

Personalisation Services for Self e-Learning Networks^{*}

Kevin Keenoy¹, Alexandra Poulouvassilis¹, Vassilis Christophides², Philippe Rigaux³, George Papamarkos¹, Aimilia Magkanaraki², Miltos Stratakis², Nicolas Spyratos³, and Peter Wood¹

¹ School of Computer Science and Information Systems, Birkbeck, University of London {kevin,ap,gpapa05,ptw}@dcs.bbk.ac.uk

² Institute of Computer Science, Foundation for Research and Technology – Hellas (FORTH-ICS) {christop,aimilia,mstratak}@ics.forth.gr

³ Laboratoire de Recherche en Informatique, Université Paris-Sud {rigaux,spyratos}@lri.fr

Abstract. This paper describes the personalisation services designed for *self e-learning networks* in the SeLeNe project. 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 facilities for subscribing to personalised event and change notification services. We show the feasibility of automatically deriving descriptions for composite learning objects and of realising the personalisation facilities using a service-based architecture, employing a combination of existing and new Semantic Web technologies including RDF/S, RQL, RVL, and ECA rules.

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 have heterogeneous educational backgrounds and learning needs.

The SeLeNe (*self e-learning networks*) project is investigating the feasibility and design of a tool to support learning communities, matching learners' needs with the educational resources potentially available on the Web. SeLeNe relies on semantic metadata describing educational material, and is developing services for the discovery, sharing, and collaborative creation of *learning objects* (LOs), facilitating a syndicated and personalised access to such resources. The

^{*} Work supported by the SeLeNe project (Self e-Learning Networks), funded by EU FP5 under action line V.1.9 CPA9 of the IST 2002 Work Programme (IST-2001-39045). See <http://www.dcs.bbk.ac.uk/selene>.

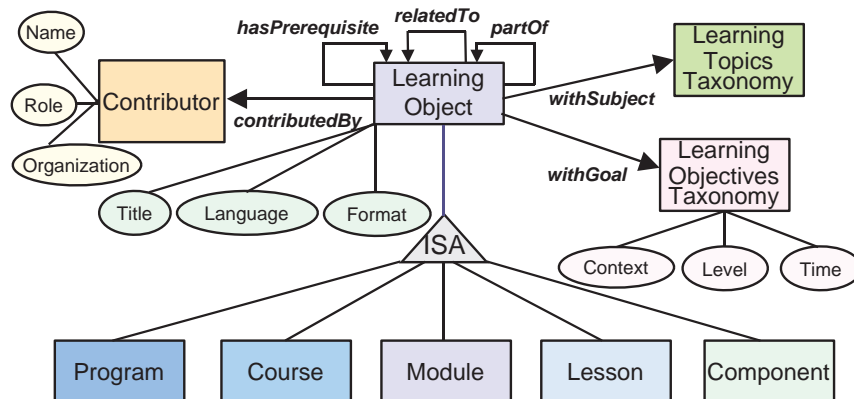


Fig. 1. Overview of the LO Descriptive Schemas in SeLeNe

LO metadata descriptions and the schemas that these conform to form SeLeNe's *LO information space*. Users need to be able to query the LO information space in order to locate resources 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.

A SeLeNe will exist to support a specific community of authors and learners. Learning communities can occur in an educational institution, in the work-place, in a geographical region, or on the Web. In the most general case, learners organise themselves into communities according to their own criteria, such as needs, interests, etc. We assume that the LOs described by a SeLeNe are those electronic, web-based, sharable chunks of reusable learning content that have been explicitly made available to the network by its users. LOs are made available to other users of the SeLeNe via metadata describing these LOs that includes the LO's URI. We call the process of metadata submission *registering a LO with a SeLeNe*.

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, LO descriptions conform to e-learning standards such as IEEE/LOM (Learning Object Metadata), and also employ topic-specific taxonomies of scientific domains such as ACM/CCS (Computing Classification System) or taxonomies of detailed learning objectives. LO schemas and descriptions are represented in the Resource Description Framework/Schema Language (RDF/S), which offers advanced modelling primitives for the SeLeNe information space.

Figure 1 illustrates the main concepts and properties of the RDFS schemas employed by SeLeNe 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 (for example, 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.Programming.Languages.Language-Classifications.Object-Oriented.Languages*). 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) 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”.

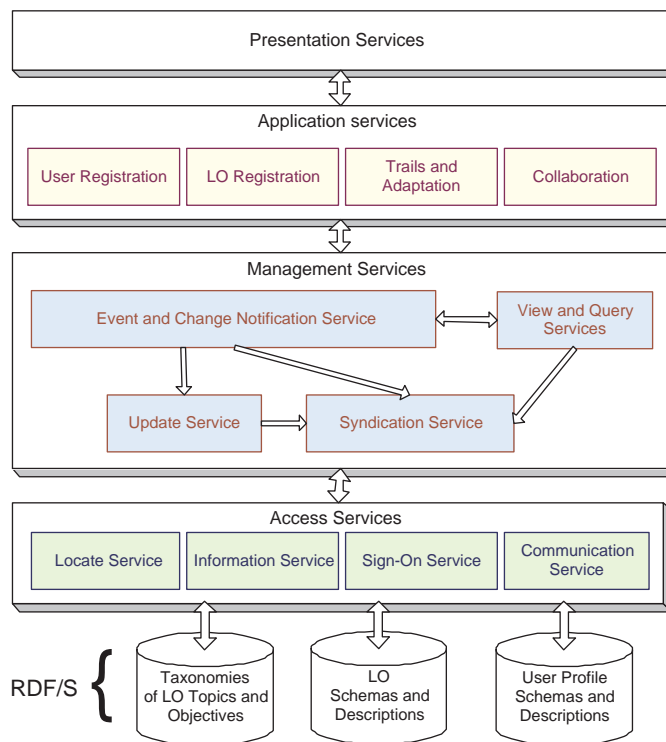


Fig. 2. SeLeNe Service Architecture

The diversity and heterogeneity of the communities we envisage using SeLeNes means that no single architectural design will be suitable to support all of them. We have thus defined a service-based architecture that can be deployed in a centralised, mediation-based or peer-to-peer fashion, so the deployment option best addressing the needs of any particular learning community can be cho-

sen. Each SeLeNe network will consist 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 paper are provided by the User Registration, LO Registration, Trails and Adaptation, Event and Change Notification, and View services. We refer the reader to [1] 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*.

Authors 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 Sect. 2.

The SeLeNe will provide facilities for defining personalised *views* over the LO information space, which we discuss in Sect. 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 Sect. 4. A user may wish to be notified of changes made to a LO's description. SeLeNe will provide automatic change detection and notification facilities by comparing the new and old description of the LO. More generally, SeLeNe will support personalised notification services depending on users' profiles. Provision of these facilities is discussed in Sect. 5.

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, \preceq) consists of a set of terms T together with a subsumption relation \preceq 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.

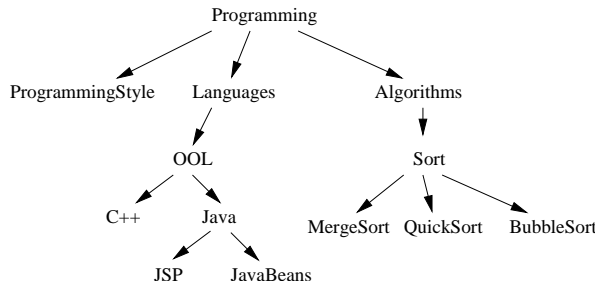


Fig. 3. A Taxonomy

An important feature of SeLeNe’s 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, \dots, 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 .

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, \dots, o_4 with the following taxonomical descriptions:

$$\begin{aligned}
 D(o_1) &= \{\text{QuickSort}, \text{Java}\}; & D(o_2) &= \{\text{BubbleSort}\}; \\
 D(o_3) &= \{\text{BubbleSort}, \text{C++}\}; & D(o_4) &= \{\text{C++}\}.
 \end{aligned}$$

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 $\{\text{Sort}\}$. 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 $\{\text{Sort}, \text{OOL}\}$. This result can be interpreted as meaning 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 $\{\text{OOL}\}$ – 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 SeLeNe 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 \preceq . 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 [2] for more details of both algorithms.

3 Declarative Queries and Views

Finding LOs in a SeLeNe will rely on RQL, a declarative query language offering browsing and querying facilities over RDF/S descriptions. For instance, consider the following RQL query, which retrieves all *Course* resources about *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.

As well as advanced querying facilities provided by an expressive RDF/S query language such as RQL, personalisation of LO descriptions and schemas is also needed. For instance, a learner might want LOs presented according to his/her educational level and current course of study. The *RVL* language [3] provides this ability by offering techniques for the reconciliation and integration of heterogeneous metadata describing LOs, and for the definition of personalised views over a SeLeNe information space.

To illustrate the functionality of RVL 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 ‘creates’ 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 with appropriate instances copied from the source description base:

```
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#
```

This statement works much like a query. However, although the input of both is an RDF/S graph, RVL produces virtual schemas and resource descriptions instead of simple variable bindings represented in some (nested) tabular form.

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 can then easily formulate queries *on the view*, 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#
```

4 Trails and Query Adaptation

Personalisation of query results will rely 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.

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

The shortcomings of existing learner profile specifications are generally in the recording of competencies, learning goals and preferred learning styles, and we have developed RDF schemas that allow the expression of all three of these. For competencies and learning goals, we adopt the same topic and learning objectives taxonomies as those used for learning object descriptions, (i.e. the taxonomies shown in Fig. 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 will allow the profile to adapt to take account of the user’s behaviour over time, automatically updating the information therein by means of a generic set of Event-Condition-Action rules (see Sect. 5) associated with all SeLeNe profiles. The learner also has a *messages* property with a *Notifications* class as its target. This will store personal notifications (of new users and new or updated LOs) for the user.

Although the underlying query mechanism in SeLeNe is RQL, users will generally search for LOs using simple keyword-based queries (possibly augmented with attribute information). Search results will 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

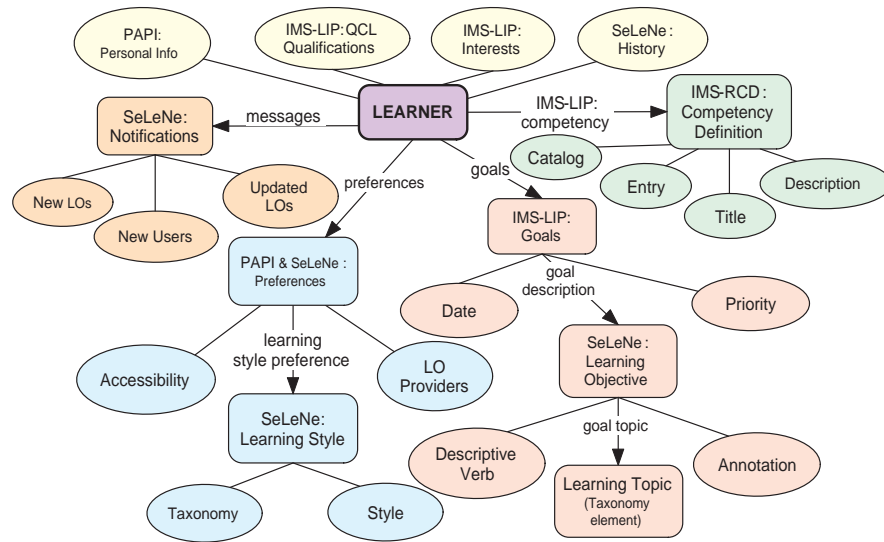


Fig. 4. SeLeNe's Personal Profile Schema

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 [4] 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 set of LO descriptions in order of relevance to the user. Relevance will be judged 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: relevance of the LO to the query; how well the LO caters for the user's accessibility requirements; 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 author); and 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 will 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. 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 [5], but the links between LOs are not explicit hyperlinks – they 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. Also known as *active rules* or *triggers*, ECA rules are supported in some form by all the major DBMS vendors. 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 database 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 database updates 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 provide 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.

In our RDF ECA rule language, the event part of a rule is an expression of one of the following three forms:

(i) [*let-expressions* IN] (INSERT | DELETE) *e*

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*. *let-expressions* is an optional set of local variable definitions of the form `let variable := p`, where *p* is a path expression. 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) [*let-expressions* IN] (INSERT | DELETE) *triple*

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

(iii) [*let-expressions* IN] 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 values of *source* in the triples which have triggered the rule.

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:

```
[let-expressions IN] (INSERT | DELETE) e
for inserting or deleting a resource;
[let-expressions IN] (INSERT | DELETE) triple (',' triple)*
for inserting or deleting the specified arcs(s); and
[let-expressions IN] 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 RDF ECA rules below, which refer to the LO schema illustrated in
Fig. 1 and the personal profile schema illustrated in Fig. 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 `LearningObject` 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 in the rule's action defines the variable `$new_los` to be user 128's new LOs collection. Finally, the `INSERT` clause inserts a new arc from `$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);;
```

The RDF ECA language we have described here has evolved from that presented in [6] and now matches more closely the update API provided by FORTH's RDFSuite [7]. We are currently implementing our language over RDFSuite's query/update API.

6 Comparison with Related Systems

In this section we consider four related systems: UNIVERSAL, Edutella, Elena, and SWAP⁴, and highlight the areas where SeLeNe extends or complements their functionality.

UNIVERSAL is a business-to-business brokerage platform aiming to support higher education institutions in the exchange of learning resources. It allows institutions to advertise their learning resources, and provides an RDF/S-based catalog which can be browsed to find and access learning resources. Edutella provides a peer-to-peer infrastructure for connecting peers supporting different types of repositories, query languages, and metadata schemas. Each peer implements a number of basic services such as querying, replication and mapping between different schemas. Elena provides a mediation infrastructure for learning services. It includes dynamic learner profiling using 'personal learning assistants'. SWAP does not address e-learning specifically, but is investigating the integration of semantic web and peer-to-peer technologies in order to support knowledge sharing. It is developing technology both for allowing users individual views of knowledge and for effective sharing of knowledge.

The novel aspects of SeLeNe compared with these systems include:

- collaborative creation and semi-automatic description of composite LOs, which do not seem to be addressed specifically by any other system;
- declarative views over combined RDFS/RDF descriptions (i.e. over both the LO descriptions and their schemas). SWAP provides users with the ability to define views over RDF descriptions only, while Edutella and Elena do not ensure the compositionality of queries with views and mappings;
- personalised event and change notification services. Event notification services are used in Edutella to assist in rule-based clustering of peers, where the events are caused by peers connecting to, or disconnecting from, a super-peer; SeLeNe's events, on the other hand, operate at the level of LO or user descriptions;
- automatic generation of trails of LOs from their descriptions.

7 Concluding Remarks

This paper has described several novel techniques for providing the personalisation services of *self e-learning networks*.

⁴ www.ist-universal.org , edutella.jxta.org, www.elena-project.org and swap.semanticweb.org respectively.

A number of open issues remain. Firstly, there is as yet no standard query or update language for RDF, although we believe that the RQL, RVL and RDF ECA languages we have described here 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.

Secondly, 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 this paper we have shown how this was done for our particular RDF ECA language, but in general the ability to analyse and optimise ECA rules needs to be balanced against their complexity and expressiveness, and this issue also needs to be borne in mind in future developments in ECA rule languages for RDF. Another important open area is combining ECA rules with transactions and consistency maintenance in RDF repositories.

Thirdly, algorithms for personalised ranking of query results need to be empirically evaluated. This evaluation will 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.

Finally, in implementing the system the design of user interfaces enabling easy and intuitive access to SeLeNe's advanced personalisation services will be crucial – 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.

References

1. Samaras, G., Karenos, K., Christodoulou, E.: A Grid service framework for Self e-Learning Networks. See <http://www.dcs.bbk.ac.uk/selene/reports/Del13.pdf> (2003)
2. Rigaux, P., Spyrtos, N.: Generation and syndication of learning object metadata. See <http://www.dcs.bbk.ac.uk/selene/reports/Del14.1-2.2.pdf> (2004)
3. Magkanaraki, A., Tannen, V., Christophides, V., Plexousakis, D.: Viewing the semantic web through RVL lenses. In: Proceedings of the Second International Semantic Web Conference (ISWC'03), Sanibel Island, Florida, USA (2003)
4. Keenoy, K., Levene, M., Peterson, D.: Personalisation and trails in Self e-Learning Networks. See <http://www.dcs.bbk.ac.uk/selene/reports/Del14.2-1.4.pdf> (2003)
5. 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)
6. Papamarkos, G., Poulouvasilis, A., Wood, P.: Event-condition-action rule languages for the semantic web. In: Proceedings of the Workshop on Semantic Web and Databases, at VLDB'03, Berlin. (2003)
7. 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)