Chapter 10

DEVELOPING COURSE MANAGEMENT SYSTEMS USING SEMANTIC WEB TECHNOLOGIES

Jorge Cardoso
Department of Mathematics and Engineering, University of Madeira, 9000-390, Funchal, Portugal, jcardoso@uma.pt

Abstract. While semantic Web technologies have reached a certain level of maturity, the industry is still skeptical about its potential and applicability. Many vendors seem to be adopting a "wait-and-see" approach while emerging standards and solutions become more fully developed. The industry and its main players are waiting to see how real-world applications can benefit from the use of semantic Web technologies. The success of the Semantic Web vision is dependant on the development of practical and useful semantic Web-based applications. To demonstrate the applicability and the benefits of using semantic Web technologies, we have developed a real-world application, a Semantic Course Management System (S-CMS), entirely based on the semantic Web that uses the latest technologies of this field such as OWL, RQL, RDQL, and SWRL.

1. INTRODUCTION

Many researchers believe that a new Web will emerge in the next few years based on the ongoing large-scale research and developments on the semantic Web. Nevertheless, the industry and its main players are adopting a "wait-and-see" approach to see how real-world

applications can benefit from semantic Web technologies (Cardoso, Miller et al. 2005). The success of the semantic Web vision (Berners-Lee, Hendler et al. 2001) is dependant on the development of practical and useful semantic Web-based applications.

While the semantic Web has reached considerable stability from the technological point of view, with the development of languages to represent knowledge (such as OWL (OWL 2004)), to query knowledge bases (RQL (Karvounarakis, Alexaki et al. 2002) and RDQL (RDQL 2005)), and to describe business rules (such as SWRL (Ian Horrocks, Peter F. Patel-Schneider et al. 2003)), the industry is still skeptical about its potential. For the semantic Web to gain considerable acceptance from the industry it is indispensable to develop real-world semantic Web-based applications to validate and explore the full potential of the semantic Web (Lassila and McGuinness 2001). The success of the semantic Web depends on its capability to support applications in commercial settings (Cardoso, Miller et al. 2005).

In several fields, the technologies associated with the semantic Web have been implemented with a considerable degree of success. Examples include semantic Web services (OWL-S 2004), tourism information systems (Cardoso 2004), semantic digital libraries, (Shum, Motta et al. 2000), semantic Grid (Roure, Jennings et al. 2001), semantic Web search (Swoogle 2005), and bioinformatics (Kumar and Smith 2004).

To take the development and widespread character of semantic Web applications a step further, we have developed a Course Management System (CMS) (Mandal, Sinha et al. 2004) based on an infrastructure entirely designed using the technologies made available by the semantic Web, namely OWL, RQL, RDQL, SPARQL, Bossom (Bossom 2005), and SWRL.

CMSs are becoming increasingly popular. Well-known CMSs include Blackboard.com and WebCT.com whose focus has centered on distance education opportunities. Typically, a CMS includes a variety of functionalities, such as class project management, registration tools for students, examinations, enrolment management, test administration, assessment tools, and online discussion boards (Meinel, Sack et al. 2002).

The system that we have developed is part of the Strawberry project[1] and explores the use of semantic Web technologies to develop an innovative Semantic Course Management System (S-CMS). The S-CMS provides a complete information and management solution for

students and faculty members. Our focus and main objective is to automate the different procedures involved when students enroll or register for class projects. Managing a large course and its class projects is a complex undertaking. Many factors may contribute to this complexity, such as a large number of students, the variety of rules that allow students to register for a particular project, students' background, and students' grades.

2. S-CMS ARCHITECTURE

The architecture of our system is composed of seven distinct layers (Figure 1): source layer, connection layer, instance layer, query layer, inference layer, application layer, and presentation layer. The layers are articulated in the following way. The source layer includes all the data and information needed by our semantic course management system. It typically includes data stored in relational databases (other types of data source are also supported). At this level, we can find information which describes which faculty members teach which courses, which students are enrolled for a particular course, which students are enrolled in a degree, personal information about students and teachers, etc. The next layer, the connection layer, is responsible for connecting to the data sources, using a variety of protocols.

The instance layer is the first layer that uses semantic Web technologies. It is responsible for managing ontologies describing university domain information such as courses, students, projects, and teachers. It is also in charge of transforming the data and information extracted from the data sources into a set of ontology instances. Essentially, this layer creates a knowledge-base that will be used by the upper layers. It gets the local schema of heterogeneous data sources under consideration and creates a unique and virtual global scheme (i.e., an ontology). Since all data sources refer to the same ontology, theoretically there are not syntactic neither semantic conflicts.

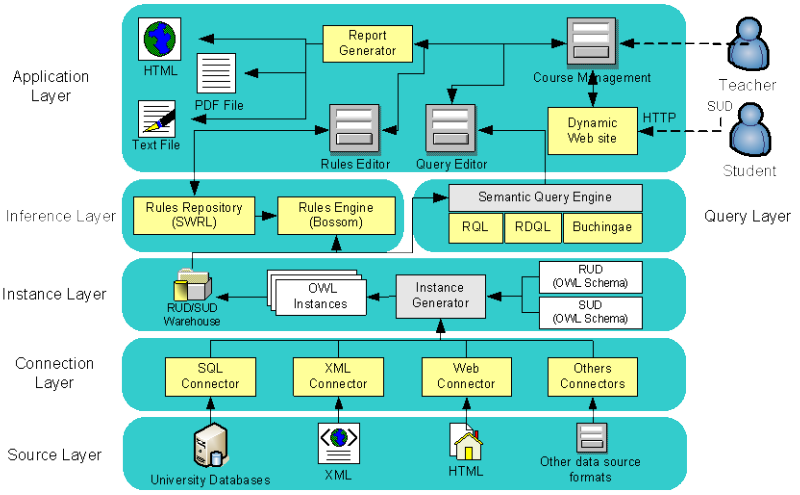

Figure 10-1. S-CMS Architecture

The query layer supplies an interface that allows querying the knowledge-base. The inference layer allows carrying out inference using semantic rules on the knowledge-base. For example, it is possible to inquire if all the students enrolled in a project have all passed on the Knowledge Engineering course. The application layer provides the Course Management System *per se* to teachers and students. Teachers are able to create projects associated with courses and define semantic enrolment rules. Students are able to specify that they wish to enroll for a specific project. Additionally this layer gathers the knowledge inferred from applying the semantic rules to the semantic knowledge-base and formats it into a suitable presentation medium (such as PDF or HTML). In the next section we will describe each of the layers of our architecture in detail.

2.1 Source layer

Course management systems need to access a variety of data sources to access data and information about students, teachers, degrees, physical resources (such as class rooms and computing facilities), courses, and grades. To develop robust course information management applications it is important to develop an architecture that can access and integrate unstructured, semi-structured, and structured data. We will see that the use of an ontology will allow us to integrate data with different structures, resolving the structural heterogeneity of data sources.

Data sources are uniquely identifiable collections of stored data, called data sets for which there exist programmatic access and for which it is possible to retrieve or infer a description of the structure of the data, i.e. its schema. We have recognized various data sources that need to be considered when integrating university management systems: flat files, HTML Web pages, XML, and relational databases.

At the University of Madeira we have identified two main types of data sources that needed to be accessed in order to retrieve relevant information about courses and projects: HTML and databases data sources. Therefore, we have developed two Eclipse plug ins to access these types of sources (see next section).

HTML. Most, if not all, the Universities have Web sites for storing and advertising the description of their course, degrees, and projects. Course management systems require integrating Web-based data sources in an automated way for querying, in a uniform way, across multiple heterogeneous Web sites, containing university related information.

Databases. In universities, it is almost unavoidable to use databases to produce, store, and search for critical data. Yet, it is only by combining the information from various database systems that course management systems can take a competitive advantage from the value of data. Different university departments use distinct data sources. To develop course management systems, the most common form of data integration is achieved using special-purpose applications that access data sources of interest directly and combine the data retrieved with the application itself. While this approach always works, it is expensive in terms of both time and skills, fragile due to the changes to the underlying sources, and hard to extend since new data sources require new fragments of code to be written. In our architecture, the use of semantics and ontologies to construct a global view makes the integration process automatic, and there is no need for a human integrator. The University of Madeira database that we have used had around 200 tables, 600 views, a diversity of data types and a large dataset. The number of students is in the range of 13 000. One main problem that we found is that there was no documentation available describing the tables, attributes, and views.

2.2 Connection layer

The connection layer maintains a pool of connections to several data sources (in our implementation we use relational databases and HTML online Web pages). We use a connection layer to achieve two goals: abstraction and efficiency. On the one hand, the connection layer adds a level of abstraction over the data sources and it is responsible for presenting a single interface to the underlying data sources. On the other hand, the connection layer provides connection pooling to considerably increase application processing. When the instance layer requires data from the connection layer, connections to the data sources must be established, managed, and then freed when the access is complete. These actions are time and resource consuming. The use of a connection layer minimizes the opening and closing time associated with making or breaking data source connections. For the S-CMS application, we have developed three Eclipse plug ins. Two of the plug ins are customized to access MySQL and Microsoft SQLServer 2000 databases, while the third plug in is dedicated to retrieve information from HTML Web pages.

At this state the major difficulty that we had was to obtain a copy of the University database from the administrative department with real data. The authorization to use the database has taken more than 3 months to arrive. Furthermore, the copy of the database that was given to us had the data fields with sensitive information altered. Examples these fields included students' PIN and phone numbers.

2.3 Instance layer

Data integration is a challenge for course management systems since they need to query across multiple heterogeneous, autonomous, and distributed (HAD) university data sources produced independently by multiple organizations units. Integrating HAD data sources involves combining the concepts and knowledge in the individual university data sources into an integrated view of the data. The construction of an integrated view is complicated because organizations store different types of data, in varying formats, with different meanings, and referenced using different names (Lawrence and Barker 2001).

We have identified four types of information heterogeneity (Sheth 1998; Ouskel and Sheth 1999) that may arise when we try to integrated HAD university data sources:

1. **System heterogeneity**: Applications and data may reside in different hardware platforms and operating systems.
2. **Syntactic heterogeneity**: Information sources may use different representations and encodings for data. Syntactic interoperability can be achieved when compatible forms of encoding and access protocols are used to allow information systems to communicate.
3. **Structural heterogeneity**: Different information systems store their data in different document layouts and formats, data models, data structures and schemas.
4. **Semantic heterogeneity**: The meaning of the data can be expressed in different ways leading to heterogeneity. Semantic heterogeneity considers the content of an information item and its intended meaning.

Approaches to the problems of semantic heterogeneity should equip heterogeneous, autonomous, and distributed software systems with the ability to share and exchange information in a semantically consistent way (Sheth 1999).

To allow the seamless integration of HAD university data sources rely on the use of semantics. Semantic integration requires knowledge of the meaning of data within the university data sources, including integrity rules and the relationships across sources. Semantic technologies are designed to extend the capabilities of data sources allowing to unbind the representation of data and the data itself and to give context to data. The integration of university data sources requires thinking not of the data itself but rather the structure of those data: schemas, data types, relational database constructs, file formats, etc.

As a solution to the problem of integrating heterogeneous data sources we provide a uniform access to data. To resolve syntactic and structural heterogeneity we map the local data sources schema into a global conceptual schema. Since semantic problems can remain, we use ontologies to overcome semantic heterogeneity. An ontology is an agreed vocabulary that provides a set of well-founded constructs to build meaningful higher level knowledge for specifying the semantics of terminology systems in a well defined and unambiguous manner. Ontologies can be used to increase communication either between humans and computers. The three major uses of ontologies (Jasper and Uschold 1999) are:

1. To assist in communication between humans.

2. To achieve interoperability and communication among software systems.
3. To improve the design and the quality of software systems.

The main component of the instance layer is the Instance Generator. The data extracted by the connection layer is formatted and represented using two different ontologies, the RUD (University Resource Descriptor) and SUD (Student University Descriptor). In the following sections we describe the two ontologies and their instances.

2.3.1 Ontology Creation

To deploy our ontologies we have adopted the most prominent ontology language, OWL (OWL 2004). The development of an ontology-driven application typically starts with the creation of an ontology schema. Our ontology schemas contain the definition of the various classes, attributes, and relationships that encapsulate the business objects that model a university domain. After conducting an analysis of ontology editors, we have selected Protégé (Knublauch, Fergerson et al. 2004) to construct our ontologies.

Since the objective of S-CMS application was to develop a system which provided the ability to a student enroll in a course projects, the inference over OWL documents (RUD and SUD) needed to answer to questions which included:

- Who are the teachers and students?
- What courses are offered by a department?
- Which courses are assigned for a specific teacher?
- For which courses a student is enrolled?
- Which projects are assigned to a course?
- What are the students' grades of taken courses?

The RUD and SUD ontologies have the following characteristics.

RUD (University Resource Descriptor). A University Resource Descriptor is a semantic knowledge-base that integrates information coming from several external data sources spread throughout the University of Madeira. As we have seen in section 2.1, data describing important resources to our S-CMS application were stored in relational databases or HTML Web pages. Our RUD integrated information about the physical recourses of the university, classes, courses and degrees offered, faculty members, students enrolled at the

university, etc. All the information is represented in OWL. The RUD schema has much more information than the one that comes from the various data sources since it establishes hundreds of relationships between concepts. The relationships are fundamental and will be used by the inference layer to infer new knowledge.

SUD (Student University Descriptor). A Student University Descriptor is a resource that describes a university student. Each student of the university has a SUD. A SUD includes information such as the student's name, ID, courses taken, courses enrolled, degree, telephone number, age, etc. In our architecture, each SUD is represented in OWL.

Students can make available their SUD using two alternatives. They can simply put their SUD in their university home page or they can rely on the SUD management system to manage and advertise their SUD. The idea of SUDs was inspired by the concept of RSS (RSS 2005) (Really Simple Syndication). The technology of RSS allows Internet users to subscribe to websites that have provided RSS feeds; these are typically sites that change or add content regularly. Unlike RSS subscriptions, SUD do not include information about news but about students. Figure 4 illustrates part of the SUD schema for students.

```
(...)
<owl:Class rdf:ID="Student">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype=

"http://www.w3.org/2001/XMLSchema#int">1
        </owl:cardinality>

        <owl:onProperty>
          <owl:DatatypeProperty
rdf:ID="AverageScore"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Person"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty
rdf:ID="StudentID"/>
        </owl:onProperty>
```

```
          <owl:cardinality rdf:datatype=

"http://www.w3.org/2001/XMLSchema#int">1
          </owl:cardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>
(...)
```

Figure 10-2. SUD schema represented in OWL

2.3.2 Ontology population

By ontology population we refer to a process, where the class structure of the RUD and SUD ontologies already exists and is extended with instance data (individuals). This can be done either by a computer or by a human editor. In our case, the RUD and SUD instances are created automatically by the instance generator. Figure 5 illustrates the SUD instance created for the student Lee Hall.

```
<Student rdf:ID="LeeHall2041999">
(...)
  <StudentID rdf:datatype=

"http://www.w3.org/2001/XMLSchema#nonNegativeInteg
er">
   2041999
  </StudentID>
  <StudentName>Lee Hall</StudentName>
  <Degree>Computer Science </Degree>
  <StudentEmail>lhall@mail.pt
  </StudentEmail>
  <Studies>
    <Subject rdf:ID="Semantic_Web">
      <SubjectName>Semantic Web</SubjectName>
(...)
    </Subject>
  </Studies>
(...)
</Student>
```

Figure 10-3. A SUD instance represented in OWL

2.3.3 Difficulties in creating and populating the ontology

During the process of creating the RUD and SUD ontology and generating the instance to populate the ontologies schema we have found the following difficulties:

- Since the University database has 200 tables and there was no documentation available it was difficult to identify and decide the relevant classes, subclasses and properties.
- There was a considerable amount of duplicate values in the database. Selecting the most appropriate values involved also a considerable ffort.
- There was no direct mapping between OWL classes and the corresponding database tables. The same happen with the HTML Web pages.
- Make use OWL expressiveness, namely with the especial properties such as transitivity, symmetry, inverse, functional, inverse functional.
- The examples of OWL documents that we have found in the Internet were few and simple and did not represent the true complexity of OWL documents.
- To take advantage of the expressive capabilities of OWL we had to increase the complexity of representation which was difficult to manage.
- As the classes and properties are connected in a recursive fashion we could not simply create all instances of a certain class because other might need to already have been defined.
- The tool that we have used to create ontologies, Protégé, although being very intuitive in its usage had an error when translating OWL documents.
- The Jena API (Jena 2005), used to programmatically manipulate OWL ontology models, did not load the models after a change.
- The generator of instances developed was not generic. It specific to this particular RUD and SUD schema, this brings some disadvantages in further enhancements because any change in the schema can led to modification at the programmatic level.

2.4 Query layer

The query layer provides a query interface to the knowledge-base formed with all the RUD and SUD ontology instances automatically generated. The query interface understands three distinct semantic query languages: RQL (RDF Query Language), RDQL (RDF Data Query Language), Buchingae, SPARQL. These languages allow querying ontology classes, navigating to its subclasses, and discovering the resources which are directly classified under them. Our initial objective was to make available to users a language that would enable us to query the native representation of our knowledge-base, i.e. OWL, but no suitable query language of this type exists yet.

Using this layer, teachers are able to query student and university information. For example, the following query expressed in RDQL allows selecting the students that have a GPA greater than 4.0 marks.

```
SELECT ?x,?c,?z
WHERE
   (?x <http://apus.uma.pt/RUD.owl#HasGPA> ?y),
   (?x <http://apus.uma.pt/RUD.owl#Studies> ?c),
   (?y <http://apus.uma.pt/RUD.owl#Value> ?z)
   AND ?z>4.0
```

As another example, the following query expressed in Buchingae allows a teacher to inquire about the students that are enrolled in a specific course,

```
query qu is p:Studies(?st, ?course) and
           p:Teaches(?prof, ?course);
```

2.5 Inference layer

We have implemented a rule management system to extract and isolate course management logic from procedural code. Since the rules associated with the enrollment of students for class projects may change quite often, these changes cannot be handled efficiently by representing rules embedded in the source code of the application logic. The option to detach enrolment rules from the source code gives teachers an effective way of creating the rule base and of building and changing rules. The following list considers some advantages of separating enrolment rules from the application logic:

1. Student enrolment rule reuse across other course management systems.

2. A better understanding of enrolment rules through separate business rules.
3. Documentation of enrolment decisions through rules.
4. Lower application maintenance costs.
5. Ease of changing enrolment rules by using visual tools.

In S-CMS, the rules are defined in SWRL (Semantic Web Rule Language) or Buchingae. They correspond to axioms about classes (concept) or their properties of the instance stored in the OWL knowledge-base. By applying these rules to the set of facts it is possible to infer new facts.

SWRL was designed to be the rule language of the semantic Web enabling rule interoperation on the Web. SWRL is based on a combination of the OWL DL and OWL Lite. It provides the ability to write Horn-like rules expressed in terms of OWL for reasoning about OWL individuals.

Since SWRL rules are fairly well-known, we give an example of a Buchingae rule. The rule states that only students that have taken the course Knowledge Engineering (CS6100) and Logic Programming (CS6550) are eligible to enroll for a the class project of the course Introduction to Semantic Web (CS8050),

```
prefix builtin =
       http://www.etri.re.kr/2003/10/bossam-
builtin#;
prefix RUD = http://apus.uma.pt/RUD.owl#;
namespace is http://www.etri.re.kr/samples#;

rulebase rb01
{
    (...)
    rule R01 is
      if
        classTaken(?x, RUD:CS6100) and
        classTaken(?x, RUD:CS6550)
         then
           eligible(?x, RUD:CS8050)
}
```

A large number of rule engines are available as open source software. Some of the most popular engines include Jess, Algernon, SweetRules, and Bossam. We chose Bossam (Bossom 2005), a forward-chaining rule engine, as the first rule engine candidate for our semantic course management system since it supports OWL inferencing, it works seamlessly with Java, is well documented, and is very easy to use and configure.

2.6 Application layer

The application layer is composed of two applications: the S-CMS manager and the dynamic enrolment Web site.

S-CMS Manager. We have developed an integrated class project management environment using Eclipse SDK 3.1.1 (Eclipse 2005). Eclipse is an open source framework focused on providing an extensible development platform and application frameworks for building software.

When a teacher interacts with the system, a list of all the courses that he is currently teaching is displayed. This information is retrieved from the RUD knowledge-base. The teacher is then able to create and delete class projects associated with a given course. Figure 6 shows the main screen of the S-CMS application.

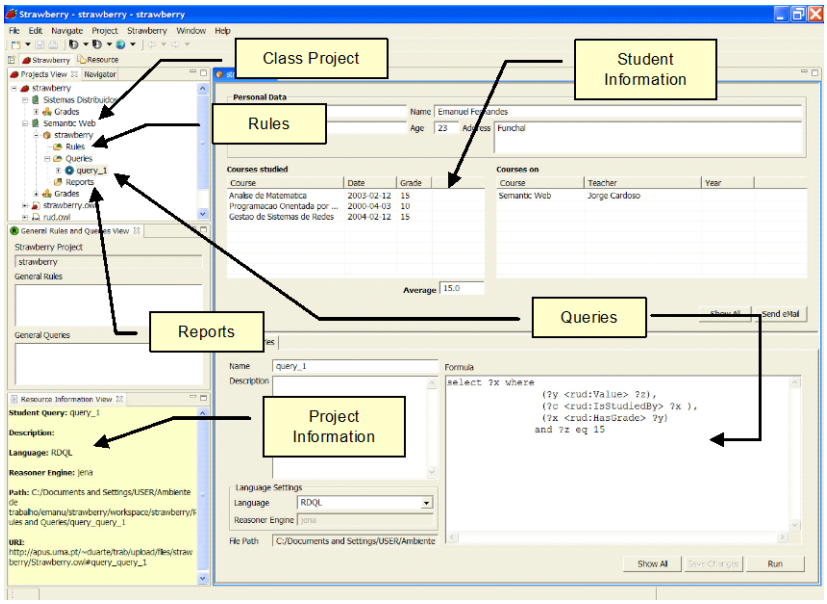

Figure 10-4. Strawberry main application

For each class project created, the teacher is responsible for creating and managing the semantic enrolment rules. The integrated class project management environment has a SWRL and Buchingae rule editor available for this purpose. Semantic enrolment rules can be defined for specific projects (project enrolment rules), for a specific

course (course enrolment rules), or for all the projects independently of the course (global enrolment rules). As explained previously, a teacher can define a project enrolment rule for the project Merging Ontologies Semi-Automatically of the course Semantic Web (CS8050) which states that only students that have taken the Knowledge Engineering course (CS6100) and Logic Programming (CS6550) can enrol.

Dynamic Enrolment Web site. The Enrolment Web site is one of the interfaces for the Strawberry project. It has two main functions. First, it allows for students to enroll into projects proposed by their lecturers and second, it allows the teacher to post reports and other relevant information so that students can easily access it, better enabling the communication process between teachers and students.

A student can enroll in a class project using the S-CMS as a single portal via HTTP/HTML. The S-CMS provides an overview Web page for each class project in which the student can enroll. The Web pages are automatically generated from the S-CMS manager.

Students can be added to a class project either by a bulk upload from the information stored in the RUD, or individually. In the latter case, the student has to indicate the URL of its SUD. The SUD will be read by the S-CMS and matched against a student instance in the RUD. All the information of a student relevant to enrolling for a class project will be retrieved automatically from the RUD. If a student may decide to drop out of a class project he only needs to resend his SUD to the system indicating that he wishes to drop out of a given project.

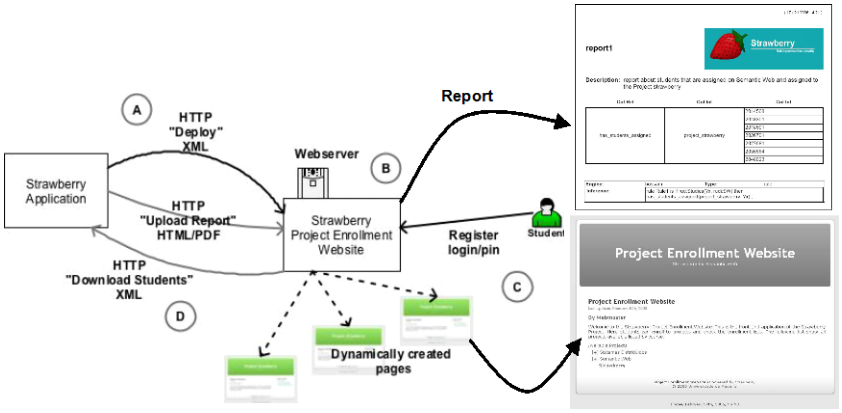

Figure 10-5. General structure of the Dynamic Enrollment Web site

Figure 7 illustrates the general structure of the Dynamic Enrollment Web site. The Web site reflects the current state of the S-CMS manager. The interaction with the S-CMS manager is the following:

a) Using the S-CMS manager, professors can dynamically deploy a Web site for students' enrollment. Professors do not have do deal with Web pages in any way, all the process is automatic. When a professor selects the deploy option, an HTTP connection is established with the Web server and an XML configuration file is uploaded. This file contains a listing of all the courses and project that should be shown in the Web site. Each course has an id, a name, a lecturer and a set of projects. Each project has an id, a name, a description, a last update date and a color.
b) The Web site automatically and dynamically creates a set of Web pages to enable students to register for projects. The module that carry out these tasks was build in PHP because it does not require any other software other than a simple HTTP Server with PHP, it allows to create pages physically dynamically, and made possible the use of XML to exchange data with the S-CMS manager.
c) Students register for projects
d) The list of students that have registered for a particular project is downloaded to the S-CMS manager using XML.

Report Generator. Once students have enrolled for class projects, it is helpful for teachers to have a tool to automatically generate a report indicating which students are in fact allowed to be part of a class project for which they want to enroll. Not all the students that send their SUD to enroll in a class project can indeed carry out the project. The decision that determines if a student can actually carry out a specific project is based on the semantic enrolment rules.

At the presentation layer, the teacher is able to generate enrolment reports that indicate which students are allowed to carry out a project and which are not. We support several formats for the reports. We use the Formatting Objects Processor (FOP 2005) to convert the results from applying enrolment rules to PDF, TXT, and HTML. The Formatting Objects Processor (FOP) is an open source Java API that can convert XML data into reports in PDF format, as well as other relevant formats such as TXT, SVG, AWT, MIF, and PS.

The Grading Ontology and its Plug-In. The grading plug-in is a feature of the Strawberry project on which a grading Ontology is used to enable grading of the students that attend a course. The plug-in

enables the user to define a grading policy on which the final grade will be calculated. It allows teachers to create new evaluation items such as exams and calculate students' final grades based on user defined evaluation items weights. The use of ontologies in modelling a grading domain certainly adds a new degree of flexibility and reuse. The way the ontology can be plugged with exiting ones makes it easier to migrate it to adapt to different Universities or Schools.

Any teacher has its own way to grade students. Different forms of evaluation exist even within one course. This implies that an automatic tool which calculates the grades is difficult to make, because there is no single way to calculate the final grade. For example, if a student does not make assignment nº1 than the final grade will be the grade of the exam, otherwise the final grade will de 60% of the exam and 40% of assignment nº1. A vast number of formulas for calculating grade can be applied. Generally rules are in the form of "if … then ... else …".

To tackle this problem, a semantic Web approach is a reasonable approach. The versatility of domain modeling and the reuse of existing domains (such as one describing a university) could make it easy to integrate a generalist grading ontology that could be easily link with existing ones describing a school.

The most interesting aspect of using the semantic Web is using its reasoning capabilities on which a grading model could be defined using rules that could be used to calculate final grade.

Reasoning support in OWL with SWRL is very flexible and would solve the problem of representing and calculating grades. However, SWRL is not easily understood by humans and the final user of the grading plug-in would not be able to understand the rules in order to create new ones and change existing ones. Another problem associated with SWRL is its limited support. There are not many inference engines that could interpret SWRL. Other languages, such as Buchingae (used in Bossom), exist and have full implementation and are even more easily understood by humans. However, they do not guarantee continued support.

To solve the main problem of the complexity of SWRL language, graphical editors for SWRL could be used such as the one used in Protégé. An editor is helpful in integrating SWRL with existing ontologies. Additionally, it makes use of a logical form for representing SWRL which is then mapped into SWRL itself. This is obviously a generic editor and the user would still need to know how SWRL works. Another alternative would be to use a simple

mathematical language, such as the one used in Microsoft Excel. This approach would make sense for heavy mathematical data, which is our case. However, it would be very difficult to make an interpreter to convert between SWRL and a mathematical language.

The Grading Ontology has been developed to represent the domain of an evaluation. It has been made simple so that it could be easily plugged into existing ontology (in our case it has been adapted for our RUD). The first class of the ontology is the evaluation. It represents the domain of the course evaluation. This means that there is one and only one for each course. An evaluation does not need to be connected specifically to a course; it can be adapted so it can use other domain which can be evaluated. The evaluation consists of evaluation items. These represent specific forms of evaluation within an evaluation domain, such as exams and projects for instance. The evaluation items are sub-classed into specify items, there is only one with the grading ontology – SimpleEvaluation. Other items can be used from the existing university ontology. In this case we are using Project as an evaluation item. For each combination of item and student there is one grade which represents the grade that student had in that evaluation item for that course. The structure of the Grading Ontology is illustrated in Figure 2.

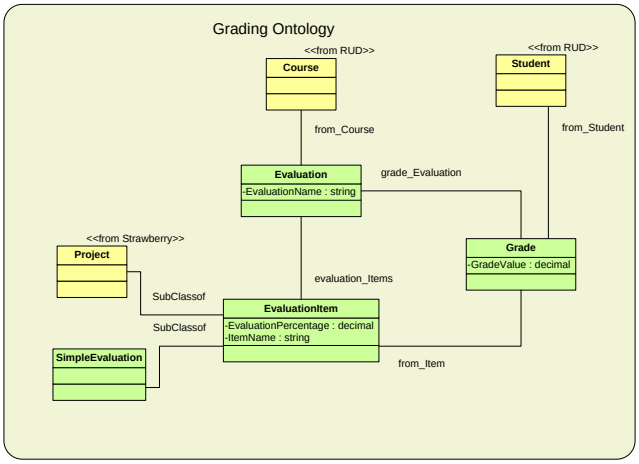

Figure 10-6. The grading Ontology

A simple way to look of the problem of the complexity of editing rules is to look at the problems domains. That is to say to look at specific cases that could be useful to teacher and developing a simple editor to achieve an editor for it. One way to do it is to take part of the

idea of a graphical editor for rules and a mathematical language and joining them. The matching strategy is based in developing a simple editor that generates rules based on a well defined mapping between a set of simple logical conditions and a rule. These logical conditions are based on a triple: (Evaluation Item, Boolean Expressing, Value). For example, the triple (Assignment1, >, 0). Matches contain many of these conditions that if validated the final grade will be calculated based on the weights defined for each item on that match. The can be many matches, one for each evaluation possibility. Figure 4 shows one specific rule that states that …

```
Rulebase CalculateGrade
{
rule Case1 is
Grading:from_Course(?Evaluation,RUD:SemWeb)
  and Grading:grade_Evaluation(?Grade,?Evaluation)
  and Grading:from_Student(?Grade,?Student)
  and
Grading:grade_Value(Grading:Assignment1,?Value)
  and [?Value>0]
 then
  FinalGrade(?Student,RUD:SemWeb,
  Grading:grade_Value(Grading:Project)*0,1+
  Grading:grade_Value(Grading:Exam)*0,9)

rule Case2 is ….
}
```

Retrieve the Evaluation structure from RUD:SemWeb. Retrieve the Grade from the Evaluation structure corresponding to a Student. Retrieve the value (grade) of Assignment1. Check if the value is greater than one, then the final grade is 10% project and 90% exam.

3. EVALUATION

To validate S-CMS we have carried out a benchmark in order to assess the scalability and performance of our architecture under system load. The application was installed at the Department of Mathematics and Engineering, University of Madeira. Our empirical experimentation has involved two machines: a server managing SQL

Server 2000 and a client running S-CMS. Both machines had the same configuration. They were each equipped with Intel P4 3.0 GHz processors, 512 MB main memory, 40GB 7,800 RPM IDE disks, and Microsoft Windows XP home. The computers were connected by a 100Mbit/s Ethernet LAN.

The server was managing the University database that had a size of 123 Mbytes with 200 tables and 600 views. The database included the description of approximately 13 000 students.

The client was running our S-CMS application. Loading the ontologies from the databases toke approximately 7 minutes and 32 seconds. The number of instances created was equal to the number of students in the database, i.e., approximately 13 000 instances. The ontology had a small footprint since we only need to import a subset of the data present in the database. (+-6 mega)

The results obtained are encouraging since loading an ontology from a database is inherently a heavy task. The system performance benchmarking exercise revealed that the proposed solution was able to scale to meet desired throughput and latency requirements.

4. RELATED WORK

There are many tools dealing with course management which have been introduced into universities to redesign teaching in many aspects. These tools include support for teachers (e.g. course delivery and administration) and students (e.g. submissions and involvement). One limitation of the tools available is that they were not developed around the concepts and technologies associated with the semantic Web. As a result, they tend to be static repositories of information for which semantic querying and inferencing on students' data is not possible. Furthermore, they do not tackle the problem of integrating disparate university data sources. For example, MIT OpenCourseWare (OCW) (OCW 2006), WebCT (one of the most widespread commercial course management systems) (WebCT 2006), AIMS (AIMS 2006), Moodle (MOODLE 2006), and BSCW (Basic Support for Collaborative Work) (Klöckner 2000) are educational resource addressing faculty and students. They offer courseware such as syllabi, readings, and quizzes. The information available is mainly static and does not provide features to support querying, inferecing, and data source integration.

Semantics and ontologies have been employed as a common basis for information integration. Ontologies allow for the modeling of the semantic structure of individual information sources, as well describing models of a domain that are independent of any particular information source. Several systems have been developed using this approach. Projects include Carnot (Woelk, Cannata et al. 1993), InfoSleuth (Bayardo, Bohrer et al. 1997), OBSERVER (Mena, Kashyap et al. 1996; Kashyap and Sheth 1998), and COIN (Bressan, Fynn et al. 1997). These projects differ from our work since they do not target a specific domain (i.e. University modeling) and they do not provide solutions to carry out inference on the ontologies created.

5. CONCLUSION

The development of the semantic Web has the potential to revolutionize the World Wide Web and its use. One fundamental aspect that will have a significant impact on the success of the semantic Web will be the ability of the research community to demonstrate the added value of using semantic Web technologies to develop better systems and applications. For the time being, the industry has adopted a "wait-and-see" approach to see how real-world applications can benefit from the semantic Web.

As a contribution to increasing the widespread use of these new technologies, we have developed a real-world application, a Semantic Course Management System (S-CMS), based entirely on semantic Web technologies. S-CMS can semantically integrate and extract heterogeneous data describing university resources, courses, degrees, and students, answer to complex semantic queries, and it is able to carry out reasoning using explicit semantic rules. The system supplies an integrated environment where teachers and students can easily manage class projects. The application presented has been employed successfully to manage student enrolment to class projects at the University of Madeira. Since S-CMS deals with heavily on semantics, the system was used to manage projects from the "Semantic Web" course taught at the Department of Mathematics and Engineering. We believe that S-CMS is also appropriate to support course projects from other departments and that it represents a good step towards the development of real-world semantic applications.

6. QUESTIONS FOR DISCUSSION

Beginner:

1. What typical data sources need to be integrated when developing a CMS?
2. .What is a RUD?
3. .What is a SUD?

Intermediate:

1. What types of information heterogeneity may arise when integrating data sources?
2. What difficulties have been found when in creating and populating the ontology described in this chapter?

Advanced:

1. Make an ontology for representing people of your business or organizations.
2. Build a Buchingae rule that states that only staff members that work on internal and external projects are eligible for travel funding.
3. Write an RDQL that selects the staff members that work in the research department.

7. SUGGESTED ADDITIONAL READING

- Antoniou, G. and van Harmelen, F. *A semantic Web primer*. Cambridge, MA: MIT Press, 2004. 238 pp.: This book is a good introduction to Semantic Web languages.
- Davies, J., Studer, R., and Warren, P. *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons, 2006, 326 pp.: This book provides a comprehensive overview of key semantic technologies. It includes the description of concepts such as knowledge management, ontology generation, and metadata extraction.
- Berners-Lee. T., Fensel, D., Hendler, J., Lieberman, H., Wahlster, W. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. The MIT Press, 2005. 503 pp.: This book covers topics such as software agents, markup languages, and knowledge systems that enable machines to read Web pages and determine their reliability.

8. REFERENCES

AIMS (2006). AIMS: Adaptive Information System for Management of Learning Content. http://www.win.tue.nl/~laroyo/AIMS/.

Bayardo, R. J., W. Bohrer, et al. (1997). InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, New York.

Berners-Lee, T., J. Hendler, et al. (2001). The Semantic Web. Scientific American. **May 2001**.

Bossom (2005). Bossom engine for the semantic Web, http://projects.semwebcentral.org/projects/bossam/.

Bressan, S., K. Fynn, et al. (1997). The COntext INterchange Mediator Prototype. ACM SIGMOD International Conference on Management of Data, Tucson, Arizona.

Cardoso, J. (2004). Issues of Dynamic Travel Packaging using Web Process Technology. International Conference e-Commerce 2004, Lisbon, Portugal.

Cardoso, J., J. Miller, et al. (2005). Academic and Industrial Research: Do their Approaches Differ in Adding Semantics to Web Services. Semantic Web Process: powering next generation of processes with Semantics and Web services. J. Cardoso and S. Amit. Heidelberg, Germany, Springer-Verlag. **3387:** 14-21.

Eclipse (2005). Eclipse open source community, http://www.eclipse.org/.

FOP (2005). FOP (Formatting Objects Processor), http://xmlgraphics.apache.org/fop/. **2005**.

Ian Horrocks, Peter F. Patel-Schneider, et al. (2003). SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.daml.org/2003/11/swrl/.

Jasper, R. and M. Uschold (1999). A framework for understanding and classifying ontology applications. IJCAI99 Workshop on Ontologies and Problem-Solving Methods.

Jena (2005). Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/,.

Karvounarakis, G., S. Alexaki, et al. (2002). RQL: a declarative query language for RDF. Eleventh International World Wide Web Conference, Honolulu, Hawaii, USA.

Kashyap, V. and A. Sheth (1998). Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies, Academic Press.

Klöckner, K. (2000). BSCW - Educational Servers and Services on the WWW, Adelaide.

Knublauch, H., R. W. Fergerson, et al. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. Third International Semantic Web Conference (ISWC 2004), Hiroshima, Japan.

Kumar, A. and B. Smith (2004). On Controlled Vocabularies in Bioinformatics: A Case Study in Gene Ontology. Drug Discovery Today: BIOSILICO. **2:** 246-252.

Lassila, O. and D. McGuinness (2001). "The Role of Frame-Based Representation on the Semantic Web." Linköping Electronic Articles in Computer and Information Science **6**(5).

Lawrence, R. and K. Barker (2001). Integrating Data Sources Using a Standardized Global Dictionary. Knowledge Discovery for Business Information Systems. J. M. Zurada, Kluwer Academic Publishers: 153-172.

Mandal, C., V. L. Sinha, et al. (2004). "Web-based Course management and Web Services." Electronic Journal of e-Learning **2**(1): 135-144.

Meinel, C., H. Sack, et al. (2002). Course management in the twinkle of an eye - LCMS: a professional course management system. Proceedings of the 30th annual ACM SIGUCCS conference on User services, Providence, Rhode Island, USA, ACM Press.

Mena, E., V. Kashyap, et al. (1996). OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. Conference on Cooperative Information Systems, Brussels, Belgium, IEEE Computer Society Press.

MOODLE (2006). Modular Object-Oriented Dynamic Learning Environment (moodle), http://moodle.org/.

OCW (2006). OpenCourseWare. http://ocw.mit.edu/index.html, MIT.

Ouskel, A. M. and A. Sheth (1999). "Semantic Interoperability in Global Information Systems. A brief Introduction to the Research Area and the Special Section." SIGMOD Record **28**(1): 5-12.

OWL (2004). OWL Web Ontology Language Reference, W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/owl-ref/. **2004**.

OWL-S (2004). OWL-based Web Service Ontology. **2004**.

RDQL (2005). Jena RDQL, http://jena.sourceforge.net/RDQL/.

Roure, D., N. Jennings, et al. (2001). Research Agenda for the Future Semantic Grid: A Future e-Science Infrastructure http://www.semanticgrid.org/v1.9/semgrid.pdf.

RSS (2005). RSS 2.0 Specification, http://blogs.law.harvard.edu/tech/rss.

Sheth, A. (1998). Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. Interoperating Geographic Information Systems. M. F. Goodchild, M. J. Egenhofer, R. Fegeas and C. A. Kottman, Kluwer, Academic Publishers: 5-30.

Sheth, A. P. (1999). Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. Interoperating Geographic Information Systems. C. A. Kottman, Kluwer Academic Publisher: 5-29.

Shum, S. B., E. Motta, et al. (2000). "ScholOnto: an ontology-based digital library server for research documents and discourse." International Journal on Digital Libraries **3**(3): 237-248.

Swoogle (2005). Search and Metadata for the Semantic Web - http://swoogle.umbc.edu/.

WebCT (2006). http://www.webct.com/.

Woelk, D., P. Cannata, et al. (1993). Using Carnot for enterprise information integration. Second International Conference on Parallel and Distributed Information Systems.