

LUI SA

Learning Content Management System Using Innovative Semantic Web Services Architecture

IST- FP6 - 027149



Deliverable D3.2 Annotation profile specification

Matthias Palmér
Fredrik Enoksson
Ambjörn Naeve

Due date of deliverable: 31/08/2006
Actual submission date: 19/09/2006

Start date of the project: 01/03/2006

Duration: 30 Months

Matthias Palmér
Uppsala Learning Lab

Version 1.0, dated 23/08/2006

Change History

Version	Date	Status	Author (Partner)	Description
0.1	04/08/2006		KMR (ULL)	Review version
0.2	22/08/2006		KMR (ULL)	Adjusted according to review comments
1.0	23/08/2006		KMR (ULL)	Finalized according to template requirements

EXECUTIVE SUMMARY

This report outlines the concept of an Annotation Profile as a way to edit metadata around resources, typically learning objects, in a manner that carries semantics. The focus on RDF is natural since it is currently the least common denominator for semantic interoperability. For the purpose of LUISA this is sufficient since the WSMML RDF expression is good enough.

Existing schema information are acknowledged as useful but not sufficient enough for providing the necessary instructions for automatically configuring editors to specific tasks. Hence, in this respect, Annotation Profiles are designed to be complementary.

Instead of inventing a formalism from scratch the decision has been made to extend the existing Fresnel Display Vocabulary which is designed to be extensible.

This is not the final specification but only an intermediate version. The final version will be complete and better aligned with Fresnel.

Document Information

IST Project Number	FP6 – 027149	Acronym	LUISA
Full title	Learning Content Management System Using Innovative Semantic Web Services Architecture		
Project URL	http://www.luisa-project.eu		
Document URL			
EU Project officer	Kypros Kyprianou		

Deliverable	Number	D3.2	Title	Annotation Profile Specification
Work package	Number	3	Title	LO Annotation

Date of delivery	Contractual	31/08/2006	Actual	
Status	Version 1.0, dated 23/08/2006		final	<input type="checkbox"/>
Nature	Report <input checked="" type="checkbox"/> Demonstrator <input type="checkbox"/> Other <input type="checkbox"/>			
Dissemination Level	Public <input checked="" type="checkbox"/> Consortium <input type="checkbox"/>			
Abstract (for dissemination)	<p>This deliverable outlines the idea, expression and intended usage of 'Annotation Profiles' for the purpose of editing and presenting RDF metadata. Within LUISA it will be used to edit and present WSML instances expressed in RDF.</p> <p>The Annotation Profiles are configurations that complement schema information for the purpose of automatic generation of domain specific user interfaces.</p>			
Keywords	Metadata editing, RDF, RDF Schema, Application Profile, SHAME, Annotation Profile, Fresnel Display Vocabulary			

Authors (Partner)	Matthias Palmér (ULL), Fredrik Enoksson (ULL), Ambjörn Naeve (ULL)			
Responsible Author	Matthias Palmér		Email	matthias@nada.kth.se
	Partner	ULL	Phone	+46 18 471 6290

Project Consortium Information

Partner	Acronym	Contact
Atos Origin S.A.E. (Coordinator)	ATOS 	Nuria de Lama Atos Origin S.A.E. c/ Albasanz 12 E-28037 Madrid, Spain Email: nuria.delama@atosorigin.com Tel.: +34 91 214 9321 Fax: +34 91 754 3252
University of Alcalá de Henares	UAH 	Dr. Miguel-Angel Sicilia Information Research Unit University of Alcalá Ctra. De Barcelona, Km 33.6 E-28871Alcalá de Henares (Madrid), Spain Email: msicilia@uah.es Tel.: +34 91 886 6603 Fax: +34 91 885 6646
University of Uppsala	ULL 	Dr. Ambjorn Naeve University of Uppsala Kyrkogårdsgatan 2 C Uppsala Email: amb@nada.kth.se Fax: +46 184-716-294
Open University	OU 	Dr. John Domingue Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom Email: j.b.domingue@open.ac.uk Tel.: +44 1908 655014 Fax: +44 1908-653-169
University Henri Poincaré	UHP 	Dr. Monique Grandbastien University Henri Pointcaré Vandoeuvre les Nancy 54506, PO Box 239, France. Email: monique.grandbastien@loria.fr Fax: : +33 383-278-319
Giunti Interactive Labs S.r.l.	GIUNTI 	Dr. Fabrizio Giorgini Giunti Interactive Labs S.r.l. Abbazia dell'Annunziata Via Portobello Baia del Silenzio 16039 Sestri Levante (GE), Italy Tel.: +39.0185.42123 Fax: +39.0185.43347
EADS – Corporate Research Centre	EADS 	Anne Monceaux EADS – Corporate Research Centre Avenue Didier Daurat - Centreda 1, Toulouse, 31700, France. Email: anne.monceaux@airbus.com Tel.: +33 561-184-725 Fax: +33 561-187-611

Table of Contents

EXECUTIVE SUMMARY.....	3
1 SCOPE OF THIS REPORT.....	7
2 INTRODUCTION.....	8
3 EDITING RDF STATEMENTS	9
4 AN EDITING EXTENSION TO FRESNEL	11
4.1 The role of Lenses.....	12
4.2 The role of Formats.....	13
4.3 The role of RDF Schema and OWL.....	13
5 OUTLINE OF FRESNEL EDITING EXTENSION VOCABULARY.....	15
6 GLOSSARY AND ACRONYMS.....	16
7 REFERENCES	17

1 SCOPE OF THIS REPORT

This report is a first intermediate version of the Annotation Profile Specification. The requirements of an Annotation Profile Specification, as outlined in the LUISA project description, can be fulfilled in many ways. We have decided to extend an existing initiative, Fresnel Display Vocabulary [1], rather than inventing a separate and overlapping solution. Fresnel is currently focused on display issues but is designed in a manner that is extension friendly.

This intermediate report fixates the *requirements* of the extension and *outlines an initial guide* to how the extension will be done. The details are still under discussion with the Fresnel community as well as within the LUISA consortium and are intentionally left out at this point. Especially since they will change a number of times before the final version.

Furthermore, the Fresnel Editing Extension will provide the groundwork upon which further designs will be layered. E.g. the concept of a Profile is currently outside of the scope of Fresnel since it describes an intention of how a resource should be edited, closely related to a metadata standard or OWL ontology. Today Fresnel only provides a set of possible presentations of which only a few or none will fit.

Another dimension currently lacking in Fresnel is that of reusability of the Fresnel configurations. To simplify for the Annotation Profile developer it would be suitable to have a library of building blocks that could easily be composed into Annotation Profiles. It would also increase interoperability if these building blocks came from a range of metadata standards or schemas.

This initial report is organized as follows:

Section 2 introduces the idea of Annotation Profiles. Section 3 discusses completeness of RDF editing. Section 4 introduces Fresnel and discusses requirements and extension mechanisms followed by a brief outline of how Fresnel constructs can be used for the purpose of editing. Finally section 5 gives a brief outline of the Editor Vocabulary that will come.

2 INTRODUCTION

The purpose of the Annotation profile is to provide an editor-independent description of how to *edit* statements in an RDF graph [4]. This is sufficient for the purposes of LUISA since WSML has an RDF expression, for a longer discussion on this see section 5 in Deliverable 3.1 – State of the art.

There exists many different approaches for editing RDF statements, some of which are specific to singular vocabularies, others make use of specific configurations, RDF schemas [2] or OWL ontologies [5] to adapt to the vocabulary at hand. It is only the latter class of editors – which we will refer to as *profile based editors* – that is the target of this specification.

An annotation profile is meant to be specific enough to allow the user interface of a profile based editor – i.e. a form – to be generated automatically without human intervention. This allows very specific user interfaces that aid the end users to edit RDF according to specific vocabularies. A learning objects Annotation Profile according to the IEEE/LOM standard is a typical scenario where profile based editors are useful. However, there are situations where the task is unspecific – for example to edit generic, completely unknown, RDF – in this situation a profile based editor is not the right choice.

We will now go through features of Annotation Profiles. The features can be divided into the categories of *completeness*, *structure*, *presentation*, and *interaction*.

- **Completeness** – includes support for editing an arbitrary well formed RDF statement.
- **Structure** – includes selection, order and grouping of properties as well as cardinality constraints.
- **Presentation** – includes labels on fields, descriptions to aid the user in deciding how to edit. It also includes font, colour, size, indentations, and everything else that has to do with appearance.
- **Interaction** – includes hints on how to choose values from vocabularies, e.g. checkboxes, radiobuttons, dropdown menus, or search dialogues. But it can also include mechanisms for string validation or control of auto-complete mechanisms.

Note that a few of these features overlap with what can be derived from RDF Schemas or OWL ontologies, e.g. cardinality constraints in OWL. When such information exists, profile based editors is required to make use of it, allowing annotation profile authors to avoid duplication of information.

For the benefit of the following discussions it is useful to introduce the concept of a *property editor* as an editor of a single statement with a fixed subject. Profile based editors will need to support a range of property editors to be complete, i.e. to edit all kinds of RDF statements. Note, that this does not prevent more complicated editor constructions, e.g. editing of multiple statements at once. However, in the simplest case an annotation profile will be realized by a set of property editors.

3 EDITING RDF STATEMENTS

The features of the completeness category are easily derived from the expressibility of RDF and can be exhaustively covered by going through different kinds of property editors.

A single property editor, capable of editing all kinds of statements as well as converting between e.g. literal and resource in object position, is doable but too generic to be useful for the end user who wants more control. Instead we will consider a range of different property editors that handle different predicate and object types. Furthermore, we will also consider different flavours of interaction for the property editors.

In the following discussion property editors are analysed by their *value type* and *value interaction style* respectively. For both the predicate and object the value type alternatives are given directly by RDF. The value interaction style alternatives are considered from the perspective of an author defining a property editor, e.g. a predicate can be fixed in the property editor or left free for the end user to input:

- **Fixed** – the value is fixed in the property editor. The fixed alternative is only allowed for the predicate when the value type is URI. A fixed object could be allowed in principle but it makes little sense.
- **Free** – means that the user can input free text. For non blank node predicates or resources the text should be a valid URI. For literals with or without language any text is allowed and for datatyped literals the string should be within the lexical space prescribed.
- **Set** – means that a set of values are prescribed. For predicates, the set can be defined as subproperties of a property or instances of a property class. For objects with value type resource, the set can be defined as instances or subclasses of a specific class. For objects with value type plain literal or datatyped literal, a set of values should be possible to define in the property editor.
- **Other** – means an open set of values that cannot be defined at the time of defining the property editor. Typically resources are found via searching or browsing separate metadata repositories. Hence, the interpretation of *other* is dependent on the context.

The following table summarizes the value types and value interaction styles that are possible and reasonable for property editors. Consequently there is support for defining them in Annotation Profiles:

Table 1: Value interaction styles for the value types.

<i>Value type</i>		<i>Value interaction style</i>			
		<i>Fixed</i>	<i>Free</i>	<i>Set</i>	<i>Other</i>
<i>Predicate</i>					
Blank node		–	–	yes	yes
URI		yes	yes	yes	yes
<i>Object</i>					
Resource	blank node	–	–	yes	yes
	URI	–	yes	yes	yes
Literal	plain	–	yes	yes	–
	with language	–	yes	–	–
	with datatype	–	yes	yes	–

Observation 1: It should be noted that in most cases the property editors have a *fixed* predicate, or, in rare cases, the predicate is chosen from some *set*. To have the predicate URI editable directly, i.e. to have its value interaction style *free*, is very uncommon among profile editors and is provided for completeness only.

Observation 2: To follow good practice when expressing RDF, certain alternatives are not encouraged and hence not supported in Annotation Profiles. First, editing blank node identifiers as free text is seldom a good idea, since they are internal identifiers specific to a serialization. Second, having vocabularies of language-coded literals is just nonsensical.

Observation 3: The completeness requirement will be satisfied by allowing creation or removal of all kinds of statement not modification of one statement into another. Hence, there is a property editor for every kind of statement and not a single property editor for all kinds of statement. Alternatively formulated, a property editor is configured once and an enduser is not allowed to change its value type or value interaction style.

Property editors also need to provide limitations or guidance on how the language or datatype is allowed to be changed for literals with language and literals with datatype respectively. Simply put, a property editor can be restricted to a specific language / datatype or list a range of valid languages / datatypes to choose among. The most common situation is for literals with language to have a range of languages and for literals with datatype to have a fixed datatype.

4 AN EDITING EXTENSION TO FRESNEL

Fresnel [1] is a browser-independent display vocabulary for *presenting* RDF models. Let us shortly introduce the basics of Fresnel. The following is quoted from the Fresnel Manual [1]:

"Fresnel introduces lenses and formats. Lenses define which properties of an RDF resource are displayed and how these properties are ordered. Fresnel formats determine how the selected properties are rendered by specifying RDF-specific formatting attributes and by providing hooks to CSS, which is used to specify fonts, colors, margins, borders, and other decorative elements."

The Fresnel manual also lists five design goals:

- *useful for rendering different output formats like HTML, SVG, PDF, plain text, and others,*
- *applicable across different RDF display paradigms, (nested box-based textual representation à la XHTML+CSS, node-link diagram, etc.),*
- *built on existing Web technology,*
- *extensible for more specialized needs,*
- *easy to learn and use.*

All of these goals are good reasons for why an extension of Fresnel was chosen for realizing Annotation Profiles. Other reasons include that Fresnel is a pure vocabulary, defined independently of specific implementations, and there is few alternatives that are RDF-centric. One of the alternatives would have been to formalize the configurations available in the SHAME framework. However, the SHAME initiative is more of a code library than a specification and consequently has achieved less acceptance in the community. A viable path that will be investigated is to extend SHAME so that it conforms to the Fresnel extension.

When fulfilling the requirements of Annotation Profiles in the form of a *Fresnel Editing Extension*, from now on abbreviated as FEE, the following requirements have been considered:

Fresnel Editing Extension must:

- **be complete** – can edit all kinds of RDF.
- **use schema information** – when available and relevant.
- **be well defined** – applications that implement the extension should produce the same RDF-data, given the same fresnel lenses and user input.
- **be consistent** – RDF data *edited* by a set of specific lenses must also be allowed to be *presented* by the same lenses.
- **be backward-compatible** – so that applications that supports only Fresnel Core can use the FEE enriched lenses and formats.

Note that these requirements do not imply which features that should be included, except those needed to achieve completeness. There are features that control e.g. appearance and interaction that has been decided to be included or left out due to best practices of a range of profile editors such as SHAME. Furthermore, the amount of features included might change in later versions.

The above requirements are important when deciding how the new features are to be introduced in FEE, we have found two distinct approaches:

There are two kinds of extensions involved in Fresnel Editing Extension:

- **semantic extension** – according to the requirements of consistency and backward-compatibility we need to specify what the existing Fresnel vocabulary means in terms of editing.
- **vocabulary extension** – in some cases we need additional vocabulary address for needs that are specific to the editing situation. This is according to the requirement of being well defined and complete.

Whether the semantic extension path is taken for a specific feature depends on if it resembles existing vocabulary enough **and** if the backward-compatibility requirement can be fulfilled.

To minimize the amount of configuration when defining lenses and formats, a range of default values have been carefully selected to fit the most common situations.

4.1 The role of Lenses

Lenses are suitable for capturing all of the features in the completeness and structure categories introduced in section 2.

The *completeness category* of features has been found to be satisfied by supporting a range of property editors. In Fresnel there is a class called

PropertyDescription that is used whenever you need to say something explicit about how a property is used in a certain lens. In many cases you can manage without this class by listing the properties via their URIs directly. From the perspective of FEE, *PropertyDescription* is the natural place to express the details of a Property Editor. If no *PropertyDescription* is used, a default Property editor will be used depending on schema information. This is discussed in more detail in section 4.3.

Fresnel Lenses provide a *LensDomain* used as a selection criteria for finding appropriate lenses for a specific resource. FEE will also be used to help decide the value interaction style for the object, i.e. which property editor to use. Furthermore, it defines which statements to generate and later remove. The purpose of these statements are to fulfil the *lensDomain*. If not generated automatically, a resource edited by a specific lens would not be presentable by the same lens. This would break the consistency requirement. Editing a resource with a specific lens can happen either when a user has specified it explicitly, or when it is used automatically as a sublens of another lens.

Another use of *lensDomains* is to generate lists of suitable resources or filter searches when choosing a value for a property, which has a sublens specified.

The *structure category* includes features such as selection and order of properties, which is already supported in Fresnel Lenses via the *showProperties* property. Grouping of properties is not supported in Fresnel but it is easily introduced by allowing lenses to occur in the list of properties pointed to by the *showProperties* property (since Lenses are in principle groupings of properties).

Cardinality constraints are also easily introduced into the *PropertyDescription* class mentioned above.

4.2 The role of Formats

The distinction between Lenses and Formats explicitly states that:

“Fresnel *formats* determine how the selected properties are rendered by specifying RDF-specific formatting attributes and by providing hooks to CSS, which is used to specify fonts, colors, margins, borders, and other decorative elements.”

This is a clear match for the features in the *presentation category*.

Today, the features in the *interaction category* have no correspondence in Fresnel formats or lenses, because the latter was designed for display and not editing. However, the resemblance between the presentation and the interaction category features provides a strong argument for keeping the latter in the format as well.

4.3 The role of RDF Schema and OWL

Fresnel is designed for displaying existing RDF graphs. Obviously, existing statements have fixed objects that are either blank nodes, URIs, or literals possibly with additional language or datatype. When editing, i.e. creating or modifying statements, you need to know in advance the value type of both the predicate and the object. The natural source for this knowledge is RDF Schemas or OWL ontologies. However, neither RDF Schema, nor OWL,

provides a mechanism for distinguishing between blank nodes and URIs or determining whether a literal should have a language or not. Reasonable defaults will suffice in most situations, but FEE will need to be able to provide the distinction when needed.

5 OUTLINE OF FRESNEL EDITING EXTENSION VOCABULARY

The following properties capture features regarding the completeness of editing RDF statements. Properties of features in the other categories are yet to be defined. All of them apply to the `fresnel:PropertyDescription` class. If no namespace is given, it is assumed to be the Fresnel namespace.

Table 2: Completeness of editing RDF statements

<i>Properties</i>	<i>Range</i>	<i>Explanation</i>
objectType	Resource * BNode * URI Literal * Plain * LiteralWithLanguage * LiteralWithData Type	<ul style="list-style-type: none"> ● If there is no OWL ontology or RDF Schema available. ● To provide distinction between BNode and URI ● To provide distinction between plain literals and literals with language
predicateInteraction	PredicateInteraction * Fixed * Set * Free	See section , especially observation 1. This property is not explicitly used. <ul style="list-style-type: none"> ● property points to property →Fixed ● property points to set of properties via selector →Set ● No property →Free
objectInteraction	ObjectInteraction * Free * Set * Other	See section , especially observation 2. RDF Schema or OWL can distinguish between Free and Set , but Other cannot be detected.
language	a language code, or a rdf:Seq containing language codes or: AllLanguages SystemLanguages UserLanguages	A language code is a string according to the RDF-3066 [3]. <ul style="list-style-type: none"> ● If no user languages are defined, the system languages are used instead. ● If no system languages are defined, all languages should be used instead.
range	any instance of <code>rdfs:Class</code> or <code>owl:Thing</code>	If no RDF Schema or OWL ontology is found and a range is needed, this is the way to express it.

6 GLOSSARY AND ACRONYMS

Annotation Profile – a set of Fresnel Lenses and Formats that together constitute the configuration for a profile editor to edit a certain set of resources. An Annotation Profile can be defined so that it specifies how to edit resources according to a specific metadata standard or schema.

Profile Editor – a metadata editor that is not restricted to a certain vocabulary / schema / metadata standard. Instead it has a configuration mechanism that enables it to edit the metadata at hand.

Property Editor – a part of a metadata editor that allows you to edit either the predicate and / or object of a statement. From the perspective of the property editor the subject is a fixed resource.

FEE – abbreviation for Fresnel Editing Extension

7 REFERENCES

- [1] Bizer C., Lee R., Pietriga E., Fresnel - Display Vocabulary for RDF, <http://www.w3.org/2005/04/fresnel-info/manual-20050726/>
- [2] Brickley D., Guha R., RDF Vocabulary Description Language 1.0: RDF Schema, <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
- [3] H. Alvestrand, Tags for the Identification of Languages: RFC3066, <http://rfc.net/rfc3066.html>
- [4] Klyne G., Carroll J., Resource Description Framework (RDF): Concepts and Abstract Syntax, <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- [5] Patel-Schneider P., Hayes P., Horrocks I., OWL Web Ontology Language Semantics and Abstract Syntax, <http://www.w3.org/TR/2004/REC-owl-semantic-20040210/>