] for the full SeLeNe profile schema and for discussion of the translation of keyword-based queries into structured queries in SeLeNe.

The next step is to rank the filtered set of LO descriptions in order of relevance to the user. Relevance of a LO to the user will be judged by matching the LO's metadata description against the combination of the personal profile and the original query. The information contained in the SeLeNe personal profile allows the following factors to be taken into consideration when judging relevance, and a combination of weighted scores for each of these factors can be used to provide an overall relevance score for each candidate LO:

- relevance of the LO to the query (judged using techniques such as those described in [16], [17]);
- whether the user has the prerequisite knowledge and experience to be able to tackle the LO;
- how well the learning objectives of the LO match the user's learning goals;
- if the user's learning styles are those catered for by the LO; if the user is likely to prefer it for other reasons (e.g., it is by a preferred provider);
- how well the LO caters for the user's accessibility requirements;
- the user's most recent activity (as contained in the history section of their profile).

The different sections of the personal profile can be matched in a focussed way against relevant sections of the LO descriptions. For one LO to be more relevant to the user than another it must meet more of the user's different preferences, or match preferences to a greater degree. For example, if the user has OOL as one of their *goal_topic*'s then a LO including OOL in its taxonomical description will score well on relevance to the user. If the LO description additionally includes the information that it caters for one of the user's learning styles then it will be considered a better match again. The best algorithm and weightings to use for this ranking needs to be determined empirically, and may well need to be adaptive.

Once a personalised ranking of the remaining LOs has been generated, the search results will then be returned to the user. SeLeNe can give the user the option of having their query results presented not as a simple list of individual LOs, but rather as a list of *trails* of LOs, where a trail is a suggested sequence of interaction with the LOs [18]. We have defined an RDF representation of trails whereby they are defined as a sub-class of the RDF *Sequence* (a sequence of LOs) with two associated properties, *name* and *annotation*, that provide additional information about the pedagogic use of the trail.

These trails will be automatically generated. Algorithms for the automatic generation of trails in hypertext systems already exist [19], but in SeLeNe's case the links between LOs are not explicit hyperlinks – links will be derived from information contained in the LO descriptions about the semantic relationships between LOs. For example, by inspecting the *LOM:Relation* fields in a collection of LO descriptions (in this case a set of results), trails of LOs with early LOs in the trail being prerequisites for later LOs can easily be derived and annotated with the term “prerequisites”.

5. Event and Change Notification

Many applications on the web need to be *reactive* i.e., to be able to detect the occurrence of specific events or changes in information content, and to respond by automatically executing the appropriate application logic. *Event-condition-action* (ECA) rules are one way to implement this kind of functionality. An ECA rule has the general syntax: `on` *event* `if` *condition* `do` *actions*. The event part specifies when the rule is *triggered*. The condition part is a query that determines if the information space is in a particular state, in which case the rule *fires*. The action part states the actions to be performed if the rule fires. These actions may in turn cause further events to occur, which may in turn cause more ECA rules to fire.

There are several advantages in using ECA rules to implement this kind of functionality: management of an application’s reactive functionality within a single rule base; analysis and optimisation techniques for ECA rules that cannot be applied if the same functionality is expressed directly in application code; and provision of a generic mechanism that can abstract a wide variety of reactive behaviours.

Motivated by these advantages of ECA rules, we proposed providing SeLeNe’s reactive functionality by means of ECA rules over RDF/S metadata. This reactive functionality includes features such as automatic propagation of changes in the description of one resource to the descriptions of other, related resources (e.g., propagation of changes in the taxonomical description of a LO to the taxonomical description of any composite LOs depending on it, or updating a user’s personal profile based on changes in their history of accesses to LOs), automatic notification to users of the registration of new LOs of interest to them and of changes in the description of resources of interest to them.

SeLeNe peers that support the Event and Change Notification service will have installed an *ECA Engine* consisting of three main components: an *Event Detector*, *Condition Evaluator* and *Action Scheduler*. The Event Detector determines which rules have been triggered by the most recent update to the local description base, by invoking the Query service to evaluate the event queries of rules that may have been triggered. The Condition Evaluator then calls the Query service to determine which of the triggered rules should fire. The Action Scheduler generates from the action parts of these rules a list of updates, which are then passed to the Update service for execution.

Our RDF ECA rule language, RDFTL [20], is implemented over FORTH’s RDFSuite query and update API [21]. In RDFTL, the event part of a rule is an expression of one of the following three forms:

(i) (INSERT | DELETE) *e* [AS INSTANCE OF *class*]

This detects insertions or deletions of resources described by the expression *e*, which is a path expression evaluating to a set of nodes, optionally followed by a clause `AS`

`INSTANCE OF` *`class`*. The rule is triggered if the set of nodes returned by *e* includes any new node (in the case of an insertion) or any deleted node (in the case of a deletion) that is an instance of the *`class`*, if specified. A system-defined variable `$delta` is available for use within the condition and actions parts of the rule, and its set of instantiations is the set of new or deleted nodes that have triggered the rule.

(ii) (INSERT | DELETE) *triple*

This detects insertions or deletions of arcs specified by *triple*, which has the form *(source, arc_name, target)*. The wildcard '_' is allowed in the place of any of a triple's components. The rule is triggered if an arc labelled *arc_name* from the *source* node to the *target* node is inserted/deleted. The variable `$delta` has as its set of instantiations the triples which have triggered the rule. The individual components of one of these triples can be referenced by `$delta`.source, `$delta`.arc_name, `$delta`.target.

(iii) UPDATE *upd_triple*

This detects updates of arcs, where *upd_triple* has the form *(source, arc_name, old* → *new)*. Here, *old* is the target node of arc *arc_name* from the *source* node before the update, and *new* is its target node after the update. Again, the wildcard '_' is allowed in the place of any of these components. The rule is triggered if an arc labelled *arc_name* from *source* changes its target from *old* to *new*. The variable `$delta` has as its set of instantiations the triples which have triggered the rule. The individual components of one of these triples can be obtained by `$delta`.source, `$delta`.arc_name, `$delta`.old_target, `$delta`.new_target.

The condition part of a rule is a query consisting of conjunctions, disjunctions and negations of path expressions. The actions part of a rule is a sequence of one or more actions. Actions can `INSERT` or `DELETE` a resource – specified by its URI – and `INSERT`, `DELETE` or `UPDATE` an arc. The actions language has the following form for each one of these cases:

```
INSERT e AS INSTANCE OF class
DELETE e [AS INSTANCE OF class]
```

for inserting or deleting a resource;

```
(INSERT | DELETE) triple (',' triple)*
```

for inserting or deleting the specified arcs(s); and

```
UPDATE upd_triple (',' upd_triple)*
```

for updating arcs by changing their target node. Wildcards are allowed in place of some of the components of triples, with the obvious semantics. We give two examples of RDFTL rules below, which refer to the LO schema illustrated in Figure 1 and the personal profile schema illustrated in Figure 4.

Firstly, if a LO is inserted whose subject (OOL, say) is the same as one of user 128's goal topics, then the following rule adds a new arc linking the newly inserted LO into the `new_LOs` collection in user 128's personal messages:

```
ON INSERT resource() AS INSTANCE OF LearningObject
IF $delta/target(rdf:type)
   = resource(http://www.dcs.bbk.ac.uk/users/128)
     /target(ims-lip:goal)
     /target(ims-lip:goaldescription)
     /target(selene:goaltopic)
DO LET $new_los :=
   resource(http://www.dcs.bbk.ac.uk/users/128)
   /target(selene:messages)/target(selene:new_LOs)
   IN INSERT ($new_los,seq++,$delta);;
```

Here, the event part checks if a new resource belonging to the `Learning Object` class has been inserted. The condition part checks if the inserted LO has a subject which is the same as one of user 128's goal topics. The `LET` clause defines the variable `$new_los` to be user 128's collection of new LOs. Finally, the `INSERT` clause inserts a new arc from `$new_los` to the newly inserted LO (the syntax `seq++` indicates an increment in the collection's element count).

As a second example, if any property of a LO whose subject is the same as one of user 128's goal topics is updated, then the following rule adds a new arc linking user 128's `updated_LOs` collection to the modified LO:

```
ON UPDATE (resource()AS INSTANCE OF LearningObject,_,_->_)
IF $delta/target(rdf:type)
   = resource(http://www.dcs.bbk.ac.uk/users/128)
     /target(ims-lip:goal)
     /target(ims-lip:goaldescription)
     /target(selene:goaltopic)
DO LET $updated_los :=
   resource(http://www.dcs.bbk.ac.uk/users/128)
   /target(selene:messages)
   /target(selene:updated_LOs)
   IN INSERT ($updated_los,seq++,$delta);;
```

We refer the reader to [20,22] for an overview of related work in ECA rule languages, the syntax and semantics of RDFTL, conservative tests for determining the termination and confluence of sets of RDFTL rules, and architecture, implementation and performance details.

6. Concluding Remarks

This chapter has described several techniques for providing personalisation services in *self e-learning networks*. Our use of RDF/S for representing the descriptions of educational material and users within a SeLeNe has allowed us to reuse and extend as necessary a number of existing metadata standards for learning objects and learners. It has also allowed us to adopt a uniform approach to manipulating information about learning objects and learners in the RQL, RVL and RDFTL languages, and a uniform approach underpinning the SeLeNe personalisation services.

The novel aspects of the SeLeNe project compared with other peer-to-peer, RDF-based systems for supporting e-learning (e.g. UNIVERSAL, Edutella, Elena[1]) included:

- the use of Semantic Web technologies to support the collaborative creation and semi-automatic description of new composite LOs, which does not seem to be addressed specifically by any other system and which gives users flexibility in the generation of learning resources tailored to their needs;
- declarative views over combined RDFS/RDF descriptions (i.e. over both the LO descriptions and their schemas) that ensure compositionality of queries with views and mappings, giving users the power to narrow their view of a rich information space to see only the parts of relevance to them;
- personalised event and change notification services, operating at the level of LO or user RDF/S descriptions, allowing users to be kept up-to-date with relevant changes and additions to a SeLeNe as it evolves;
- automatic generation of trails of LOs from their descriptions assists users in planning individual learning paths.

Many open issues still remain. As the SeLeNe project was a feasibility study, the complete architecture designed in the project and illustrated in Figure 2 was not implemented, and therefore it was not possible to undertake a usability evaluation. The facilities described in Sections 2, 3 and 5 were fully implemented and those in Section 4 partially implemented. The algorithms for personalised ranking of query results remain to be empirically evaluated. This evaluation would need to include a study of the comparative usefulness, for personalisation purposes, of the different cognitive models included in our taxonomy of learning and cognitive styles.

There is still no standard query or update language for RDF, although we believe that the RQL, RVL and RDFTL languages we have developed provide sound and expressive foundations for the development of such standards, and also for development of optimisation techniques for query, update and view languages over RDF. Whatever standards eventually emerge for such RDF languages, if ECA rules are to be supported on RDF repositories then the event sub-language for RDF ECA rules needs to be designed so that it matches up with the actions sub-language. In general, the ability to analyse and optimise ECA rules needs to be balanced against their complexity and expressiveness. Another important open area is combining ECA rules with transactions and consistency maintenance in RDF repositories.

Finally, in implementing a system such as this, the design of user interfaces enabling easy and intuitive access to SeLeNe's advanced personalisation services is crucial – end-users will need to be shielded from the complexities of RDF and the RQL, RVL and ECA languages, and also from the complex taxonomies of topics, competencies and goals in use by the system.

[1] `http://www.ist-universal.org`, `http://edutella.jxta.org`, `http://www.elena-project.org`

References

[1] The European Commission: CORDIS FP5web. See http://cordis.europa.eu/fp5/ (2002)

[2] Berners-Lee, T., Fielding, R., Masinter, L.: Request for Comments 2396: Uniform Resource Identifiers (URI) Generic Syntax. See http://www.ietf.org/rfc/rfc2396.txt (1998)

[3] IEEE: Draft Standard For Learning Object Metadata. See http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (2002)

[4] Association for Computing Machinery: The ACM Computing Classification System (1998 Version). See http://www.acm.org/class/1998/ (1998)

[5] Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. See www.w3.org/TR/rdf-schema (2004)

[6] Bloom, B. S., Krathwohl D. R. (Eds.): Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain. Longman, New York. (1956)

[7] Samaras, G., Karenos, K., Christodoulou, E.: A Grid service framework for Self e-Learning Networks. See http://www.dcs.bbk.ac.uk/selene/reports/Del3.pdf (2003)

[8] Rigaux, P., Spyratos, N.: Generation and syndication of learning object metadata. See http://www.dcs.bbk.ac.uk/selene/reports/Del4.1-2.2.pdf (2004)

[9] Magkanaraki, A., Tannen, V., Christophides, V., Plexousakis, D.: Viewing the Semantic Web Through RVL Lenses. In: Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 1(4), 2004, pp. 359-375

[10] Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., Tolle, K.: RQL: A Functional Query Language for RDF. In: The Functional Approach to Data Management: Modelling, Analyzing and Integrating Heterogeneous Data, P.M.D.Gray, L.Kerschberg, P.J.H.King, A.Poulovassilis (Eds.), LNCS Series, Springer-Verlag, (2004)

[11] Athanasis, N., Christophides, V., Kotzinos D.: Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). In: Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. (2004)

[12] Kotzinos, D., Pediaditaki, S., Apostolidis, A., Athanasis, N., Christophides, V.: Online Curriculum on the Semantic Web: The CSD-UoC Portal for Peer-to-peer e-learning. In: Proceedings of the 14th International World Wide Web Conference (WWW'05), Chiba, Japan, May 10-14, 2005. (2005)

[13] LTSC Learner Model Working Group of the IEEE: IEEE P1484.2/D7, 2000-11-28 Draft Standard for Learning Technology - Public and Private Information (PAPI) for Learners (PAPI Learner). IEEE, (2000)

[14] Smythe, C., Tansey, F., Robson, R.: IMS Learner Information Package Information Model Specification. See http://www.imsproject.org/profiles/lipinfo01.html (2001)

[15] Keenoy, K., Levene, M., Peterson, D.: Personalisation and trails in Self e-Learning Networks. See http://www.dcs.bbk.ac.uk/selene/reports/Del4.2-1.4.pdf (2003)

[16] Anyanwu, K., Sheth, A. P.: Rho queries: enabling querying for semantic associations on the semantic web. In: Proceedings of the Twelfth International World Wide Web Conference, WWW2003, pages 690-699, Budapest, Hungary, May 2003. ACM, (2003)

[17] Stojanovic, N., Studer, R., Stojanovic, L.: An approach for the ranking of query results in the Semantic Web. In: Proceedings of ISWC 2003, Sanibel Island, Florida, USA, October 2003. LNCS 2870, Springer (2003)

[18] Peterson, D., Levene, M.: Trail Records and Navigational Learning. London Review of Education, 1, pp 207-216, (2003)

[19] Wheeldon, R., Levene, M.: The Best Trail algorithm for assisted navigation of web sites. In: Proceedings of the 1st Latin American Web Congress (LA-WEB'03), Santiago, Chile (2003)

[20] Papamarkos, G., Poulovassilis, A., Wood, P.: Event-Condition-Action rules on RDF metadata in P2P environments. Computer Networks 50(10), pp 1513-1532 (2006)

[21] Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., Tolle, K.: The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In: Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb 2001) (2001)

[22] Papamarkos, G., Event-Condition-Action Rule Languages over Semi-Structured Data, PhD Thesis, University of London (2007).