Semantic Metadata Interoperability and Inference-Based Querying in Digital Repositories

Dimitrios A. Koutsomitropoulos, University of Patras, Greece

Georgia D. Solomou, University of Patras, Greece

Andreas D. Alexopoulos, University of Patras, Greece

Theodore S. Papatheodorou, University of Patras, Greece

ABSTRACT

Metadata applications have evolved in time into highly structured "islands of information" about digital resources, often bearing a strong semantic interpretation. Scarcely however are these semantics being communicated in machine readable and understandable ways. At the same time, the process for transforming the implied metadata knowledge into explicit Semantic Web descriptions can be problematic and is not always evident. In this article we take upon the well-established Dublin Core metadata standard as well as other metadata schemata, which often appear in digital repositories set-ups, and suggest a proper Semantic Web OWL ontology. In this process the authors cope with discrepancies and incompatibilities, indicative of such attempts, in novel ways. Moreover, we show the potential and necessity of this approach by demonstrating inferences on the resulting ontology, instantiated with actual metadata records. The authors conclude by presenting a working prototype that provides for inference-based querying on top of digital repositories.

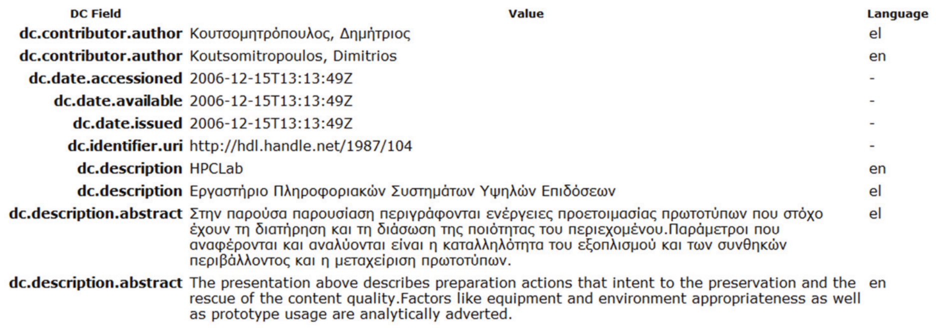
Keywords: *Metadata, Ontologies, Data Semantics, Semantic Web, Knowledge Discovery, Semantic Matching*

INTRODUCTION

Metadata are today one of the most widely adopted paradigms to facilitate description, integration, discovery and preservation of information and resources stored in remote databases or hosted in web-accessed portals and digital libraries. One of the reasons that the Dublin Core (DC) schema is widely adopted in such scenarios is probably its simplicity and its general applicability that makes it suitable for a number of different metadata-intensive applications.

DOI: 10.4018/jitr.2009062903

Figure 1. Detailed item view in DSpace



DC Field	Value	Language
dc.contributor.author	Κουτσομητρόπουλος, Δημήτριος	el
dc.contributor.author	Koutsomitropoulos, Dimitrios	en
dc.date.accessioned	2006-12-15T13:13:49Z	-
dc.date.available	2006-12-15T13:13:49Z	-
dc.date.issued	2006-12-15T13:13:49Z	-
dc.identifier.uri	http://hdl.handle.net/1987/104	-
dc.description	HPCLab	en
dc.description	Εργαστήριο Πληροφοριακών Συστημάτων Υψηλών Επιδόσεων	el
dc.description.abstract	Στην παρούσα παρουσίαση περιγράφονται ενέργειες προετοιμασίας πρωτοτύπων που στόχο έχουν τη διατήρηση και τη διάσωση της ποιότητας του περιεχομένου.Παράμετροι που αναφέρονται και αναλύονται είναι η καταλληλότητα του εξοπλισμού και των συνθηκών περιβάλλοντος και η μεταχείριση πρωτοτύπων.	el
dc.description.abstract	The presentation above describes preparation actions that intent to the preservation and the rescue of the content quality.Factors like equipment and environment appropriateness as well as prototype usage are analytically adverted.	en

In many standard repository configurations (including the DSpace digital repository software) the DC Metadata Element Set (DCMES) is implemented as a flat aggregation of elements. This is also true for qualifiers, which are not always implemented as sub-properties of main elements; rather, they often appear at the same level as parent elements and the sub-element/qualifier relationship is maintained only in the label. This situation, evident also in the DSpace-based University of Patras institutional repository (http://repository.upatras.gr/ dspace/) is depicted in Figure 1.

The semantic interpretation of the DC model that, as we see, is not always represented in applications, is formalized through the DC Abstract Model (DCAM) specification (Powell, Nilsson, Naeve, Johnston, & Baker, 2007) as well as the most recent recommendation for expressing DC in the Resource Description Framework (RDF) (Nilsson, Powell, Johnston, & Naeve, 2008). These documents virtually suggest an ontology of DC, expressed in RDF(S), a Semantic Web standard.

Such a DC ontology bears its own semantic structure that may be taken advantage of, in order to enable more refined descriptions of resources. This of course is reminiscent of the well-known Semantic Web "bootstrapping problem" (Dill, Eiron, Gibson, et al., 2003; Hendler, 2008): The availability of high-quality, complex and interconnected resource descriptions is a key aspect for the Semantic Web to be of some value; on the other hand, the burden to create a whole new set of rich annotations is too high, both from a conceptual (hard to conceive) as well as from an effort (too much time) point of view. It is unlikely that existing implementations, already employing flat descriptions on a large number of resources, would invest in reorganizing from scratch their underlying data model. Even if such a venture is undertaken, the cost for aligning and enriching existing descriptions can be prohibiting.

Having these in mind, we propose an implementation of the DC ontology that is to be carried out in terms of a most centralized approach. To do this we are based on the *semantic profiling* technique, well-applied previously on fully-structured knowledge domains, such as the CIDOC Conceptual Reference Model (CIDOC-CRM) (Crofts, Doerr, & Gill, 2003) and introduced in (Koutsomitropoulos, Paloukis, & Papatheodorou, 2007). Using this technique we try to better capture the intended semantics of the DC metadata domain, having the DC RDF(S) schema as a starting point.

Our goal is to upgrade this ontology up to OWL and OWL 2 level (Parsia, & Patel-Schneider, 2008), by incorporating new constructs and refinements, available only in these languages. At the same time, we build upon the initial model and do not require any alternations in its original specification. The resulting ontology, including the new refinements, is then populated in an automated way from metadata already existing within the live DSpace installation of the University of Patras institutional repository, using the system's OAI-PMH (Open Archives Initiative - Protocol for Metadata Harvesting) interface.

In digital repositories it is often the case that other metadata schemata are employed as well. This is for example, what has been done with the University of Patras repository, where the original DSpace schema has been extended with learning object (LOM) metadata (IEEE LTSC, 2002). Therefore, during the process of creating the repository's ontological model, we also investigate potential semantic relationships and mappings between DC-based information and LOM metadata, in order to achieve *semantic interoperability*, also between disparate schemata.

As a result of the aforementioned approach, we are able to afford queries on the instantiated ontology model that: a) Show the same retrieval capabilities as the original model, i.e. we can make similar queries and retrieve identical results and b) acquire implicit information which is impossible to retrieve using the existing configuration, by taking advantage of the richer semantic constructs of OWL 2 (like, for example, role-chains). Finally, to prove our concept, we develop a concrete, working prototype that provides for inference-based search and navigation on top of the DSpace repository system.

This article is further organized as follows: First we give an overview of the current DC implementation in RDF and its underlying semantics. We examine existing approaches for expressing DC in OWL and discuss options for coping with identified problems. We proceed by describing our process for creating a DC Web ontology: we present our implementation steps and provide indicative examples of our elaborations. We then comment on the capabilities enabled by presenting a series of reasoning-based queries that can be conducted on this new ontological model. Further, we briefly introduce the prototype we have developed and its features, namely semantic search and navigation amongst a repository's instances. Finally, the last section summarizes the conclusions of our work.

ISSUES IN CREATING A DUBLIN CORE ONTOLOGY

Before moving on to the construction of an ontology out of the DC metadata schema, it is important to take a closer look at how this particular schema's semantics could be adopted in a digital repository and to what extent are they being represented and exploited. It appears that the self-descriptive nature of DC schema can easily render a DC-based ontology undecidable; therefore, we try to answer this decidability problem and discuss how punning relates to its solution.

Dublin Core RDF Implementations and Semantics

An effort to capture the DC inherent semantics was initiated back in 2005 with the specification of an "Abstract Model" for DC (DCAM). DCAM, which is in fact a meta-modeling abstraction bearing the RDF(S) semantics, was last revised in June 2007. The equivalence between some of the notions in the DC Abstract Model and the corresponding RDF notions is given in (Powell, Nilsson, Naeve, Johnston, & Baker, 2007) and is presented in the following table (Table 1).

Although DC was already specified through XML and RDF syntaxes, it took almost six years for the DCAM to be fully incorporated and implemented in these specifications (Nilsson, Powell, Johnston, & Naeve, 2008). This brand-new recommendation for expressing DC in RDF, that takes into account the DCAM constructs, comes to face a number of problems identified with the legacy specifications, including (Nilsson & Baker, 2008):

- The existence of two legacy expressions of DC metadata in RDF, namely *simple* DC (Beckett, Miller, & Brickley, 2002) and *qualified* DC (or DC *Terms*) (Kokkelink, & Schwänzl, 2002).
- Conflicting recommendations in the above documents regarding the use of literal strings in DC metadata.

Table 1. Correspondence between DCAM and RDF(S)

DCMI Abstract Model	RDF/RDFS
Resource	Class `rdfs:Resource`
property or *element*	Class `rdf:Property`
Class	Class `rdfs:Class`
syntax encoding scheme	Class `rdfs:Datatype`
has domain relationship	Property `rdfs:domain`
has range relationship	Property `rdfs:range`
sub-property of relationship	Property `rdfs:subPropertyOf`
sub-class of relationship	Property `rdfs:subClassOf`
plain value string	Plain literal
typed value string	Typed literal

- Implementation complexity due to the use of idiosyncratic constructs.
- Difficulties in providing formal ranges and domains for DC properties.

This recommendation tries to address these issues by:

- Defining DC elements (simple DC) and qualifications (DC Terms) as RDF properties.
- Providing formal domain and range restrictions (Powell, Johnston, & Baker, 2008).
- Defining RDF classes for domains and ranges and organizing them in hierarchies.
- Allow both datatype- and object- properties (in OWL terminology) based on the specified range.

Therefore, it appears that the DCAM along with its new implementation try to bring the DC model closer to the Semantic Web reality.

This lack between a web-oriented resource description standard and the Semantic Web has also been previously recognized. The AIFB institute at University of Karlsruhe maintains an implementation of the DC in OWL (http://www.aifb.uni-karlsruhe.de/WBS/rvo/ontologies/). In it, an incremental approach is followed, where finally the full underlying semantics of DC are accommodated, rendering unfortunately the ontology undecidable (since it is in OWL Full). In particular:

- Properties are defined, that are used then for the description of properties and classes.
- These properties are defined as annotation properties.
- The DCMI Type vocabulary elements are defined as instances of the DCMIType class.
- Finally, in the full version, the annotation properties are organized in a hierarchy and the vocabulary elements are defined also as classes, making the ontology OWL Full.

The problem with this technique is that individuals, properties and classes are no more disjoint, a major requirement for the decidability of OWL 1.0 ontologies. Keeping properties as annotations is also problematic, since, while desirable in self-descriptions, this has poor semantic interpretation and would contribute very little, if at all, in a reasoning process.

In a similar effort, DRC Inc. proposes an OWL DC ontology (http://orlando.drc.com/semanticweb/ Topics/Ontology/Ontologies.asp) where the DMCES core properties (i.e. without qualifications) are implemented as datatype properties. In this way the model semantics are better captured, however properties fillers are deemed to be plain literals.

Apparently, none of these implementations does provide semantics richer that the ones inherent in RDF(S). Also, DRC actually implements a new DC model defined under a new namespace (dces-ont). In our approach, we try to maintain and reuse the original schema by adding (importing) an optional mediating semantic profile in an attempt to literally capture the implied semantics of DC.

Retaining Ontology Decidability

As mentioned previously, it is mostly the self-referencing properties that cause a DC ontology to be in OWL Full. However, even intuitively, this appears too much a price comparing to what the benefits are, that is, the mere textual annotation of properties and classes:

- Properties that describe properties.
- Classes as instances of other classes.

In the first case it is easy to see that it is of little interest to the end-user the date, for example, when a particular DC property changed recommendation status. Thus, it is more natural to consider such characteristics as textual annotations.

In the second case, this is a clear meta-modeling requirement and hardly ever the double nature of a class (both as instance *and* a class) would be necessary for useful conclusions. In both cases, the notion of *punning* that comes with OWL 2 appears as a more elegant solution to the problem, keeping at the same time the decision complexity low.

Punning is a syntactic convention (Golbreich, & Wallace, 2008; Patel-Schneider, & Horrocks, 2007) based on which a name defined in an OWL 2 ontology can be interpreted as a class, property or individual simultaneously, depending on its use. For example, the `Agent` class defined in the DCAM as an *instance* of `AgentClass` leads to the appearance in conforming implementations of a class *and* an individual under the same name 'Agent'. Punning has no further semantic implications (it is not considered at all in the underlying description logic): for example, a complying reasoner has not necessarily to infer that `Agent` is also a *subclass* of the `AgentClass`.

Further, property names can also be considered as instances. This implies that another property could be used to characterize the property name/instance. For example, the DC Terms property `issued` can assign a date to the property `subject` while reasoning remains sound and complete.

Therefore, in order to overcome this obstacle, the following four options seem to be available:

1. Duplicate DC properties (elements) definitions under separate namespaces.
2. Rely solely on OWL 2 punning.
3. Define all properties as annotations.
4. Mix the above techniques.

Solution (3) is in fact the one followed by AIFB and, as already mentioned, it has poor semantic interpretation.

Option (4) means to identify most problematic properties (i.e. those that are mostly used in self-references, but only assign literal values) and define them as annotations or re-define them under separate namespaces. All the rest can be defined as object/datatype properties. Remaining conflicts are left to be resolved by punning.

Solution (2) means to let punning decide upon ambiguities, but our experimentation reveals that this has unstable behavior in supporting implementations (e.g. in Protégé 4 that also has *inherent* DC annotations).

Solution (1) may appear definitive, but is quite risky since the definition of new namespaces jeopardizes the applicability of the schema and is against the whole profiling philosophy. Also this has to be done in cost of backward compatibility problems. Despite these limitations, the new DC recommendation follows this particular solution, at least for properties definitions. In particular, traditional `dc:` elements such as `contributor` and `creator` are duplicated under the existing `dcterms:` namespace. `dc`: elements would accept literal values, while `dcterms:` properties will be filled with non-literal objects (individuals).

During the creation and profiling of our DC ontology we inevitably adopt the aforementioned solution, since it has been officially endorsed by the DC Metadata Initiative; however we see that punning as it is currently supported, can sufficiently deal with ambiguous classes such as Agent. In addition, `rdf:Property` can be interpreted as either datatype- or object-property, depending on its use.

CREATION OF AN ONTOLOGICAL MODEL FOR REPOSITORY METADATA

In this section we give an account of how we actually develop a Semantic Web ontology out of repository's metadata. First, we give an overview of the semantic profiling technique and indicate how it can be adapted for constructing rich ontologies out of semi-structured sources as well, which is often the case in digital repositories. Then we document in some detail the steps that actually implement this ontology. We also take into consideration how the LOM schema can be incorporated in the resulting model.

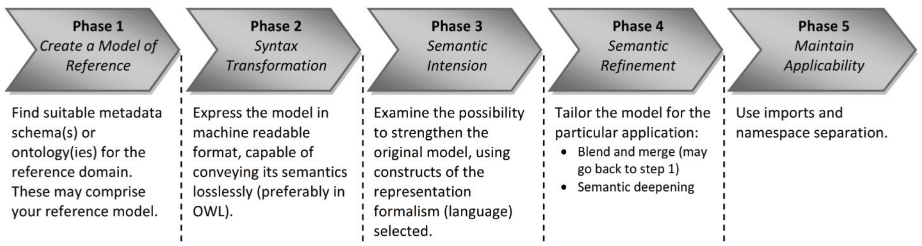
The Semantic Profiling Technique

Metadata application profiles form a well-known paradigm for tailoring and mixing metadata schemata in order to accommodate particular application's needs (Heery, & Patel, 2000). For semantics-intensive applications however, as for example reasoning-based knowledge discovery, and to ensure semantic interoperability, a richer level of semantic correlations is necessary, as is the case with Semantic Web ontologies.

The semantic profiling technique has been recently introduced as a means to profile a knowledge domain schema, not only by mixing and matching, but also by narrowing its interpretation in accordance to a particular application or usage scenario. The technique starts with a domain schema, represented in an adequately expressive language, such as RDF(S) and above (OWL, OWL2), and then predicts a series of steps in order first, to make the domain model as specific as possible, without affecting its interoperability (*semantic intension*); and second, to further narrow the model, this time in regard to the application per se (*semantic refinement*). The technique is outlined in Figure 2.

Although semantic profiling is more than a mix and merge, this does not mean that the technique overlooks the merit of traditional application profiles. When refining for the application, apart from the semantic narrowing, it is often necessary to consider other schemata or existing ontological models that could contribute in a more thorough representation of the application domain. It is in this sense that

Figure 2. The semantic profiling process


Phase 1
Create a Model of Reference
Phase 2
Syntax Transformation
Phase 3
Semantic Intension
Phase 4
Semantic Refinement
Phase 5
Maintain Applicability
Find suitable metadata schema(s) or ontology(ies) for the reference domain. These may comprise your reference model.
Express the model in machine readable format, capable of conveying its semantics losslessly (preferably in OWL).
Examine the possibility to strengthen the original model, using constructs of the representation formalism (language) selected.
Tailor the model for the particular application:
• Blend and merge (may go back to step 1)
• Semantic deepening
Use imports and namespace separation.

the semantic refinement phase may revisit the first phase in the technique. As far as the repository's domain is considered, we also look into the integration of the LOM schema with the ontological model, which is developed mainly out of Dublin Core metadata.

Finally, care has to be taken in order to maintain backwards compatibility and interoperability of the resulting model. For this, it is best to follow an incremental procedure, where every phase adds a more detailed semantic layer over the previous one. Each of these layers can, and actually is, physically and logically distributed; the use of namespaces keeps definitions separate, while the `owl:imports` directive (Motik, Patel-Schneider, & Parsia, 2008) unifies distributed ontologies in a transparent knowledge base.

It is important to note that semantic profiling takes for granted that a suitable domain model already exists or it can be constructed, and that this model is already instantiated in machine readable format. As XML-based metadata schemata are often monolithic, in the sense given in the introduction, their implementations tend to describe their underlying domain in a semantically semi-structured manner.

As a result, an important contribution in this article is the way we propose in order to translate and extract semantic relations and other information from flat metadata. In this fashion, we take the semantic profiling technique further, adjusting it for semi-structured knowledge domains, while ontological descriptions and instances are automatically populated from existing metadata.

Therefore in the following, it is worth examining in some detail the application of this technique to a digital repository's model and metadata, as this may not be a trivial procedure. In addition, the syntax transformation phase is, in this context, not a mere transcoding of a model from a Semantic Web language to another, but indeed an effort to dynamically construct a model, by identifying and making explicit semantic knowledge out of XML sources.

Syntax Transformation

As already pointed out, repository's metadata are not adequate of forming a well-structured knowledge domain on their own as, for example, the one expressed by CIDOC-CRM; indeed, our instances are virtually a flat aggregation of elements. The DC RDF implementation may be a good starting point, but there is still a gap to be filled in order to reach this state. Therefore, we need a transition from an absent semantic model to an item-property-predicate situation, thus invoking the RDF semantics. This transition is offered by an XSLT which can be considered part of the syntax transformation phase in the semantic profiling process.

This transformation amounts to:

a. Minimal syntax transformation to "clean-out" OAI overhead statements. For example XML elements `<responseDate>` and `<request>` are OAI-specific and are not required. The same holds for records headers.
b. Assign types (datatypes) to literal values. DC offers an abstraction for datatypes, called *syntax encoding schemes*. These schemes are in fact equivalent to the actual XML Schema datatypes that are more likely to be supported in OWL applications (Motik, Patel-Schneider, & Parsia 2008). For example, `dcterms:W3CDTF` corresponds to `xsd:date, dcterms:RFC3066 to xsd:language` and so on.
c. Reify literals (as objects) originating from specific (controlled) vocabularies, such as MIME types. In particular, DSpace relates items with literals corresponding to MIME types through the `dcterms:format` property. The use of MIME types as fillers to `dcterms:format` is also a DCMI recommended practice (DCMI Usage Board, 2008).
d. Reify other literals to be able to identify and express potential semantic relations. For example, DSpace author names are defined as objects and their comma separated first and last names (as stored by DSpace)

are parsed and assigned to corresponding properties from the popular FOAF ontology (http://xmlns.com/foaf/ spec/).

e. Introduce and re-assign namespaces. This means that `dc:` is replaced by `dcterms:` and also `dspace-ont:` for DSpace specific elements is introduced (see Table 2).

The replacement of the `dc:` namespace with `dcterms:`, even for traditional DC elements, is unavoidable in order to achieve compatibility with the DC RDF implementation, discussed in the previous section.

The result of this phase is the creation of an RDF document that contains triples describing the repository resources, mixed with the XML Schema and FOAF namespaces; let it be 'instances.rdf'.

Semantic Intension

Second, we can commence with the semantic intension of the DC model. To do this we need to establish (or devise) a core model first. Such a model is:

- offered by the RDF semantics, inherent in the DC RDF implementation,
- further profiled by the new DC implementation that incorporates DCAM as well as domain and range restrictions,
- further profiled by post-RDF characterizations of properties, like inversity, symmetry, transitivity, functionality. For example, we define `dcterms:relation` as symmetric and `dcterms:hasPart` as the inverse of `dcterms:isPartOf`.

The result of this phase is the creation of an OWL document that imports the original DC Terms RDF implementation document ('dc-terms.rdf') and contains the above refinements (e.g. 'dc-ont.owl'). This document comprises the DC ontology, expressed in OWL format, which refines the original specification by utilizing OWL-specific constructs, while retaining at the same time its interoperability.

Semantic Refinement

Now, it is time to add semantic refinements for our particular application scenario, i.e. the University of Patras institutional repository. This is conducted by:

a. Model vocabularies in subsumption hierarchies. For example, the MIME type vocabulary that DSpace supports is implemented as a partition of subclasses, with instances, and is related to `dcterms:FileFormat`.
b. Identify and represent the DSpace notions of 'item', 'collection' and 'community' as classes. 'Collection' is further defined as a subclass of 'community', conveying the fact that, in DSpace, collections refine communities. Items relate to collections through the `dcterms:isPartOf` property.

Table 2. Example mapping of some DC elements to DC Terms properties

DC Element	DC Terms Property
`dc.contributor`	`dcterms:contributor`
`dc.contributor.author`	`dspace-ont:author`
`dc.description.abstract`	`dcterms:abstract`
`dc.format`	`dcterms:format`
`dc.description`	`dcterms:description type="http://www.w3.org/2001/XMLSchema#anyURI"`
`dc.type`	`dcterms:type type= "dspace-ont:dspacetype"`

Table 3. Mapping of LOM elements to DC Terms properties

LOM Element	DC Terms Property
`lom.annotation`	`dcterms:description`
`lom.context`	`dcterms:educationLevel type="lom:context"`
`lom.difficulty`	`dcterms:type type="lom:difficulty"`
`lom.intendedenduserrole`	`dcterms:audience type="lom:intendedenduserrole"`
`lom.interactivitytype`	`dcterms:instructionalMethod type="lom:interactivitytype"`
`lom.learningresourcetype`	`dcterms:type type="lom:learningresourcetype"`
`lom.typicallearningtime`	`dcterms:extent type="lom:typicallearningtime"`
`lom.status`	`dcterms:type type="lom:status"`
`lom.version`	`dcterms:hasversion`

c. Identify and represent the non-DC notions of '`author`' and '`sponsorship`' and relate them to the initial model: we define, under a different namespace, author and sponsorship as OWL object properties, and make them sub-properties of `dcterms:contributor` and `dcterms:description` respectively.
d. Model complex relations using role-forming operators and restrictions available in OWL 2: For example, we define the notion of 'co-author', in Manchester Syntax (see next section), as

```
inv(author) o author SubPropertyOf
            co_author
```

and refine sponsorship by

```
inv(dcterms:contributor) o sponsorship
    SubPropertyOf sponsorship
```

meaning that authors of items are also receiving sponsorship from the same institution. We also state that:

```
not(item) and (dcterms:hasPart some
  item) SubClassOf collection
```

meaning that everything that 'hasPart' an item is a collection, unless it is an item itself. In addition, collections may only have items as parts:

```
collection SubClassOf (dcterms:hasPart
          only item)
```

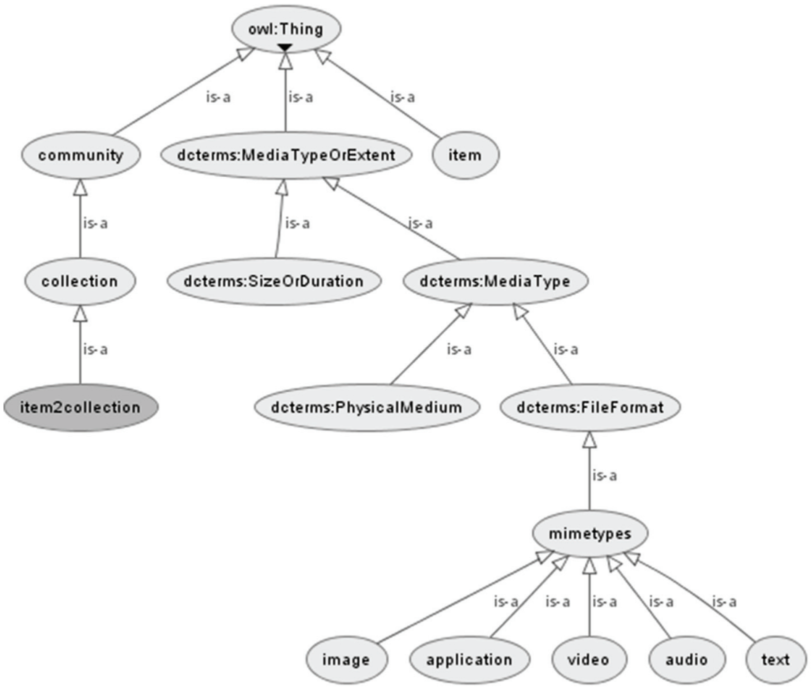
This phase results in creating a new OWL 2 document ('dspace-ont.owl') containing OWL constructs that refines the DSpace data model in a semantic manner and imports all other documents (Figure 3).

Incorporating the LOM Metadata Schema

At this point, the semantic profiling technique foresees the need to accommodate other additional schemata that may be found suitable for a more precise representation of our particular application's problem domain. As mentioned, in the University of Patras DSpace installation, we have also enriched the original DC schema with LOM metadata in a way to better convey the educational characteristics of resources.

Therefore we may jump back to the syntax transformation phase and we need to consider a potential correspondence between LOM elements and DC Terms properties. This mapping, presented in Table 3, is based on the LOM to simple DC mapping suggested in (IEEE LTSC, 2002) and on the semantic interpretation of both the LOM and qualified DC elements. LOM concepts are prefixed with the `lom:` namespace. Recently, such an attempt has also been presented in (IEEE LTSC, 2008), where

Figure 3. Partial view of the DSpace ontology class hierarchy (excluding some imported axioms)



a potential LOM to DCAM mapping is being considered.

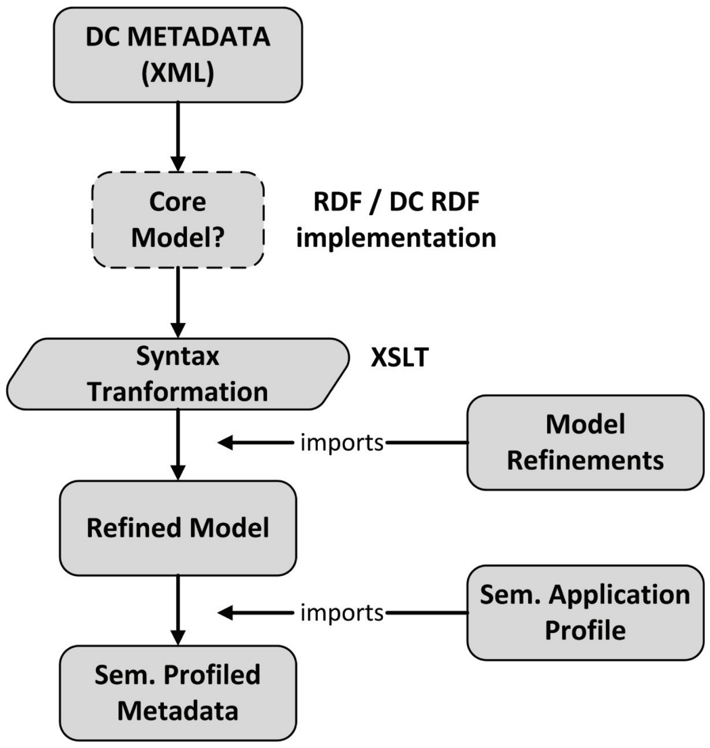
In addition, we create a new ontology model representing (part of) the LOM schema: we construct OWL classes grouping LOM vocabulary values and, when it is semantically consistent, we relate them to existing DC Terms classes (domains and ranges). For example we declare `lom: typicalLearningTime` to be a subclass of `dcterms:SizeOrDuration`. These classes also serve as a means of automatically giving types to LOM property values during syntax transformation, in accordance to the property's recommended value space (IEEE LTSC, 2002). In Table 3, the `type` attribute specifies exactly which non-literal type the property value must have. The LOM-specific constructs are kept in a separate document ('lom.owl').

The implementation steps described above are outlined in Figure 4. All documents are available at: http://ippocrates.hpclab.ceid.upatras.gr:8998/dc-ont.

TECHNICAL CONFIGURATION AND RESULTS

In the previous sections we have described the process for creating the ontological model, by constructing a semantic application profile of the qualified DC ontology, tailored for our repository's domain. The next step is to populate this ontology with descriptions coming from the University of Patras institutional repository. We then experiment upon the instantiated ontology model by carrying out a number of inference-based queries. The results are indicative of knowledge acquisition using Semantic Web ontologies. They also suggest that such a semantic-enabled search can be fruitful in digital repositories, as compared to existing, keyword-based search methodologies.

Figure 4. Implementation steps of the repository's ontology



Ontology Population

DSpace offers the ability to harvest its metadata by providing an OAI-PMH interface. Internally it follows a particular application profile, borrowing heavily from the Library Application Profile (DCMI-Libraries Working Group, 2004). Including qualifications, DSpace fosters a total of 66 elements, not all of which are visible to the user.

OAI-PMH is an HTTP based protocol for interoperable metadata harvesting (Lagoze, Van de Sompel, Nelson, & Warner, 2002), and OAI-compliant harvesting interfaces and service providers are becoming common in digital repositories and libraries. Though it would be easier to access the repository's database directly, we opted for OAI, precisely in order to show in practice how our approach can be generalized and maintain the interoperability of our implementation.

The repository's OAI-PMH facility is configurable as to what elements are to be exported (including their namespace and label) and also supports simple DC (oai_dc) as well as its qualifications (qdc). Part of the syntax transformation phase is implemented exactly in this configuration: First, we configure OAI to export metadata also in qualified DC format, as richer from a semantic point of view. Then, in the appropriate configuration file, we re-assign namespaces and map LOM and simple DC elements to corresponding DC Terms properties (step (e) of the syntax transformation phase). As a result, we are able to harvest as much of the qualified DC elements as possible and achieve at least a *syntactic* level of interoperability with our ontology model.

A series of syntax transformations, which we have implemented in an XSLT file and described in the previous section, tries to better capture the semantic relations implied in these

Table 4. Mapping of Description Logics Symbols in Manchester Syntax

OWL Expression	Description Logics Symbol	Manchester Syntax
`someValuesFrom`	∃	`some`
`allValuesFrom`	∀	`only`
`hasValue`	∋	`value`
`minCardinality`	≥	`min`
`cardinality`	=	`exactly`
`maxCardinality`	≤	`max`
`intersectionOf`	⊓	`and`
`unionOf`	⊔	`or`
`complementOf`	¬	`not`
`SubPropertyChain`	∘	`o`

metadata as well as to construct the OWL specific instantiations, thus achieving a *semantic* level of interoperability with our ontology. Indeed, a web service is responsible for the dynamic construction of the populated ontology, by automatically harvesting the repository's metadata to an XML file through OAI and transforming it according to the XSLT document. The resulting OWL document imports the semantic application profile ('dspace-ont.owl') as well as the LOM-specific constructs ('lom.owl'). This service's response can then be fed to any application capable of managing and querying ontologies, like the Protégé environment.

Conducting Queries

In the following we present a series of "intelligent" queries, posed to the produced OWL 2 document. One way to do this is to use the DL Query Tab of Protégé 4.0 by taking advantage of its bundled reasoners, i.e. FaCT++ and Pellet.

Queries can be expressed in a "less logician like" and more user-friendly syntax, known as the *Manchester Syntax* for OWL 2 (Horridge & Patel-Schneider, 2008). Manchester Syntax maps Description Logics symbols to English words and phrases, as shown in Table 4.

Queries are in the form of ontological class names, class expressions or Boolean combinations of class expressions. When a user poses a query, a parser maps the given expression into a concept expression (class). Consequently, all instances classified by the reasoner under this particular concept are retrieved.

All queries mentioned below, are summarized in Table 5 and we reference them using their corresponding row number.

First we conduct a string-based query that would also be possible from inside the standard DSpace querying interface. In particular, we ask for the items authored by an author having a particular surname (Query 1).

By conducting some similar inferences including conjunctive ones (e.g. Query 2), we confirm that it is possible to produce identical results and conclude that our ontology is at least semantically equivalent to the existing model. However, google-like searches and string wildcards, common in DSpace, are not supported in this configuration, since none of the bundled reasoners appears to work with the XML Schema `pattern` facet.

Nevertheless, the main advantage of inference-based querying is that it permits the expression of queries that are impossible to be conducted just by keyword-based search. What's more, it allows retrieval of *more* as well

Table 5. Example Queries

	Query	Ask for:
1	`dspace-ont:author` **`some`** `(foaf:surname` **`value`** `"Dawson")`	*Items authored by a particular author surname*
2	`dcterms:type` **`value`** `dspace-ont:Book` **`and`** `dcterms:audience` **`value`** `lom:Student`	*Items of a specific type (Books) that have intended end user role (audience) 'Student'*
3	`dcterms:format` **`some`** `dspace-ont:image`	*Items that contain image files*
4	`dspace-ont: sponsorship` **`value`** `dspace-ont: Hellenic_Ministry_of_Culture`	*Items/authors that draw sponsorship from a specific institution*
5	**`inv`**`(dspace-ont: author)` **`some`** `(dcterms:format` **`min`** `2 owl:Thing)`	*Authors of items that have at least two different formats*
6	`dspace-ont:co_author` **`some`** `(foaf:name` **`value`** `"Bekiari")`	*The co-authors of an author*

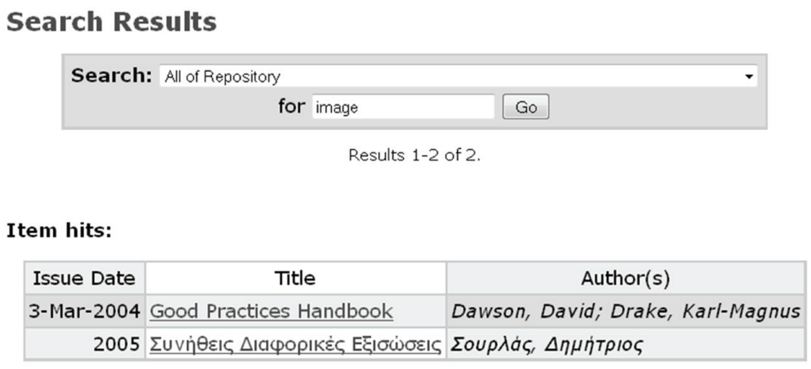
as *more precise* results. These results may be implied by the current data model, but there is no way to retrieve them using the standard configuration.

As an example, suppose we would like to retrieve all items that contain image files. Searching with the keyword 'image' in the traditional repository search returns 2 items (Figure 5). However, examining each of these items metadata reveals that only the latter has actually an 'image/gif' format. The other has a format of 'application/pdf'; the reason why it is returned by traditional search is that DSpace searches also inside the document's text and happens to meet the word 'image'.

On the other hand, the corresponding query in Manchester Syntax (Query 3) fetches just one item, exactly the one that has format 'gif'. In this manner, this approach achieves to improve the *precision* of retrieval by fetching more accurate results.

Suppose now we would like to find out who draws sponsorship from a specific institution, for example, what is funded by the 'Hellenic Ministry of Culture' (Query 4). Searching with these keywords through traditional search returns an item that includes this organization's name in its 'sponsorship' metadata field. Inference-based search however retrieves also the author of this item, aside from the item itself. This is a direct consequence of the role-chain we have declared in our ontology, as described in step (d) of the semantic refinement phase, and

Figure 5. Simple search in DSpace about the keyword 'image'



Search Results

Search: All of Repository
for image Go

Results 1-2 of 2.

Item hits:

Issue Date	Title	Author(s)
3-Mar-2004	Good Practices Handbook	*Dawson, David; Drake, Karl-Magnus*
2005	Συνήθεις Διαφορικές Εξισώσεις	*Σουρλάς, Δημήτριος*

confirms that our approach can also improve the *recall* of retrieval.

In the 'image' example above, semantic-based search is able to fetch the particular item, despite the fact that its format is declared just as 'gif' (i.e. it does not contain the keyword 'image'). This is because in our ontology, 'gif' is an instance of the 'image' class and thus the underlying reasoner is able to conduct an *inference*; that is, since we ask for an 'image' format, we also ask for every instance of this class.

This knowledge discovery capability can also be determined by asking, for example, for the authors of those items (Query 5) that have at least two different formats, using a cardinality restriction on `dcterms:format`. It is easy to see that such a query is impossible to be expressed through traditional search. Similarly, with Query 6 we ask for the co-authors of an author, based on her surname. Due to the definition of the `co_author` property this request becomes possible and the result is straightforward.

We can also perform queries involving the other semantic refinements we have introduced. All above results lead us to the conclusion that the proposed approach enables better information discovery with little manual intervention.

PROTOTYPE DEPLOYMENT IN DSPACE

A step forward in this work was the implementation of an extensible semantic search and navigation facility on top of DSpace. The provided facility is designed to be independent of the underlying system, following a "plug-in" philosophy. In combination with the ontology creation and population process, this facility could semantically enable any web-based digital repository system. It mostly revolves around two main services: *Semantic-enabled search* of DSpace content and *semantic navigation* amongst the repository's instances.

The semantic search service is provided through a simple interface, which, in collaboration with the appropriate inference engine, allows for the construction, submission and evaluation of a semantic query. Queries are typed as simple text in the provided textarea field whereas their formulation is based on the Manchester OWL Syntax. Since users don't know in advance the contents of the ontology, an auto-complete facility is provided as well. Retrieved results are displayed here in the form of a list.

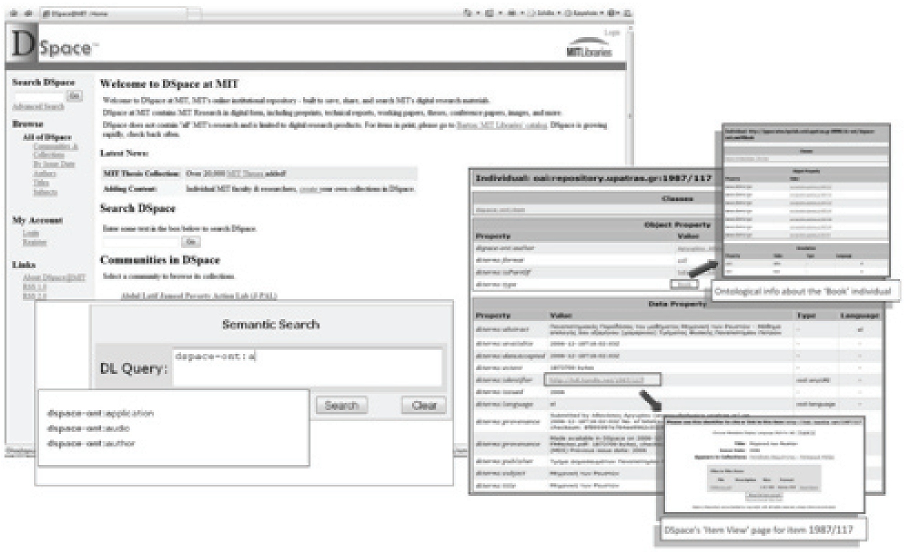
The semantic navigation interface is where detailed ontological information about a selected entity (individual) is presented. More specifically, the main role of this interface is to give a detailed reference to the individual's ontological information, regarding the following:

- the *Classes* to which the selected individual belongs,
- the *Object Properties* that relate the current individual to other individuals,
- the *Data Properties* and *Annotation Properties* that reveal the individual's relations with text (literal) values, which don't correspond to individuals.

The underlying structure of the implemented semantic search and navigation facility is further complemented by the *Ontology Population* module which refers to the dynamic construction of the ontology. The latter comes from DSpace's OAI harvested metadata, after applying the appropriate XSLT transformation on them, a process already described in previous sections.

The populated ontology is dynamically constructed and silently fed to the reasoner over HTTP. In fact, the semantic search and navigation services are designed and implemented in such a way, that they can work with any OWL document, not just the one populated with the repository's metadata. Furthermore, the user interface is easily parameterized to ask for an ontology URL, thus making our implementation totally independent of the specific ontological model. This 'plug-in' architecture is further enhanced by the fact that, in the implementation,

Figure 6. Semantic Search and Navigation Service in DSpace



we avoid any references to DSpace-specific code or any direct access to DSpace's database. In this way we ensure that our services are also system-independent.

The prototype has been successfully deployed on top of the official University of Patras repository. An extensive description of the prototype is out of the scope of this article; however, further information and source code can be found at http://wiki.dspace.org/index.php/User:Kotsomit.

Finally, an intuitive view of the provided semantic search and navigation facility is depicted in Figure 6.

CONCLUSION

Semantic Web can offer a new and challenging dimension in the way information is managed and manipulated by traditional digital repositories. The representation of metadata as Web ontologies can take advantage of their underlying logical framework: First, this enables the possibility to answer queries based on automated reasoning that would otherwise be impossible; on the other hand, it enriches existing descriptions by revealing or making explicit the implied semantic relations between resources, now regarded as ontological instances.

While it can be costly to produce rich semantic descriptions from scratch, we have shown that we can achieve semantic enrichment of existing flatly described resources in a centralized manner; in such a way the burden on content curators and end-users is alleviated and a potential solution to the Semantic Web "chicken-egg" problem (Hendler, 2008) is suggested.

The application of the semantic profiling technique in a domain with little semantic structure as the one underlying flat metadata implementations (in addition to fully semantically structured domains as the CIDOC-CRM) can also produce added value in terms of knowledge discovery and semantic interoperability. The latter is achieved exactly by "unleashing" the repository's metadata in web-accessible OWL format, but most importantly, by the "upgrade" of this metadata in concrete semantic structures. Moreover, the incorporation of the LOM schema into the ontological model confirms that new metadata concepts can be seamlessly integrated, thus further enhancing the interoperability of our approach.

Finally, we give evidence that a semantic search facility can improve traditional keyword search, as it is currently supported in existing methodologies, in at least two ways: First, by retrieving more items (improved *recall*) and second, by fetching more semantically accurate results (improved *precision*). The feasibility of integrating such a facility within digital repositories and other information systems is verified by the development and deployment of a concrete prototype on top of a real repository installation.

ACKNOWLEDGMENT

Part of this work has been conducted under the University of Patras "Operational Program for Education and Initial Vocational Training" (ΕΠΕΑΕΚ II), funded by the Hellenic Ministry of Education and the European Commission.

REFERENCES

Beckett, D., Miller, E., & Brickley, D. (2002). Expressing Simple Dublin Core in RDF/XML. *DCMI Superseded Recommendation*. Retrieved January 20, 2009, from http://dublincore.org/ documents/2002/07/31/dcmes-xml/

Crofts, N., Doer, M., & Gill, T. (2003). The CIDOC Conceptual Reference Model: A standard for communicating cultural contents. *Cultivate Interactive, 9*. Retrieved January 12, 2009, from http://www.cultivate-int.org/issue9/chios/

DCMI-Libraries Working Group. (2004). *Library Application Profile*. DCMI Working Draft. Retrieved January 20, 2009, from http://dublincore.org/ documents/2004/09/10/library-application-profile/

DCMI Usage Board. (2008). DCMI Metadata Terms. *DCMI Recommendation*. Retrieved January 22, 2009, from http://dublincore.org/documents/dcmi-terms/

Dill, S., Eiron, D., Gibson, D., et al. (2003). SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. *12th International Conference on World Wide Web* (pp. 178–186).

Golbreich, C., & Wallace, E. (2008). *OWL 2 Web Ontology Language: New Features and Rationale*. Retrieved January 9, 2009, from http://www.w3.org/TR/2008/WD-owl2-new-features-20081202/

Heery, R., & Patel, M. (2000). Application Profiles: Mixing and Matching Metadata Schemas. *Ariadne, 25*. Retrieved February 3, 2009, from http://www.ariadne.ac.uk/ issue25/app-profiles/

Hendler, J. (2008). Web 3.0: Chicken Farms on the Semantic Web. *Computer*, *41*(1), 106–108. doi:10.1109/MC.2008.34

Horridge, M., & Patel-Schneider, P. F. (2008). *OWL 2 Web Ontology Language: Manchester Syntax*. Retrieved January 28, 2009, from http://www.w3.org/TR/2008/WD-owl2-manchester-syntax-20081202/

IEEE LTSC. (2002). *Draft standard for Learning Object Metadata (LOM) (IEEE 1484.12.1-2002)*. Retrieved February 3, 2009, from http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

IEEE LTSC. (2008). *Draft Recommended Practice for Expressing IEEE Learning Object Metadata Instances Using the Dublin Core Abstract Model (IEEE P1484.12.4/D1)*. Retrieved February 3, 2009, from http://dublincore.org/educationwiki/DCMIIEEELTSCTaskforce?action=AttachFile&do=get&target=LOM-DCAM-newdraft.pdf

Kokkelink, S., & Schwänzl, R. (2002). *Expressing Qualified Dublin Core in RDF/XML*. DCMI Proposed Recommendation. Retrieved January 20, 2009, from http://dublincore.org/documents/2002/04/14/dcq-rdf-xml/

Koutsomitropoulos, D., Paloukis, G., & Papatheodorou, T. (2007). From Metadata Application Profiles to Semantic Profiling: Ontology Refinement and Profiling to Strengthen Inference-based Queries on the Semantic Web. *International Journal on Metadata . Semantics and Ontologies*, *2*(4), 268–280. doi:10.1504/IJMSO.2007.019445

Lagoze, C., Van de Sompel, H., Nelson, M., & Warner, S. (2002). *The Open Archive Initiative Protocol for Metadata Harvesting*. Retrieved February 12, 2009, from http://www.openarchives.org/OAI/openarchivesprotocol.html

Motik, B., Patel-Schneider, P. F., & Parsia, B. (2008). *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. Retrieved February 3, 2009, from http://www.w3.org/TR/owl2-syntax/

Nilsson, M., & Baker, T. (2008). *Notes on DCMI specifications for Dublin Core metadata in RDF.* Retrieved January 20, 2009, from http://dublincore.org/ documents/dc-rdf-notes/

Nilsson, M., Powell, A., Johnston, P., & Naeve, A. (2008). *Expressing Dublin Core metadata using the Resource Description Framework (RDF)*. Retrieved January 20, 2009, from http://dublincore.org/documents/dc-rdf/

Parsia, B., & Patel-Schneider, P. F. (2008). *OWL 2 Web Ontology Language: Primer.* Retrieved January 28, 2009, from http://www.w3.org/TR/owl2-primer/

Patel-Schneider, P. F., & Horrocks, I. (2007). *OWL 1.1 Web Ontology Language Overview.* Retrieved January 28, 2009, from http://www.webont.org/owl/1.1/overview.html

Powell, A., Johnston, P., & Baker, T. (2008). *Domains and Ranges for DCMI Properties*. Retrieved January 20, 2009, from http://dublincore.org/documents/domain-range/

Powell, A., Nilsson, M., Naeve, A., Johnston, P., & Baker, T. (2007). *DCMI Abstract Model. DCMI Recommendation*. Retrieved December 3, 2008, from http://dublincore.org/ documents/abstract-model

Dimitrios Koutsomitropoulos is an adjunct assistant professor at the Department of Computer Engineering and Informatics and a researcher at the High Performance Information Systems Laboratory (HPCLab), University of Patras. He has received a PhD, an MSc and a Computer and Informatics Engineer diploma from the same department. His research interests include knowledge discovery, automated reasoning, ontological engineering, metadata integration, semantic interoperability and the semantic web.

Georgia Solomou is currently pursuing a doctoral degree in Semantic Web technologies at the University of Patras in Greece. She holds a master's degree in computer science from the same University. She is a member of the High Performance Information Systems Laboratory (HPCLab) where she has taken part in many European research projects. Georgia's research interests are related to the Semantic Web standards and technologies, mainly focusing in the field of Digital Libraries.

Andreas Alexopoulos has received his BSc in Computer Science from the Computer Science Department of University of Crete in Greece. He is currently a postgraduate student in the Department of Computer Engineering & Informatics of the University of Patras. His master thesis and his research interests are about educational metadata and interoperability in digital repositories.

Theodore Papatheodorou is a professor at the Department of Computer Engineering and Informatics, University of Patras, since 1984. Since 2005 he is the Department's Chairman. He is the head of the High Performance Information Systems Laboratory (HPCLab). He has received a PhD in Computer Science in 1973 and a MSc in Mathematics in 1971 from Purdue University as well as a BSc in Mathematics from University of Athens in 1968. He has authored hundreds of scientific publications in several areas of computer engineering and computer science.