Journal of Computer Science 2 (8): 619-626, 2006
ISSN 1549-3636


# A Web-Based E-Learning System Using Semantic Web Framework

[1]Fayed F. M. Ghaleb, [1]Sameh S. Daoud, [2]Ahmad M. Hasna, [2]Jihad M. Jaam and
[2]Hosam F. El-Sofany
[1]Department of Mathematics, Faculty of Science, Ain Shams University
[2]Department of Engineering and Computer Science, Faculty of Engineering, Qatar University

**Abstract:** E-learning is being increasing viewed as an important activity in the field of distance and continuing education. Web-based courses offer obvious advantages for learners by making access to educational resource very fast, just-in-time and relevant, at any time or place. In this study, based on our previous work, we present a framework for our web-based e-learning system using the Semantic Web technology. In addition we present an approach for implementing a Semantic Web-based e-learning system, which focus on the RDF data model, OWL ontology language and RAP for parsing RDF documents. Also the use of RAP – a Semantic Web toolkit for developing our application is discussed in more details.

**Key words:** RAP, RDF, e-learning, semantic web, ontology

## INTRODUCTION

Increasingly, the WWW is used to support and facilitate the delivery of teaching and learning materials. This use has progressed from the augmentation of conventional courses through web-based training and distance learning to the web-based and e-learning education. E-learning is not just concerned with providing easy access to learning resources, anytime, anywhere, via a repository of learning resources, but is also concerned with supporting such features as the personal definition of learning goals and the synchronous and asynchronous communication and collaboration, between learners and between learners and instructors[1,2].

One of the hottest topics in recent years in the AI community, as well as in the Internet community, is the Semantic Web. It is about making the Web more understandable by machines. It is also about building an appropriate infrastructure for intelligent agents to run around the Web performing complex actions for their users[3]. Furthermore, Semantic Web is about explicitly declaring the knowledge embedded in many web-based applications, integrating information in an intelligent way, providing semantic-based access to the Internet and extracting information from texts[4]. Ultimately, Semantic Web is about how to implement reliable, large-scale interoperation of Web services, to make such services computer interpretable – to create a Web of machine-understandable and interoperable services that intelligent agents can discover, execute and compose automatically[5].

The problem is that the Web is huge, but not smart enough to easily integrate all of those numerous pieces of information from the Web that a user really needs. Such integration at a high, user-oriented level is desirable in nearly all uses of the Web. Unfortunately, the Web was built for human consumption, not for machine consumption - although everything on the Web is machine-readable, it is not machine-understandable[6]. We need the Semantic Web to express information in a precise, machine-interpretable form, ready for software agents to process, share and reuse it, as well as to understand what the terms describing the data mean. That would enable web-based applications to interoperate both on the syntactic and semantic level.

Note that it is Tim Berners-Lee (inventor of the WWW, URIs, HTTP and HTML) himself that pushes the idea of the Semantic Web forward. The father of the Web first envisioned a Semantic Web that provides automated information access based on machine-processable semantics of data and heuristics that use these metadata[7,8]. The explicit representation of the semantics of data, accompanied with domain theories (ontologies), will enable a Web that provides a qualitatively new level of service - for example, intelligent search engines, information brokers and information filters[9].

People from the World Wide Web Consortium (W3C) already developed new technologies for web-friendly data description[10]. Moreover, AI people have already developed some useful applications and tools for the Semantic Web[11].

We are introduce an implementation of Semantic Web concept on the e-Learning environment offered by our web-based e-learning system[12], which used by the Qatar University' students. The facilities that the application will provide include allowing e-learning

**Corresponding Author:** Fayed F. M. Ghaleb, Department of Mathematics, Faculty of Science, Ain Shams University

content to be created, annotated, shared and discussed, together with supplying resources such lecture notes, course description, documents, announcements, student papers, useful URL links, exercises and quizzes for evaluation of the student knowledge.

Recently, several researchers studied the issue of Web-based application. F. P. Rokou et al. distinguished three basic levels in every web-based application: the Web character of the program, the pedagogical background and the personalized management of the learning material[13]. They defined a web-based program as an information system that contains a Web server, a network, HTTP and a browser in which data supplied by users act on the system's status and cause changes. The pedagogical background means the educational model that is used in combination with pedagogical goals set by the instructor. The personalized management of the learning materials means the set of rules and mechanisms that are used to select learning materials based on the student's characteristics, the educational objectives, the teaching model and the available media.

Many works have combined and integrated these three factors in e-learning systems, leading to several standardization projects. Some projects have focused on determining the standard architecture and format for learning environments, such as IEEE Learning Technology Systems Architecture (LTSC), Instructional Management Systems (IMS) and Sharable Content Object Reference Model (SCORM). IMS and SCORM define and deliver XML-based interoperable specifications for exchanging and sequencing learning contents, i.e., learning objects, among many heterogeneous e-learning systems. They mainly focus on the standardization of learning and teaching methods as well as on the modeling of how the systems manage interoperating educational data relevant to the educational process[14].

IMS and SCORM have announced their content packaging model and sequencing model, respectively. The key technologies behind these models are the content package, activity tree, learning activities, sequencing rules and navigation model. Their sequencing models define a method for representing the intended behavior of an authored learning experience and their navigation models describe how the learner and system initiated navigation events can be triggered and processed.

Juan Quemada and Bernd Simon have also presented a model for educational activities and educational materials[15]. Their model for educational activities denotes educational events that identify the instructor(s) involved and take place in a virtual meeting according to a specific schedule. Rokou *et al.*[16] described the introduction of stereotypes to the pedagogical design of educational systems and appropriate modifications of the existing package diagrams of UML (Unified Modeling Language).

The IMS and SCORM models describe well the educational activities and system implementation, but not the educational contents knowledge in educational activities. Juan Quemada's and F. P. Rokou's models add more pedagogical background by emphasizing educational contents and sequences using the taxonomy of learning resources and stereotypes of teaching models. But the educational contents and their sequencing in these models are dependent on the system and lack standardization and reusability. Thus, we believe that if an educational contents frame of learning resources can be introduced into an e-learning system, including ontology-based properties and hierarchical semantic associations, then this e-learning system will have the capabilities of providing adaptable and intelligent learning to learners.

The hierarchical contents structure is able to show the entire educational contents, the available sequence of learning and the structure of the educational concepts, such as the related super- or sub- concepts in the learning contents. Furthermore, some of semantic relationships among the educational contents, such as 'equivalent', 'inverse', 'similar', 'aggregate' and 'classified', can provide important and useful information for the intelligent e-learning system.

For this purpose, ontology is introduced in our model. It can play a crucial role in enabling the representation, processing, sharing and reuse of knowledge among applications in modern web-based e-learning systems because it specifies the conceptualization of a specific domain in terms of concepts, attributes and relationships. Moreover, the number of ontology-centered researches has increased dramatically because popular ontological languages are based on Web technology standards, such as XML and RDF(S), so as to share and reuse it in any web-based knowledge system[17,18]. Thus, we have devised a model that provides the contents structure using an ontology for an adaptive and intelligent e-learning system.

**Semantic web overview:** There are a number of important issues related to the Semantic Web. Roughly speaking, they belong to four categories: Semantic Web languages, ontologies, semantic markup of Web pages and Semantic Web services.

**Semantic web languages:** In order to represent information on the Semantic Web and simultaneously make that information both syntactically and semantically interoperable across applications, it is necessary to use specific languages. It is important for Semantic Web developers to agree on the data's syntax and semantics before hard-coding them into their applications, since changes to syntax and semantics

necessitate expensive application modifications[19]. There are a lot of such languages around and most of them are based on XML (eXtensible Markup Language), XML Schemas, RDF (Resource Definition Framework) and RDF Schemas, all four developed under the auspices of W3C and using XML syntax[20]. While HTML is layout-oriented, XML is more structure-oriented. HTML is based on a fixed set of tags to format text, while in XML, tags are arbitrary (user-defined) and bear some semantic information themselves. Figure 1a and b shows an example of representing the same piece of information in HTML and in XML.

XML Schema provides the necessary framework for creating XML documents by specifying the valid structure, constraints, the number of occurrences of specific elements, default values and data types to be used in the corresponding XML documents, Fig. 1c. The encoding syntax of XML Schema is XML and just like XML itself XML Schema documents use namespaces that are declared using the xmlns attribute. Namespaces define contexts within which the corresponding tags and names apply.

```
<UL>
<LI>H.M. Deitel and P.J.Deitel <EM> C++ How To Program </EM>,
    Prentice Hall Publishing Co., New Jersey, 2003.
</UL>
```
(a)

```
<BOOK>
<AUTHOR> H.M. Deitel and P.J.Deitel </AUTHOR>
<TITLE> C++ How To Program </TITLE>
<PUBLISHER> Prentice Hall Publishing Co. </PUBLISHER>
<YEAR> 2003 </YEAR>
</BOOK>
```
(b)

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<xsd:element name="BOOK" type="BOOKTYPE"/>
<xsd:complexType name="BOOK_TYPE" >
<xsd:element name="AUTHOR" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
<xsd:element name="TITLE" type="xsd:string"/>
. . .
<xsd:attribute name="isbn" type="xsd:string"/>
</xsd:complexType>
</xsd:schema>
```
(c)

Fig. 1: (a) A piece of HTML code, (b) The same information in XML code and (c) An example of XML Schema

RDF is a framework to represent data about data (metadata) and a model for representing data about "things on the Web" (resources). It comprises a set of triples (O,A,V) that may be used to describe any possible relationship existing between the data – Object, Attribute and Value[10]. Alternatively, each RDF model can be represented as a directed labelled graph, as Fig. 2b, or in an XML-based encoding.

Regardless of the representation syntax, RDF models use traditional knowledge representation techniques order to provide better semantic interoperability (traditionally, O-A-V triplets are natural semantic units for representing a domain). Still, an RDF model just provides a domain-neutral mechanism to describe metadata, but does not define the semantics of any application domain. Figure 2a and b shows that each statement is essentially a relation between an object (a resource), an attribute (a property) and a value (a resource or free text). RDF Schema (RDFS) defines the vocabulary of an RDF model. It provides a mechanism to define domain-specific properties and classes of resources to which those properties can be applied, using a set of basic modeling primitives (class, subclass-of, property, subproperty-of, domain, range, type). An RDFS can be specified using RDF encoding, Figure 2c shows an example. However, RDFS is rather simple and it still doesn't provide exact semantics of a domain.

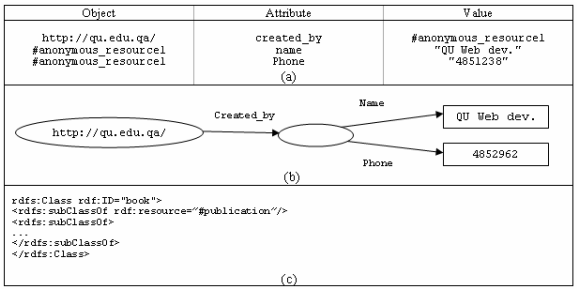


Fig. 2: (a) A simple RDF model and (b) the equivalent directed labelled graph and (c) An example of RDF Schema code

**Ontologies:** Ontology comprises a set of knowledge terms, including the vocabulary, the semantic interconnections and some simple rules of inference and logic for some particular topic[21]. Ontologies applied to the Web are creating the Semantic Web[22]. Ontologies provide the necessary armature around which knowledge bases should be built[23] and set grounds for developing reusable Web-contents, Web-services and applications[24]. Ontologies facilitate knowledge sharing and reuse, i.e. a common understanding of various contents that reaches across people and applications.

Technically, an ontology is a text-based piece of reference-knowledge, put somewhere on the Web for agents to consult it when necessary and represented using the syntax of an ontology representation language. There are several such languages around for representing ontologies[4] for an overview and comparison of them. It is important to understand that most of them are built on top of XML and RDF.

By 2004, the most popular higher-level ontology-representation languages were OIL (Ontology Inference Layer) and DAML+OIL[25,26]. An ontology developed in any such language is usually converted into an RDF/XML-like form and can be partially parsed even by common RDF/XML parsers[10]. Of course, language-specific parsers are necessary for full-scale parsing. There is a methodology for converting an ontology developed in a higher-level language into RDF or RDFS[9].

In early 2004, W3C has officially released OWL (Web Ontology Language) as W3C Recommendation for representing ontologies[10]. OWL is developed starting from description logic and DAML+OIL. The

increasing popularity of OWL might lead to its widest adoption as the standard ontology representation language on the Semantic Web in the future. Essentially, OWL is a set of XML elements and attributes, with well-defined meaning, that are used to define terms and their relationships (e.g., Class, equivalentProperty, intersectionOf, unionOf, etc.). OWL elements extend the set of RDF and RDFS elements and the owl namespace is used to denote OWL encoding. Figure 3 shows a piece of a simple ontology developed using the OWL language.

In practice, ontologies are often developed using integrated, graphical, ontology-authoring tools, such as Protégé-2000, OILed and OntoEdit[27]. They are used to develop new ontologies and modify existing ones. They let the author edit and develop ontologies concentrating on the domain's concepts and relationships, without worrying much about ontology-representation languages. The author can choose ontologies from a list, choose attributes and relations from another list, edit, add, remove and merge ontologies. The output is usually produced in a specific high-level ontology-representation language such as OWL, in RDF/RDFS, in HTML, or in plain text.

```
<owl:Class rdf:ID="Description">
   <rdfs:subClassOf rdf:resource="#Course"/>
   <owl:disjointWith rdf:resource="#Documents"/>
   <rdfs:seeAlso rdf:resource="#Useful_links_7"/>
</owl:Class>
```

Fig. 3: A simple ontology defined in OWL

**Semantic markup:** Ontologies merely serve to standardize and provide interpretations for Web content, but are not enough to build the Semantic Web. To make Web content machine-understandable, Web pages and documents themselves must contain semantic markup, i.e. annotations which use the terminology that one or more ontologies define and contain pointers to the network of ontologies, Fig. 4. Semantic markup persists with the document or the page published on the Web and is saved as part of the file representing the document/page. Services also must be properly marked-up, to make them computer-interpretable, use-apparent and agent-ready. They must contain pointers to the corresponding service ontologies.

Semantic markup of a Web page, document, or service might state that a particular entity is a member of a class, an entity has a particular property, two entities have some relationship between them and that descriptions from different people refer to the same entity. Typically, semantic markup is published using an XML encoding for a high-level ontology-representation language syntax[3,28].

Using ontologies as references in marking-up pages and services on the Semantic Web enables knowledge-based indexing and retrieval of services by intelligent agents, agent brokers and humans alike, as well as automated reasoning about the services, such as how to use them, what parameters to supply and what results to expect.

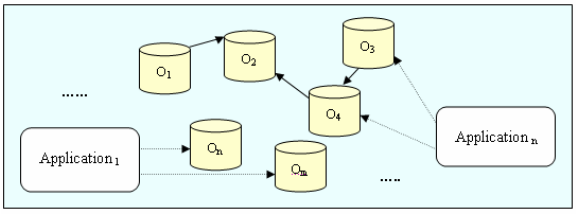


Fig. 4: Semantic markup provides mappings between Web pages and ontologies ($O_i$ - ontologies)

The annotation is done by using appropriate tools. These tools can be part-of or integrated with ontology-authoring tools, such as OIL tools[22]. They can also be standalone tools, such as the Knowledge Annotator tool[3]. Furthermore, they can operate through a COTS tool, as in the case of the Briefing Associate tool that uses MS PowerPoint GUI[28]. Finally, they can be integrated with specific Semantic Web applications. An example of this last approach is ITtalks, a fielded application that facilitates user and agent interaction for locating talks on information technology[11], which automatically generates DAML+OIL descriptions (markup) of user profiles when they register.

**Semantic web services:** Intelligent, high-level services like information brokers, search agents, information filters, intelligent information integration and knowledge management, are what the users want from the Semantic Web. They are possible only if a number of ontologies populate the Web, enabling semantic interoperation between the agents and the applications on the Semantic Web, i.e. semantic mappings between terms within the data, which requires content analysis.

One specific kind of ontology is necessary to enable high-level Semantic Web services- ontologies of services themselves[5]. These ontologies should include a machine-readable description of services (as to how they run), the consequences of using the service (e.g., the fee) and an explicit representation of the service logic (e.g., automatic invocation of another service). Services have their properties, capabilities, interfaces and effects, all of which must be encoded in an unambiguous, machine understandable form, to enable agents to recognize the services and invoke them automatically.

Semantic web framework for web-based e-learning system: In the following subsections, based on the Semantic Web technology and e-learning standards we describe our proposed framework for the web-based e-learning system, illustrated in Fig. 5.

**The web-based services:** Our model Fig. 5, provides the student with two kinds of contents, Learning content and Assessment content. Each content has different types of services such as:

**Learning services:** provide registration, online course, interactive tutorial, course documents (is a repository for files that the instructor have made available to the student as a part of your course), announcements (displays information to the students that the instructors of the course want him to know), links (displays a list of useful URL links that have been identified by the course instructors), student papers (students can post/upload requests files to the instructor) and Semantic search (helps the student to search for resources).

**Assessment services:** provide exercises and quizzes for evaluation of the student knowledge.

During the learning process, a dynamic selection presentation of both contents will be accomplished.

On other hand, our web-based e-learning system allows instructors to create his course websites through a browser and monitoring the student's performance. they have many services and tools such as: publish documents in any format (Word, PDF, Video, ...) to the students, manage a list of useful links, compose exercises/quizzes, make announcements and have students submit papers. To illustrate the services architecture, we will now go through an e-learning scenario. A student first searches for an online course: the broker handles the request and returns a set of choices satisfying the query. If no course is found, the user can register with a notification service. Otherwise, the user may find a suitable course among the offerings and then makes a final decision about registering for the course.

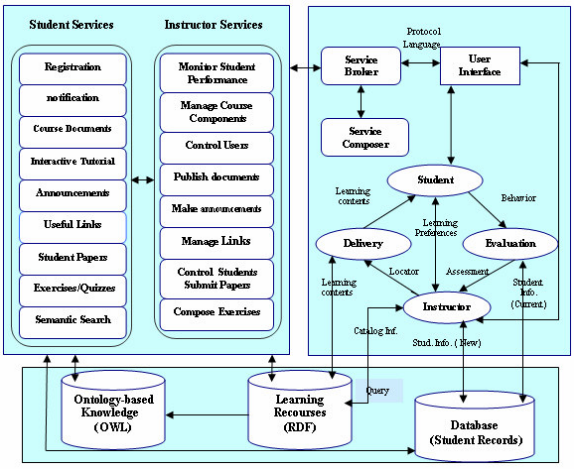


Fig. 5: Proposed framework for web-based e-learning system

Processing the registration can be seen as a complex service involving registering with the system, creating a confirmation notification, creating a student account (authentication/ authorization) and providing learning materials. Once all these in place, the student can start the course. As part of the course, a student will be logging on and checking his learning agenda (e.g.

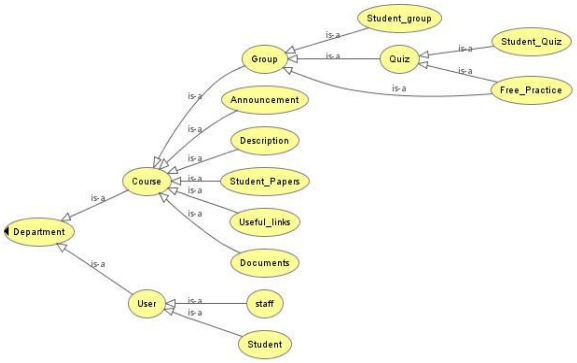


Fig. 6: A snapshot of the proposed ontology using Protégé 2000

next assignment due). This request is answered by combining several sources of information, such as course schedule, current date and student progress to date (e.g. completed units).

**The ontology-based model:** Before describing our ontology-based model, we will discuss learning environments illustrated in Fig. 5. Course sequencing generally starts with the student entity component that receives the learning contents, while the student's behavior is being observed. The instructor sends queries to the learning resources to search for learning content that is appropriate for the student entity component. The ontological knowledge is added to the learning resources as a resource for contextual learning and it may be searched by means of queries. The student's performance is measured by the evaluation component and the result is stored in the student records database. The data in it can be used by the instructor component to locate a new content.

Searching learning resources and sequencing a course can be done using a knowledge base of learning resources and a delivery component. To implement the knowledge base, first of all, the leaning resources have to be described by means of metadata. The metadata consists of the contextual knowledge of the learning resources, i.e., an ontology in our model. It contains the general representation of the structural knowledge on specific domains, such as computer science, mathematics, biology and so on.

The ontology can be used for adaptive learning to retrieve the context of a course and to structure the contents. Also the metadata actually consists of the framing description of each learning object of a subject, i.e., the modularized content, which is linked to the concept of the ontology. For instructors to be able to sequence courses and create exercises adaptively, the suitability of different approaches has to be analyzed based on the relationships between the resources and their descriptions. Figure 6 shows a snapshot of our web-based e-learning ontology with the classes and properties in the Protégé 2000 ontology editor and Fig. 7 shows a portion of the ontology source in the OWL language.

```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns="http://www.owl-
    ontologies.com/Ontology1131457231.owl#" xmlns:p1="http://www.owl-
    ontologies.com/assert.owl#" xml:base="http://www.owl-
    ontologies.com/Ontology1131457231.owl">
  <owl:Ontology rdf:about="" />
- <owl:Class rdf:ID="Group">
- <rdfs:subClassOf>
  <owl:Class rdf:ID="Course" />
    </rdfs:subClassOf>
    </owl:Class>
- <owl:Class rdf:ID="staff">
- <rdfs:subClassOf>
  <owl:Class rdf:ID="User" />
    </rdfs:subClassOf>
```

Fig. 7: A portion of the proposed ontology in OWL language

**Implementation:** The main agents used in our system are: Student and Instructor, both of them are implemented as PHP classes, as illustrated in Fig. 5. Users are served by the appropriate agents, which parse the metadata and tailor the user interface to satisfy the user's needs, whether student or instructor. The agents interact and communicate between each other by means of PHP, MySQL database and using the Apache Web Server. Figure 8, show a snapshot of our proposed system.

Users will add any metadata to a document referenced via the RDF learning resources repository through dynamic PHP web pages. For the end-user, this process of annotation is identical to the action of filling out fields in a Web form. After the user submits the form, the application automatically converts this additional information to a set of RDF statements using the RAP API and then adds them to the existing RDF statements for this document in the repository. Because the RDF specifications provide an XML syntax for writing down and exchanging RDF statements (called RDF/XML), the repository is implemented as a set of RDF/XML files. However, the RDF/XML syntax is quite complex and developing an RDF parser is not a trivial task.

Motivated by the need for an RDF parser, we are using a Semantic Web toolkit called RAP for developing our application. In the following subsections, we will illustrate the RAP API in more details.

**What is RAP?:** RAP - RDF API for PHP is a Semantic Web toolkit for PHP developers. It offers features for parsing, manipulating, storing, querying, serving and serializing RDF graphs. RAP was started as an open source project by the Freie Universität Berlin in 2002 and has been extended with code contributions from the Semantic Web community. The core of RAP are two implementations of statement storages which hold RDF graphs either in memory or in a relational database. Around these storages RAP provides rich programming interfaces for manipulating RDF graphs on different abstraction layers. Furthermore, RAP supports RDFS inference as well as some OWL entailments, allowing programmers to work with implicit (virtual) statements. Various tools complement the RAP package: an up-to-date RDF/XML parser, further I/O modules for alternative serialization techniques (i.e. N3, N-Triple, RDF embedded in XHTML), an integrated RDF server

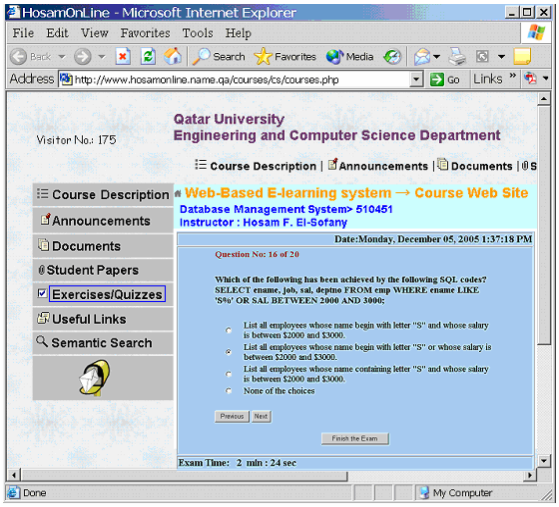


Fig. 8: A snapshot of the proposed system

and a graphical user-interface for managing database-backed RDF models as well as an implementation of the RDQL query language.

**Working with RDF graphs:** In RAP, RDF graphs are represented as instances of class Model. The elements within a Model are Statements; each Statement comprises three Nodes: the subject, predicate and object. A Node represents a Resource identified by a URI, a BlankNode (also known as bNode ), or a Literal. RAP offers three programming interfaces for manipulating RDF graphs: the statement-centric Model API which allows manipulating an RDF graph as a set of statements, the resource-centric ResModel API for manipulating an RDF graph as a set of resources having properties and the ontology-centric OntModel API which provides extra functionality for handling ontologies.

**Statement-centric programming interface:** The Model API exposes an RDF graph as a set of RDF statements. This API is very similar to the statements storage structure and leads to a very small overhead in accessing the graph. The core methods for modifying RDF graphs support adding, deleting and replacing of single statements inside a graph. StatementIterators allow sequential access to all statements within a graph. The most significant part of this API are the find()and findAsIterator() methods providing a fast and straightforward way to query RDF statements. The former method delivers a new model, the latter returns an iterator over all the statements of the queried model, which match the triple pattern (S, P, O). S/P/O can either be instances of the subclasses of Node or be equal NULL (meaning anything). For example, the pattern (S, NULL, NULL) with S being an instance of Resource will match all statements describing this particular resource S.

**Resource-centric programming interface:** The resource-centric API represents an RDF graph as a set

of resources having properties. This interface enables to manipulate and navigate through an RDF graph in a much more comfortable way. For example, if a resource is known to be of type rdf:Collection, then viewing the corresponding resource as a collection that allows easier access to its members without having to deal with the sophisticated list-structure.

This ResModel API is implemented on top of the statement-centric interface. Thus, each ResModel always has an underlying in-memory or persistent statement store and is only providing a resource-centric view on this model. To ensure data consistency, there is no caching being done between the layers. Each method call is translated into a series of find(), add(), or remove() calls of the underlying model. Therefore, working with ResModels is slightly slower than using the Model API directly, but offers the comfort of accessing the information about resources in an object-orientated way. The ResModel API is very similar to the Jena Model API[29] allowing programmers, which are used to Jena, to readily write RAP code.

**Ontology-centric programming interface:** The ontology-centric API is an extension of the resource-centric interface. It adds support for ontological primitives: classes (in a class hierarchy), properties (in a property hierarchy) and individuals. The properties defined in the ontology language map to accessor methods. For example, if a resource is known to be an rdfs:Class in the given RDF graph it has a method to list its super-classes which correspond to the values of the rdfs:subClassOf property. This interface supports not only the RDF-Schema ontology language but also parts of OWL by using a loadable vocabulary. Thus, a new class is generated as an rdfs:Class or an owl:Class depending on the vocabulary currently loaded.

**Storing RDF graphs:** The core of RAP are two implementations of statement storages, which hold RDF graphs either in memory or in a persistent store. Working with in-memory models, however, has one major disadvantage: after finishing the execution of a PHP script, all models created and manipulated would be lost, unless saved to a file. But even if serialized to file, the document containing RDF data would have to be parsed any time a PHP script would be executed and additionally the search index built if efficient queries should be performed. Both processes are rather time-consuming, especially while working with large in-memory models. To address this problem RDF API for PHP supports persistent storage of RDF models in a relational database. Storing models in a database not only saves main memory, but moreover allows quick access to RDF data by using the internal indexing and query optimization capabilities of the database. The core of RAP's database backend is built by two classes: DbStore and DbModel. The former is used to set the database connection as well as create, store, list and retrieve RDF models, whereas the latter provides methods for manipulating each model.

In the RAP toolkit there is also RDF DB Utils included - a graphical user-interface for managing database-backed RDF models. It allows convenient browsing through a selected persistent model to view, edit, or delete statements.

**RDQL (RDF data query language):** RDQL[30] is a query language for extracting information from RDF graphs. Queries are formulated by specifying a subgraph, with missing parts having assigned variable names, which is matched against an RDF graph. RDQL is implemented in several RDF toolkits and has been submitted to the W3C for standardization[30]. In order to ensure the greatest possible compatibility RAP's RDQL implementation follows the current de facto standard set by the Jena[29] implementation.

An RDQL query consists of a graph pattern, expressed as a list of triple patterns (S, P, O). S/P/O can either be named variables or RDF values (URIs or Literals). Literals may additionally be constrained by their language and datatype. Furthermore, an RDQL query can have a set of constraints on the values of query variables. Filter expressions supported by RAP are: arithmetic conditions, string equality expressions and Perl-style regular expressions. Multiple constraints can be combined using logical operators. A list of variables required in the answer set is specified in the SELECT clause of an RDQL query. To make the query easier to read and write for humans, RDQL provides a way to shorten the length of URIs by defining a string prefix. Consider the following example:

```
SELECT ?student
WHERE (?student, <info:age>, ?age)
AND ?age >= 2o
USING info FOR <http://example.org/people#>
```

The above triple pattern matches all statements having predicate http://example.org/ people#age. The variable ?student will be bound to the label of the statement subject, the variable ?age to the literal value of the statement object. The query returns all values of ?student from statements matching the specified pattern and having the object value greater or equal 20.

## CONCLUSION

The main contribution of this study was our outline framework for web-based e-learning system, using the Semantic Web technology. Our architecture including various services and tools in the context of a semantic portal, such as: course registration, uploading course documents and student assignments, interactive tutorial, announcements, useful links, assessment and simple semantic search. A metadata-based ontology is introduced for this purpose and added to our model. The OWL language is used to develop our ontologies. In these ontologies, the actual resources and properties specified in the RDF models are defined. The Protégé 2000 ontology editor is used to create the e-learning ontology classes and properties.

A list of the technologies used in the implementation of our web-based e-learning system includes PHP Platform, Apache Web Server, MySQL database and RAP Semantic Web Toolkit. We believe that there are two primary advantages of our Semantic web-based framework. One is that the proposed model, which contains a hierarchical contents structure and semantic relationships between concepts, can provide related useful information for searching and sequencing learning resources in web-based e-learning systems. The other is that it can help a developer or an instructor to develop a learning sequence plan by helping the instructor understand the why and how of the learning process.

## REFERENCES


1. Barker, P., 2000. Developing Teaching Webs: Advantages, Problems and Pitfalls. Educational Multimedia, Hypermedia & Telecommunication (AACE) Conf., 2000.
2. Drucker, P., 2000. Need to Know – Integrating e-Learning with High Velocity Value Chains. Delphi Group White Paper. www.delphigroup.com.
3. Heflin, J. and J.A. Hendler, 2001. Portrait of the semantic web in action. IEEE Intelligent Systems, 16: 54-59.
4. Gómez-Pérez, A. and O. Corcho, 2002. Ontology languages for the semantic web. IEEE Intelligent Systems, 17: 54-60.
5. McIlraith, S.A., T.C. Son and H. Zeng, 2001. Semantic web services. IEEE Intelligent Systems, 16: 46-53.
6. Lassila, O., 1998. Web Metadata: A matter of semantics. IEEE Internet Computing, 2: 30-37.
7. Berners-Lee, T., J. Hendler and O. Lassila, 2001. The semantic web. Scient. Am., 284: 34–43.
8. Berners-Lee, T., M. Fischetti and T.M. Dertouzos, 1999. Weaving the web: The original design and ultimate destiny of the world wide web by its inventor. San Francisco: Harper.
9. Decker, S., S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, 2000. The semantic web: The roles of XML and RDF. IEEE Internet Computing, 4: 63-74.
10. W3C site: http://www.w3c.org. ( www.w3.org/XML , www.w3.org/RDF, www.w3.org/TR/2004/ REC-owl-features-20040210/) and http://www.w3.org/2000/10/swap/Primer.html.
11. Scott, C.R., T. Finin, A. Joshi, Y. Peng, C. Nicholas and I. Soboroff *et al*., 2002. ITtalks: A case study in the semantic web and DAML+OIL. IEEE Intelligent Systems, 17: 40-47.
12. Hosam, F.El-Sofany, A.M. Hasnah, J.M. Jaam and F.F.M. Ghaleb, 2005. A web-based e-learning system experiment. Proc. Intl. Conf. E-Business and E-learning, PSUT, Amman-Jordan.
13. Rokou, F.P. *et al.*, 2004. Modeling web-based educational systems: process design teaching model. Educat. Technol. Soc., 7: 42-50.
14. Adelsberger, H. *et al*., 2003. The Essen model: A step towards a standard learning process. http://citeseer.ist.psu.edu/515384.html.
15. Quemanda, J. and B. Simon, 2003. A use-case based model for learning resources in educational mediators. Educat. Technol. Soc., 6: 149-163.
16. Merrill, M.D., 2003. Knowledge objects and mental-models. http://reusability.org/read.
17. Sure, Y. et al., 2002. Methodology for development and employment of ontology based knowledge management applications. ACM SIGMOD Record, 31: 18-23.
18. Brewster, C. *et al*., 2004. Knowledge representation with ontologies: The present and future. IEEE Intelligent Systems, 19: 72-81.
19. Wuwongse, V., C. Anutariya, K. Akama and E. Nantajeewarawat, 2002. XML declarative description: A language for the semantic web. IEEE Intelligent Systems, 17: 54-65.
20. Klein, M., 2001. Tutorial: The semantic web-XML, RDF and relatives. IEEE Intelligent Systems, 16: 26-28.
21. Hendler, J., 2001. Agents and the semantic web. IEEE Intelligent Systems, 16: 30-37.
22. Fensel, D., F. van Harmelen, I. Horrocks, D.L. McGuinness and P.F. Patel-Schneider, 2001. OIL: An ontology infrastructure for the semantic web. IEEE Intelligent Systems, 16: 38-45.
23. Swartout, W. and A. Tate, 1999. Ontologies, Guest Editors' Introduction. IEEE Intelligent Systems, 14: 18-19.
24. Devedzic, V., 2001. Knowledge modeling- State of the Art. Integrated Computer-Aided Engg., 8: 257-281.
25. Horrocks, I., D. Fensel, J. Broekstra, S. Decker, M. Erdmann and C. Goble *et al.*, 2002. The ontology inference layer oil. Tech. Report, Vrije Universiteit, Amsterdam. Retrieved March 19, 2002. http://www.ontoknowledge.org/oil/TR/oil.long.html
26. Horrocks, I. and F. van Harmelen, 2000. Reference description of the DAML+OIL ontology markup language. http://www.daml.org/2000/12/reference.html.
27. Protégé, 2000. http://protege.stanford.edu/. OILed: http://img.cs.man.ac.uk/oil and OntoEdit: http://ontoserver.aifb.uni-karlsruhe.de/ontoedit.
28. Tallis, M., N.M. Goldman and R.M. Balzer, 2002. The briefing associate: Easing authors into the semantic web. IEEE Intelligent Systems, 17: 26-32.
29. Carroll, J. *et al.*, 2003. Jena: Implementing the Semantic Web Recomandations. Bristol. 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-146.pdf
30. Seaborne, A., 2004. RDQL-A Query Language for RDF, W3C Member Submission. 9 Jan. 2004. http://www.w3.org/Submission/RDQL/.