Ontology-based learning objects search and courses generation

M. Arrigoni Neri, M. Colombetti

Dipartimento di Elettronica ed Informazione, Politecnico di Milano, piazza Leonardo Da Vinci, 20100 Milano, Italy

Learning objects paradigm is widely adopted in e-learning environments. Learning objects management can be improved using semantic technologies from ontology engineering and the semantic web. In this paper we use a semantic model of the repository to improve both learning objects retrieval and composition. The use of domain knowledge enables automatic reasoning and makes the system able to import new domain models and use them to interrogate the repository [1]. Learning objects composition is one of the main challenges in e-learning management systems and can be improved exploiting ontological reasoning. The building of a course can be carried out in two phases, in the first one we compose concept level entities to obtain an outline of the course, then we fill such an outline with actual resources from the repository. Both phases can use ontology based models to capture specific domain knowledge [2]. In order to give an intuitive and expressive ontology representation we briefly propose a graphical syntax for the well known ontology language OWL.

1. Introduction

Learning objects paradigm is the main response to the need for intelligent e-learning supports. A first definition of learning object comes from LTSC[20]:

"any entity, digital or non-digital, that can be used, re-used, or referenced during technology supported learning".

Such a definition is maybe too general and includes almost any physical or conceptual entity[27]. For this reason another definition was proposed by Wiley[28], that restricts the domain to digital libraries:

"learning object is a digital resource that can be reused to support learning", [28]

Learning objects management needs users and artificial agents to share problem semantics; in early CAI[2] systems the largest part of semantics was managed by the user, while agents simply used known keywords from metadata whose meaning was not formally defined. Recently, the development of ontologies as formal and explicit representations of

---

1Learning Technology Standards Committee

2Computer Aid Instruction
world conceptualisations[11] and their applications to the semantic web [6] defined languages for ontologies like RDF\(^3\) [26] and OWL\(^4\) [17] and let us go beyond this simple scenario, thanks to the description logics [3] funded semantics of ontological languages [24,15] and the presence of optimised reasoning algorithms [13,14]. First of all ontology engineering improves system modularization, since each software agent can be tailored to work with specific ontology. For example information retrieval agent can understand an application ontology of electronic resources, while a tutor agent can understand a pedagogical terminology. Ontology mappings can then allow the agents to talk each other and share the same repository model. Furthermore, before the development of formal logics for ontologies specific domain semantics had to be directly implemented in the code, unfortunately designer cannot know everything about the domain learning objects will talk about. As pointed out in [22], a learning objects management system must be open to external contributions and must be able to understand third part domain models.

Ontological systems can be used both to improve already available services and to provide completely new ones. In literature many applications of ontologies are known, such as closed questions generation [9] and ontology composition tests [19]. In our system the semantic model of the repository is used to provide two different services:

**Learning objects retrieval** can be improved by an ontology. The ontology language, namely OWL [17], comes with a formal semantics based on Description Logics [3], so the user can use terms from the ontology to express complex semantic queries relying on a sound and complete reasoning to answer them.

**Learning objects sequencing** involves the knowledge about domain model, learning objects and the student. This complex process can be carried out by a specialised artificial agent that understands the pedagogic meanings of metadata. Ontology mapping is fundamental to reinterpret the model in a pedagogic perspective.

Furthermore, ontologies are fundamental to:

1. improve interaction with the user, since ontology is shared by user and artificial agents

2. improve system modularization; each agent in the system is built using a task specific ontology, ontological mapping can then let agents talking each other and sharing the same repository model.

3. make the system open to external knowledge integration; since data semantics has been formalised in a declarative way it is possible to import ontological models from the web.

The paper is structured as follows: in section 2 we briefly describe the global architecture of the system and its ontological model, in section 3 we introduce a graphical syntax for the well known ontology language OWL, we defined it to represent ontological models in an intuitive and powerful way. Section 4 describes the overall ontological model and how

---

\(^3\)Resource Description Framework

\(^4\)Ontology Web Language
it can be interpreted in an ontological environment, in section 5 we deal with the resource retrieval problem and we show how the ontological stuff can be used to query the repository with different levels of complexity. In section 6 we show the pedagogic planning process and the coordination of different actors through ontology mapping. Section 7 concludes the paper with some closing remarks.

2. System overview

VICE is a learning object repository management system whose final users are teachers that want to find resources to compose lessons. At the lower level the semantic model consists of a complete description of each available resource through an RDF[26] graph.

We designed and implemented an ontology based architecture that lets the platform to semantically tag metadata using ontologies freely imported from the web. The obtained semantic model of the repository can be used to search for learning objects that match user needs. In our architecture we distinguish between the resources model and the contents model. Resources model is the translation in a semantic web language of a classic metadata schema. The ontology of this model is one of the outcome of VICE project and therefore it is static and makes our model able to dialogue with the lower levels of the platform. On the other hand, content model describes the particular knowledge domain the resources talk about. This higher level model contains abstract objects, that is the teachable contents the system can handle, and cannot be assumed to be known by the system designer since typically changes during system life; so the system allows the user to enrich domain model adding new ontologies from the web.

![Global VICE functional schema](image)

Figure 1. Global VICE functional schema

Figure 1 shows the whole VICE architecture. At the lower level MILOS handles row
data and metadata, that are archived in two different databases, encapsulating learning objects according to SCORM\textsuperscript{5} standard in order to eventually support web-based content delivery. MILOS API allows higher levels to access its services. Semantic module interfaces with MILOS API to maintain an RDF copy of the metadata repository. It provides an ontology importing agent to maintain domain models and provides the classic syntactic applications with a simple API to access semantic enriched tagging.

The upper level ontology of VICE expresses the relation between resources and their contents. Each resource in the repository will be classified inside the resource ontology, that is a learning object will be connected with one or more concepts inside resource ontology through instance-of links. The relationship between each resource and its pedagogical contents allows the system to go beyond classic human readable metadata. In fact, while in classic repositories the topic of a lesson or the topic of an exercise is simply described in natural language, upper ontology uses the role \textit{hasTopic} to link the resource to a generic \textit{TeachableContent}, represented by an individual inside a specific content model.

![Figure 2. Upper ontology: a LearningObject has some topics that are generic Teachable-Contents](image)

\textit{LearningObject} and \textit{TeachableContent} are the top concepts, respectively, of resources and the contents ontologies.

3. A graphical syntax for \textit{OWL}

The first step in ontology integration is the choice of a suitable ontological language. We choose \textit{OWL}\textsuperscript{6} from W3C consortium. In particular, we used \textit{OWL-DL}, that is the more expressive \textit{OWL} decidable sublanguage; it corresponds to the well known description logic \textit{SHOIN}(D+) \cite{15} and allows for complex concept composition in a classical Tarsky style extensive semantics using both boolean operators and cardinality restrictions over roles.

Ontology languages are fundamental to make artificial agents able to “understand” the meaning of row data but, by the same time, as the language complexity grows it can become unfeasible for a non technical human user to manipulate semantic descriptions. Nowadays there are graphical syntaxes for the semantic web lower level languages, \textit{XML} and \textit{RDF}. Ontology editors like Protége\textsuperscript{7} from Stanford University typically organise terms in a tree-like visualisation that represents the ontology backbone. However, inter-concept relations different from subsumption are usually rendered through an \textit{RDF} graph,

\[5\text{Sharable Content Object Reference Model: http://www.imsglobal.org/}\]
\[6\text{Ontology Web Language}\]
\[7\text{http://protege.stanford.edu/}\]
so specific OWL constructs are represented as categorial symbols or, in the best case, a general purpose representation like UML is used. In order to make user interaction easier we defined a simple and intuitive graphical syntax for OWL – DL, namely GrOWL\(^8\), in which each OWL constructs has its own stereotype.

Figure 3. Principal elements of GrOWL syntax

First of all each concept, that is each class, is represented by an ellipse, just like in graph RDF representation. As shown in the upper-left part of figure 3, special diamond stereotypes are used to represent top and bottom concepts and class definitions can be related to each other through logical operators. OWL assertions about classes are represented by links, as shown in the upper left part of figure 3, like subsumption (\(\sqsubseteq\)), equivalence and disjointness. The upper right part shows syntax for properties (or roles in description logics terminology). A property is represented by a bidirectional arrow with a double arrow on both sides. These arrows are to be taught as linking an object on its left to an object on its right, the presence of a single arrow on one side means that the property is functional in that direction. Finally figure 3 shows the stereotypes for transitive and symmetric properties. To indicate the domain and the range of a property it can be connected with incoming arrows from domain classes and out coming arrows to range classes (see for example figure 4). The central part of figure 3 shows two of the many concept constructors: concept conjunction and existential quantification. The first describes the set of objects that belong both to \(C6\) and \(C7\) concepts (\(C6 \cap C7\) in DL syntax), while

\(^8\)Graphic OWL
the second describes the set of individual with at least one \( r - \text{successor} \) that is instance of class \( C \) (\( \exists r.C \) in DL syntax).

An individual can be connected to one or more classes by an instance - of (\( \in \)) link, user can declare two object to be aliases or explicitly different and can link two individuals with a role labelled link, that is the instance of a property.

4. Learning objects semantic metadata

In this section we discuss in more details the ontological model, both describing resource ontology and addressing the problem of third part content ontology integration. Finally we use GrOWL syntax from previous section to semantically query the joint ontological model.

4.1. Resource taxonomies

First of all learning objects are electronically reachable resources [28], but due to their content, they are particular resources with a didactic use. The relationship between a resource and its content has been addressed by the upper ontology, however we have to classify resources themselves. VICE resource ontology is a multitaxonomy, that is the collection of three taxonomies used to classify learning objects with respect to different and orthogonal dimensions.

To a first approximation we can consider a learning object just as a generic electronic resource. The first taxonomy, namely electronic resource taxonomy considers the file type as the main characteristic of the resource, so a learning object can be classified as a video, a slide presentation or a text. This information allows the teacher to take into account specific student abilities and course peculiarities. For example if the teacher wants to make a class, he will prefer slide presentations and videos, while if he is building up the course notes he will look for texts. File classification is the basis for cognitive filtering. A particular care has been taken in disability aware filtering: if the student has a vision disability, the teacher can avoid videos and the system will look for audio contributions.

On the other side, we can consider a learning object a pedagogical artifact, regardless it is digital or not. Along this dimension the classification is centred on didactic and pedagogical use of the resource. So we can distinguish between lessons and exercises, examples and introductions. This Pedagogic taxonomy is fundamental in learning objects selection, but is even more important if we want an artificial agent to automatically compose courses. In fact pedagogic dimension allows the agent to articulate a class using a classic path like introduction-explanation-example-exercise.

The first two taxonomies describe the electronic and the pedagogical aspects of resources. As shown by Wiley [29], the combination of the pedagogical value and the digital nature of the resource generates new issues, that are not characteristic of classic didactic material nor generic digital resources. Digital artifacts can interact with the user and can modify the way the content is presented or even the content itself. Wiley taxonomy identifies five types of learning objects:

- Single-type: resource that can be used atomically.
- Combined-intact: a learning object composed by two or more components (for example a photo and a text)
Ontology-based learning objects search and courses generation

- Combined-modifiable: a combined resource that can dynamically choose and combine its components

- Generative-presentation: a resource able to dynamically generating the content to be presented to the user. For example an application that can generate a different example at each request

- Generative-instructional: a complete learning environment

Resource ontology translates VICE metadata schema in an OWL T-Box using LOM⁹[18] from IEEE as the staring point. Resource ontology metadata are of three different types:

**taxonomy metadata** allows semantic module to classify learning objects in resource multitaxonomy. Metadata editor has no access to semantic descriptions and is not able to manage type links. However object classification can be related to special metadata from the resource taxonomy itself. For example in the taxonomy we can say that CompositionalLO are exactly the learning object whose loType metadata assumes a specific value (see for example figure 11).

**generic LOM metadata** are metadata from VICE platform with no special meanings for semantic module. They can be used by the user to further restrict query results

**hasTopic metadata** is the most expressive one. It not only assumes semantically rich values, but possible values depends on the specific content domain. In order to make the user to enter only correct values without make him to browse the whole content ontology a keyword-based interface with metadata editor was developed. The user can search by keyword for the intended topic. A textual description is returned for each compatible individual to support user choice.

### 4.2. Importing content ontologies

While resources are classified through a static ontology, content models have to be dynamic. The system can be tailored to work with learning objects about any knowledge domain, so the user must be able to define its own domain model and use such a model to tag resources. We allow the user to indicate a new domain ontology through an URI; the system recovers the new model as an OWL file. So, a knowledge domain model can be assumed to be a generic OWL ontology.

Unfortunately, such a flexible solution comes with a drawback. An ontology designed to represent the knowledge about a specific domain is usually not directly suitable for resource metadata. In general, an ontology engineer designs the ontology as a collection of concepts, more or less organised in taxonomies. Then he uses language constructs to further constraint ontology interpretations and finally he fills its ontology with some relevant individuals. The result is that some domain terms are used as concepts, while others are used as individuals. The level of this division largely depends on designer preference and the specific goal the ontology is designed for. When the ontological model is imported in VICE and is used as the range of hasTopic role we must be able to link a specific resource to both individuals and concepts names. In fact, a learning object can talk about

---

⁹Learning Object Metadata
any domain concept regardless its abstraction level and without any commitment about subconcepts or individuals. We will say the topic relation does not distribute on concept instances nor through subsumption chains.

For example, let the imported ontology talk about mathematics. We could suppose \textit{Function} to be a T-Box concept. \textit{GoniometricFunction} could be a \textit{Function} subconcept with an instance \textit{sin}.

![Image](image.png)

Figure 4. A tiny fragment of a repository that refers to a mathematics content ontology

Figure 4 shows an example of a a textual learning object that is an introduction and is Simple (as classified in Wiley taxonomy). If the introduction is about the specific function \textit{sin} no problem rises, but what happens if the learning object is an introduction of the concept of function? In this case the resource cannot be linked to any individual in the ontology. The easiest solution could be to link the resource directly to an intermediate node of function taxonomy; however this should result in a second order model, since we have a link between an individual and a class and make unfeasible any automatic reasoning.

In order to obtain a first order model third part content ontology must be processed. We must guarantee that for each term in the ontology an appropriate individual is available to be linked to resource descriptors. In order to accomplish this we use a \textit{reification} that adds to the ontology some new individuals as facets for classes. Reification module works with a very simple application ontology that contains only the concept for new generated objects, namely \textit{Representant}. The new imported ontology is processed; for each named concept \textit{C} a facet individual \textit{\tilde{c}} is added such that \textit{Representant(\tilde{c})}. It is then necessary to build new concepts to classify reified objects. For example (see figure 4) lets the user querying for learning objects about \textit{Funtions}; if the concept of \textit{Function} is considered in the original ontology meanings only resources about specific instances of \textit{Function} concept will be found. If \textit{Function} is interpreted in an extended meanings, that is taking in account the reified objects, general introductions to the \textit{Funcion} concept will be returned, as well
as introductions to particular subconcepts like *GonimetricFunction*. So, starting from $C$ a new extended concept, namely $C^+$ is built adding $\tilde{c}$:

$$C \cup \{\tilde{c}\} \subseteq C^+$$ (1)

Finally extended concepts must be organised in a taxonomy following original concepts one.

Ontology importing, therefore, consists in four phases:

1. connect imported ontology to VICE upper ontology making each concept a subconcept of *TeachableContent*
2. reify each concept $C$ in an individual $\tilde{c}$
3. build extended concepts $C \cup \{\tilde{c}\} \subseteq C^+$
4. organise extended concepts in a taxonomy

![Ontology Diagram](image)

**Figure 5.** An example of reification

After these steps each term has a corresponding individual and the user can take advantage of reification technique in ontological queries.

Unfortunately, import procedure makes the ontology to grow to nearly the double of its original size. Even without this expansion handling the whole repository description can be unfeasible for realistic databases. Since reasoning with expressive description logics is at least exponential $[14,13,16]$ ontological reasoning can be performed only on small databases. For this reason VICE semantic module clusters resources in islands. Each island contains learning objects that are like to be accessed in the same context. So an island is simply the set of all the learning objects that refer to the same content ontology. When the user starts using semantic module he is asked to select a domain ontology, so the system will load only the fragment of repository about the chosen domain.
5. Learning Objects retrieval

Once the ontological model has been built, the user can describe the learning object using all the available ontologies. An instance retrieval service from an antology reasoner will then retrieve all the resources from the repository that match user needs. In our experiments we used RACER [12] as description logic reasoner, our platform is based on Jena \(^{10}\) and the DIG[5] interface has been used to access reasoner services.

More the expressive power you give to the user, more complex is query composition and more skills are required to use the system. So in VICE we developed three different query interfaces, with different complexities.

5.1. Query interfaces

The first and simplest query interface gives to the user the three resources taxonomies and the content ontology through a graph.

![Ontology navigation based query](http://jena.sourceforge.net/)

The user selects concepts from resources taxonomies and navigates content ontology following subsumption links and properties. Reification is hidden by the application, the user analyses content domain exploring the original imported ontology. Concrete reified individuals and extended concepts are possible qualifications of desired content. Once the

---

\(^{10}\)http://jena.sourceforge.net/
user has selected the node he can indicate if he wants:

- only the instances from original ontology, that is the query for $C$ concept instances
- only the concept itself, that is the individual $\tilde{c}$
- both the previous, plus any reification of $C$ subconcepts, that is extended concept $C^+$

In this first query the user simply select the right nodes inside an existing ontology. An ontological model allows for much more flexible uses. In fact, the user can use existent terminology to define its own concepts and describe a rich query. The second interface allows the user to manipulate content ontology, placing concepts and individuals on a blank table and building new concepts through GrOWL syntax. In this case each named concept from content ontology must be qualified respect to reification process\footnote{selecting $N$ for original concept, $P$ for the reification of the concept and $R$ for the whole extended concept}. The tool automatically compose the wanted concept and execute the instance retrieval on:

$$C_w \sqcap C_p \sqcap C_r \sqcap \exists hasTopic.C_c$$

(2)

where $C_c$ is the user defined concept and $C_w$, $C_p$ and $C_r$ are, respectively, the nodes selected in Wiley, pedagogical and resource taxonomies. Finally, if the user is skilled in ontology use he can freely combine concepts from the whole ontological model. In this last interface the user describes the learning object he is looking for combining both resources and content ontologies and uses top ontology role $hasTopic$ to bind the two parts of the description.

Free composition of all the terminologies brings to a strictly more expressive query language. in fact the user can express logical constraints on resource part of the ontology, looking for example for any multimedia resource except video files and exercises. Furthermore the explicit use of $hasTopic$ role allows the user to express more than one possible content. For example he could look for any video exercise about any GoniometricFunction and about substitution solving method; such a query cannot be expressed using the second interface, in fact the conjunction of GoniometricFunction and Substitution on the content side produces the query for a learning object that talks about a domain content that is, at the same time, both a function and a solving method. To ensure the system to return only learning objects the system will perform the instance retrieval task on the concept

$$LearningObject \sqcap C_u$$

(3)

where $C_u$ is the concept described by the user.
5.2. Query answering with closure

When the query is described using simple or full graphic composition a possible semantic problem raise. In fact, in these cases user is allowed to describe a rich query using virtually any valid OWL construct, including disjunction and free negation, so the result of the query changes if the system assumes the model to be complete or not. This is one of the main differences between ontological systems based on Description Logics semantics[15] and classic database systems. In Description Logics you usually have an open world assumption (OWA), that is you assume tour model to be incomplete and you admit that other assertions on the same resources can be done by other people in other parts of the world and your reasoning must be correct according to this vision. In a database system, on the other side, you assume your data to be complete, that is everything you can not read in your data is assumed to be false, so you work under a closed world assumption (CWA).

Description logics semantics is founded on a classic Tarsky semantics over an interpretation domain $\Delta$. In general an interpretation $I$ is a couple $I = \langle \Delta^I, \cdot^I \rangle$ where $\Delta^I$ is the interpretation domain and $\cdot^I$ is an interpretation function assigning a set of individuals to each atomic concept and a set of pairs of individuals to each role name. OWL-DL language is the DL based fragment of OWL; it is a syntactic variant of SHOIN description logic, whose concept constructors are listed in table 1. Am OWL-DL ontology can be reduced to a finite set of axioms in the form $C \sqsubseteq D$ and $R \sqsubseteq S$ where $C$ and $D$ are concepts and $R$ and $S$ are roles, forcing concept (role) to be interpreted as a subset of the other. An interpretation $I$ is a model of an ontology if it satisfies all the axioms in the ontology.

OWL-DL language is expressive enough to show the difference between OWA and CWA queries, since it allows concepts free negation and universal quantification over roles.
Obviously, when a user asks for objects in a repository he assumes a CWA, so he can obtain an unexpected behaviour if the retrieval technique is based on DL reasoning, just like in our case. In order to regain a closed world semantics we can procede in two different ways:

- by implementing a new DL reasoner that works under CWA, this is a difficult way since the most of tableaux based algorithm actually used to reason on DLs [4,16,13,14] cannot be practically modified to take into account CWA. Furthermore, such a solution should make the system incompatible with the rest of semantic web solutions

- by forcing the closure at the ontology level, reformulating both the query and the ontology. This is a more practical solution since it does not require the implementation of new ontology reasoning stuff.

In general, if we call $M(O)$ the set of models of $O$ we have that the evaluation of a query $Q$ on the ontology $O$ returns the set of individuals that are interpreted inside $Q$ in any model of $I$:

$$\text{eval}(Q,O) = \{ x : \forall I \in M(O), x \in Q^I \} \quad (4)$$

Following the second approach our goal is to evaluate a query $Q$ on an ontology $O$ that contains all the individuals from the repository obtaining a set of resulting individuals, namely $\text{evalc}(Q,O)$, such that $x \in \text{eval}(Q,O)$ iff $x \in Q^I$ in any $I$ that is a model of $O$ under a CWA. If we apply CWA we have to change the instance retrieval on concepts in the forms $\forall R.C$, $\neg C$ and $\leq nR$. 
Table 1
OWL-DL basic constructors syntax and semantics, where C and D are concepts, R is a role and n is a natural number

<table>
<thead>
<tr>
<th>OWL constructor</th>
<th>DL syntax</th>
<th>( CL )</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>( C \cap D )</td>
<td>( CL \cap DL )</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>( \exists R.C )</td>
<td>{ ( x \in \Delta^I : \exists y : x,y \in R^I ) }</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>( \forall R.C )</td>
<td>{ ( x \in \Delta^I : \forall y : x,y \in R^I ) }</td>
</tr>
<tr>
<td>unionOf</td>
<td>( C \cup D )</td>
<td>( CL \cup DL )</td>
</tr>
<tr>
<td>complementOf</td>
<td>( \neg C )</td>
<td>( \Delta^I \setminus CL )</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>( \leq nR )</td>
<td>{ ( x \in \Delta^I : # { y : x,y \in R^I } \leq n } }</td>
</tr>
<tr>
<td>minCardinality</td>
<td>( \geq nR )</td>
<td>{ ( x \in \Delta^I : # { y : x,y \in R^I } \geq n } }</td>
</tr>
<tr>
<td>cardinality</td>
<td>( = nR )</td>
<td>{ ( x \in \Delta^I : # { y : x,y \in R^I } = n } }</td>
</tr>
<tr>
<td>oneOf</td>
<td>{i}</td>
<td>{i}</td>
</tr>
</tbody>
</table>

In our approach, starting from the query \( Q \) and the reference ontology \( O \) we want to build a new query \( Q' \) and a new ontology \( O' \) such that:

\[
eval(Q',O') = evalC(Q,O) \tag{5}
\]

A possible approach is to extend the semantics of the corresponding DL (SHOIN in our case) with an epistemic modal operator \( K \). According to Fitting [10] epistemic extensions could have not a unique interpretation and the choice of a first order semantics depends on application. An in depth investigation of the properties of a modal description logic is beyond the focus of this paper, however we notice that since we use knowledge operator \( K \) to force a closure assumption in query evaluation we can assume rigid designators, that is there is an infinite set of designators and this is the interpretation domain for every worlds. This allows us to interpret an epistemic concept \( KC \) as the same set in every world extending to the modal setting the idea of interpreting concepts as sets [7]. Instead of adding an explicit knowledge modal operator to OWL it is possible to define a procedural semantics for K, adding to the ontology new axioms that fixes the interpretation of K-closed concepts (that is concepts in the form \( KC \) ) and K-closed roles (in the form \( KR \)) according to their minimal interpretations, so:

\[
KC^I = eval(C,O) \tag{6}
\]

\[
KR^I = \{ < x, y > : x \in eval(\exists R.\{y\},O) \} \tag{7}
\]

Given a query \( Q \), lets call \( C(Q) \) and \( R(Q) \) respectively the set of concept names and roles in \( Q \). In order to evaluate \( Q \) under CWA it is necessary to close the interpretation of all the roles and concepts in \( Q \), this operation, namely closure of concepts and roles consists in adding new K-closed versions of the concepts and roles:

- set \( O' = O \)
for every $C \in C(Q)$
set $O' = O' \cup \{KC = eval(C, Q)\}$

for every $R \in R(Q)$
let $Dom(R) = eval(\exists R. \top, O)$
set $O' = O' \cup \{Dom(R) = \exists KR. \top\}$
for every $x \in Dom(R)$ let $Imm(R, x) = eval(\exists R^{-}.\{x\}, O)$
set $O' = O' \cup \{\exists KR^{-}.\{x\} = Imm(R, x)\}$

The second step is to modify the query to take into account the closed versions of both concepts and roles. Since OWL-DL is propositional closed it is possible to express any concept in negative normal form (NNF) in linear time w.r.t. the size of the concept. A concept is in NNF if negation appears only in front of concept names (or enumerated concepts); we can push negation inside the formula in an inductive way w.r.t. the number of concept constructors using De Morgan laws and quantification rewriting rules:

<table>
<thead>
<tr>
<th>negated subformula</th>
<th>equivalent NNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg\neg C$</td>
<td>$C$</td>
</tr>
<tr>
<td>$\neg(C \sqcap D)$</td>
<td>$\neg C \sqcup \neg D$</td>
</tr>
<tr>
<td>$\neg(C \sqcup D)$</td>
<td>$\neg C \sqcap \neg D$</td>
</tr>
<tr>
<td>$\neg\forall R.C$</td>
<td>$\exists R.\neg C$</td>
</tr>
<tr>
<td>$\neg\exists R.C$</td>
<td>$\forall R.\neg C$</td>
</tr>
<tr>
<td>$\neg \leq nR$</td>
<td>$\geq (n + 1)R$</td>
</tr>
<tr>
<td>$\neg \geq nR$</td>
<td>$\leq (n - 1)R$ if $n &gt; 1$, $\forall R.\bot$ otherwise</td>
</tr>
</tbody>
</table>

$Q'$ can be obtained from the NNF of $Q$ by substituting $C$ with $KC$ in any occurrence of subformula $\neg C$ and by substituting $R$ with $KR$ in any occurrence of subformulas $\forall R.C$, $\leq nR$ and $= nR$.

6. Learning objects composition

Course sequencing is the process that composes existent resources to build a more complex pedagogic entity, a course, that responds to student and teacher needs. In most cases system is provided with an overlay model of the student, that is the list of all the contents the student is assumed to know. The system must know the pedagogic goals too, that is the final student model. This is another overlay model indicating all the contents the user is expected to know after the course.

Pedagogic planning requires the cooperation of different actors, each one with a specific competence:
domain expert is responsible of the correct formalisation of the domain. He must have a complete and sound knowledge of the topics and of all the domain specific relations between them. In principle, he could have no pedagogic skills at all.

domain pedagogist is a teacher specialised in the domain. It provides a pedagogic view on the domain structure. Can be less skilled in the domain than the domain expert, but has to integrate the global schema of the domain with a standard pedagogic view.

pedagogic-style expert knows the tactics to be implemented during teaching. In principle can have no specific skills in the domain, since his contribution is largely domain-independent. He provides the knowledge about cognitive aspects of the learning process, at both the strategic and the tactical level. At the strategic level he defines some policies on how to use the pedagogic view of the domain to build a suitable curriculum, while at the tactical one he gives the rules to compose actual teaching units with actual learning objects.

Each actor has its own perspective on the teaching process and describes it in a task specific language. Exploiting knowledge management techniques it is possible to integrate the different contributions and coordinate the work of the actors to achieve the final goal. The main integration problem is in the schema level, that is different experts describes the same domain using different terminologies, so formal ontologies can work as a bridge between different descriptions and as the glue of the whole system.

Course sequencing can be divided in two phases: content planning and delivery planning. Ontologies can be used in planning in three ways: translating an ontological model to another language, giving an operational semantics to a specific ontology through an ad-hoc planner or reducing plan composition to an ontological reasoning task. In our work we use the first approach for content planning and the second in delivery planning.

content planning First of all it is necessary to build a content plan, that is the outline of the course, reporting a linear or non linear plan of contents in the order they will be presented to the student. This phase uses initial and final student models to extract a plan from the domain ontology. Since it involves content structure, the skills of all the actors must be coordinated to achieve a content-level planning: the modelling skills of the domain expert, the specific competences of domain pedagogist and the pedagogic-style expert with specific focus on strategic (that is content level) teaching approach.

delivery planning In the second phase content plan is filled with actual resources from the repository. The system must use metadata from repository and resources ontology to organise learning objects according to teacher pedagogical preferences. Delivery planning is tightly related to the pedagogic style of the teacher, so the main actor involved is the pedagogic-style expert, with his specific competence on pedagogic tactics.

Figure 9 shows a simple scenario. On the left we have the graph of content knowledge, initial and final overlay models for the student are represented as empty bordered nodes.
Ontology-based learning objects search and courses generation

6.1. Content planning

The first planning phase, given the set of available domain concepts $DOM$, an initial overlay model $I \subseteq DOM$ and a target overlay model $T \subseteq DOM$ produces an outline as an ordered sequence $O = \langle ..C_i.. \rangle$ of selected nodes such that $T \subseteq O$ and for each $C_i \in O$, every $\tilde{C}$ that is a prerequisite for $C_i$ is either $\tilde{C} = C_j \in O, j < i$ or $\tilde{C} \in I$.

Planning on content graphs, and in particular on conceptual graphs, rises two problems:

- planning graph is not known a priori. Not only its geometry is unpredictable, but even the terminologies used to label its links cannot be forced at design time. In fact, content ontology can vary from a repository to another and can even be imported from the web in a semi-automatic way [1]

- pedagogic approach, that is the strategy on which link to follow during planning, is a fundamental teacher contribution to the planner and should be provided in an effective and simple way.

Such issues can be assessed exploiting terminological reasoning. Content planning must use a graph representation of content domain in which contents are connected by pedagogic

---

http://zeus.ing.unibs.it/lpg/
relations. The first step is then the definition of a pedagogic ontology. Specific content ontologies can differ by the number and types of concepts and roles, but each domain must be mapped to pedagogic planner model in order to be used in automatic course generation.

When the system administrator wants to enable automatic planning in a specific model, that is on a specific content ontology, he maps domain model to pedagogic ontology; this mapping is carried out in two phases. First some roles from the content ontology are mapped to pedagogic ones. For example in a mathematics repository the administrator could indicate usesLemma to be a special case of hasStrongPrerequisite to indicate that the planner must check that each lemma are presented to the student before the theorem. In a second phase system administrator can browse content ontology and manually add specific pedagogic links between content individuals.

We stress that, once ontological mapping has been carried out, the planner has no need to access domain ontology, since any possible assertion has been translated in pedagogic links.

The planning problem is itself described through an ad-hoc ontology of problems and plans figure 12. Problem description contains, among others initial and desired student models and the description of the pedagogic approach to be applied. In the upper part of figure 12 is reported the ontology to describe the problem, while in the lower part is shown the terminology used to represent the composed plans.

Starting from pedagogic ontology, PDDL translation is performed in two steps. In the first phase the content composition is mapped to a planning problem described in the problem ontology. In the second phase such a model is finally translated to PDDL. This intermediate translation allows a division from OWL and PDDL domain, in the
Figure 11. Repository queries are expressed using content and resource ontologies. Content planning uses content ontology individuals with the mapping to pedagogic ontology. Finally learning object selection uses resource ontology to select suitable objects.

First translation the translation takes into account pedagogic specific problems, while the second phase is related to strictly syntactic mappings and handles all the language translating problems.

Pedagogic ontology basically contains two kinds of roles:

**admissibility roles** are roles that express constraints on the final plan. Typical example of admissibility roles is the *strong prerequisite*. No concept can be presented if a prerequisite cannot be assumed known by the student.

**step roles** are generic pedagogical roles that label the links between two contents. The planner follows these links to build a path from initial knowledge to final student status. Example of step roles is *analogy*, that can be usually followed to introduce new concepts.

The choice of the next step depends both on the goal knowledge and the type of the links to follow. User can give a weight to each pedagogic relation, so a low weight to the *analogy* role will make the planner to prefer such links among others, while an high weight of *falseAnalogy* role will avoid to present consecutively two interfering contents. So, a pedagogic approach, or *pedagogic strategy* is simply the weight assignment to each step role. Given the domain model expressed in pedagogic ontology and the model of the problem, VICE planner translates the problem in *PDDL* \(^{13}\)[8], passes the obtained *PDDL* problem to an external planner and translates back the resulting plans in *OWL* filling the remaining part of the problem semantic model \(^{14}\). The resulting plans are ordered in global

---

13Planning Domain Definition Language
14using problems and plan ontology
Figure 12. Problem and plan intermediate ontology, the planning problem description contains the set of possible action types, while the resulting plan is composed of actual tokens, each one has an action from the problem as its type.

fitness descending order using the plan ontology *isBetterThan* role. *PDDL* description of the problem is built assembling the following fragments:

**student model** In the translation the overlay status of knowledge of the student is represented by a *PDDL* predicate "knows" that expresses the fact that, in a given planning world, the student knows a given topic. Planning actions modifies such representation adding new acquired knowledge every time a new topic is scheduled by the planner. Initial partial worlds description is obtained from the initial overlay model, while goal world is obtained enumerating each intended outcome.

**pedagogic structure of the domain** each pedagogic link that can be inferred by the ontology mapping must be translated in *PDDL*. The simplest solution is to foresee a binary predicate for each role in pedagogic ontology from figure 10.
**actions** in a pure pedagogical vision, we should create an action for each content in the domain model. However, plans must be valid with respect to strong prerequisite structure. So, our translator produces five basic PDDL actions, parametric with respect to actual content, each one considering different relations between the next topic and the current structure of the plan.

**cost function** all admissible plans must guarantee to teach all target contents following only existing links and respecting all strong prerequisite constraints. Furthermore selected plans should be optimal with respect to a global cost. Cost function simply calculates the cost of a plans as the sum of links weight as indicated by pedagogic strategy.

The whole pipeline from domain description to PDDL plan problem description is shown in figure 13 and allows all the different actors to work together. In fact, the *domain expert* provides the initial domain fragment, *domain pedagogist* maps the fragment to the pedagogic ontology filling the gap from the domain and its pedagogic use, while the *pedagogic expert* gives the pedagogic strategy.

Ontologies for planning are not a new research topic. Specific ontologies for planning problems have been proposed for *DAML*\(^\text{15}\)*, the *OWL* predecessor, and a *DAML* to

\(^{15}\)Darpa Agent Markup Language
PDDL translator has been developed\textsuperscript{16}. However, our work is different. Our system generates a new planning problem from the pedagogic task description. We are anyway evaluating to generate an intermediate OWL model for the generated plan and delegate to DAML/PDDL converter the final translation.

6.2. Delivery planning

Delivery planner fills the course outline from the previous phase with actual learning objects from the repository, obtaining a sorted set of learning objects ready to be delivered to the students. In this phase, tactical pedagogical choices must be done. The planner receives the course outline, that is the ordered set of concepts plus the costs of each step from planning graph; such costs will be considered proportional to the foreseen difficulty of the step. Once the set of topics is decided as well as the order in which they must be presented, any teacher can express some rules to build ”good” courses. A simple and powerful way to express pedagogic tactics is through resources composition templates. Each template can apply to one or more consecutive concepts in content plan and describes pattern of pedagogic types. For example a classic template could be the triple \texttt{< presentation, exercise, test >}. Each typed placeholder is a slot, so the example plan is composed of three slots.

Figure 14. Template ontology

Figure 14 shows the ontology used by the pedagogic-style exper to describe templates, it provides two main concepts:

Template is the set of templates available during planning. Each template is applicable to a fragment of course outline, that is can cover one or more concepts, given that the global difficulty of the fragment is inside a given difficulty interval. Template difficulty interval expresses the teacher knowledge about the cognitive complexity

\textsuperscript{16}http://www.cs.yale.edu/homes/dvm/daml/pddl2owl/translator1.html
the template can manage. Admissible interval is expressed by \textit{minComplex} and \textit{maxComplex} roles. Once a template has been allocated to one or more contents it forces some temporal constraints on learning objects; the total time amount of filling resources must be inside another interval expressed by \textit{minTime} and \textit{maxTime}.

\textbf{Slot} is the concept collecting specific slots that compose templates. Each slot covers a fraction of the time allocated for the template (role \textit{temporalRatio}). Actually, a slot must provide information about the admissible pedagogic types of the learning objects, this can be expressed refining \textit{Slot} concept in a set of type bounded slots (see \textit{ExerciseSlot} in figure 14).

Delivery planner composes the plan in the same order it will be executed using an \textit{A* - like} search algorithm. At each step the planner chooses a template for the next concepts or continue with a partially filled template.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{slots_subslots_decomposition.png}
\caption{Slots / subslots decomposition}
\end{figure}

When a template is assigned to a set of concepts a slot expansion is necessary to take into account multiple contents module and to manage time allocation through template inner structure. Figure 15 shows a template \textit{Temp} with two slots \textit{S}_1 and \textit{S}_2 applied the couple of contents \textit{C}_1 - \textit{C}_2. Temporal constraints are allocated to each slot using corresponding \textit{temporalRatio}, that is a slot \textit{S}_i has an admissible interval \([t_i = t \cdot tr(S_i); T_i = T \cdot tr(S_i)]\). Each slot is homogeneous w.r.t. resources type, but must be filled with resources about each content the template is applied to. Inter contents time allocation can exploit relative difficulties of each step in content plan, allocating more time more difficult can be the content. So, for each slot \textit{S}_i and for each content \textit{C}_j a \textit{subslot} \textit{S}_{ij} is created to represent the resources inside \textit{S}_i that talk about \textit{C}_j. Subslots time constraints are calculated as follows:

\[ S_{ij} : [t_{ij} = t_i \cdot \frac{w_j}{\sum_i w_i}; T_{ij} = T_i \cdot \frac{w_j}{\sum_i w_i}] \]  

\texttt{(8)}
As soon as $S_{11}$ is completed, its actual allocated time $\tilde{t}_{11}$ can further constraint the remaining subslots reverting tree coefficients (with a tolerance $\delta$ greater than zero to preserve satisfiability).

$$t_{ij} \rightarrow \max(t_{ij}, \tilde{t}_{11} \cdot \frac{w_j \cdot tr(S_i)}{w_1 \cdot tr(S_1)} \cdot (1 - \delta))$$ (9)

$$T_{ij} \rightarrow \min(T_{ij}, \tilde{t}_{11} \cdot \frac{w_j \cdot tr(S_i)}{w_1 \cdot tr(S_1)} \cdot (1 + \delta))$$ (10)

Plan optimality must be evaluated w.r.t. a cost function in the form $f(x) = g(x) + h(x)$ where $g(x)$ is the current partial plan cost and $h(x)$ is an optimistic predictor for the rest of the plan. For simplicity seek we use learning time as a cost function, that is the planner will try to reach pedagogic goals as quick as possible. Under this assumption $g(x)$ is the learning time of already selected learning objects, while $h(x)$ is the minimum time foreseen for the rest of the course. $h(x)$ assumes each unfilled subslot $S_{ij}$ to be completed in $t_{ij}$ time units and all unmanaged content to be completed in the minimum time (calculated in a preprocessing phase combining each plan fragment with any possible template). Since $f(x)$ is an optimistic approximation of actual time cost and is monotonic w.r.t. tree exploration it can be proven to be admissible and the planning process obtains an optimal solution [21].

Another issue is about learning objects selection. Given that delivery planner is working on the subslot $S_{ij}$ of pedagogic type $P$ and about a content $c$ resource retrieval can rely on ontological reasoning executing an instance retrieval reasoning task on a query concept $Q$ in the form $Q \equiv P \sqcap \exists hasTopic.\{c\}$

7. Conclusions

We presented our contribution to VICE project. Ontology engineering allows a modular design of the resource repository. A very simple application upper ontology allows the interoperation of the different components. Repository ontology is composed by a resource multitaxonomy for learning object classification and a set of content ontologies that can be imported from the web. The joint ontological model is maintained aligned with lower level data descriptions and is used with different semantic query interfaces.

Another semantic based service is learning object automatic sequencing. We proposed a two level composition that works first at the content level, producing an outline of the course, and then at the resource level, filling the outline with actual resources. A massive use of semantic descriptions allows to map specific domain concepts to a fixed pedagogic ontology the planner can understand. All the actors involved in pedagogic planning have their soles in the system during content planning, while the resource planning is guided by the pedagogic-style expert only.

REFERENCES

Ontology-based learning objects search and courses generation


