Deploying a Semantically-Enabled Content Management System in a State University

Maria Befa[1], Efstratios Kontopoulos[1], Nick Bassiliades[1], Christos Berberidis[1,2], Ioannis Vlahavas[1,2]

[1] Department of Informatics, Aristotle University of Thessaloniki
GR-54124, Thessaloniki, Greece
{mmpefa, skontopo, nbassili, berber, vlahavas}@csd.auth.gr
[2] School of Science & Technology, International Hellenic University
GR-57001, Thessaloniki, Greece

Abstract. Public institutes often face the challenge of managing vast volumes of administrative documents, a need that is often met via Content Management Systems (CMSs). CMSs offer various advantages, like separation of data structure from presentation and variety in user roles, but also present certain disadvantages, like inefficient keyword-based search facilities. The new generation of content management solutions imports the notion of semantics and is based on Semantic Web technologies, such as metadata and ontologies. The benefits include semantic interoperability, competitive advantages and dramatic cost reduction. In this paper a leading Enterprise CMS is extended with semantic capabilities for automatically importing and exporting ontologies. This functionality enables reuse of repository content, semantically-enabled search and interoperability with third-party applications. The extended system is deployed in semantically managing the large volumes of documents for a state university.

Keywords: semantic web, metadata, content management, ontology.

1. Introduction

Public institutes, like state universities, often face the challenge of managing vast volumes of (mainly administrative) documents, which are rapidly increasing every day. The need to manage massive volumes of content imposes the deployment of *content management systems* (*CMS*), software applications that allow creating, editing, managing and publishing content, as well as controlling access and versioning [1]. CMSs mainly offer the advantage of separating data structure from presentation, defining user roles, in order to control what (subset of) information each user has access to.

Nevertheless, besides simply managing content, a key functionality is also the ability to efficiently and effectively retrieve information. Paradoxically, most current CMSs offer only basic keyword-based search, similarly to Web search engines [2]. Although several techniques (word occurrence frequency measurement, lexical analysis etc.) have been developed for improving the retrieved results, they have not been adequately evolved yet, in order to offer more advanced features than basic search.

A lot of effort is now concentrated on developing a new generation of content management solutions for efficiently managing the rapidly growing volumes of information, based on the notion of *semantics* (study of meaning). The use of semantic technologies in content management (referred to as "*Semantic Content Management*") will bring various benefits to organizations working with large volumes of data, like syntactic/semantic interoperability, dramatic cost reductions, revenue improvements and opportunities for competitive advantages. Semantic Content Management methods are based on the *Semantic Web* (SW), an evolving effort towards extending Web content with semantic information (metadata) [4]. A key factor in the development of the Semantic Web are *ontologies* – data models that represent a given domain and are used to reason about the objects in that domain and the relations between them [5]. *RDF Schema* (*RDF/S*) is an ontology language for encoding ontologies [6], providing a vocabulary for describing properties and classes of RDF resources as well as hierarchies among them. Overall, the Semantic Web provides a set of standards for encoding and querying knowledge, so that data integration is facilitated. Therefore, Semantic Web technologies seem promising for improving CMS data handling capabilities and for leading to advantages in all situations, where data integration is beneficial.

Especially in *e-Government* (*e-Gov*) that involves a wide array of IT-related aspects, like human-computer interaction (HCI), software/systems engineering and Web service design, the SW can resolve many HCI- and access-related issues and allow users to facilitate e-Gov processes [7]. Additionally, e-Gov networks are among the largest networks in existence, according to size, number of users and information providers; thus, creating a Semantic Web infrastructure to meaningfully organize e-Gov networks is highly desirable. At the same time, the complexity of the existing e-Gov implementations also challenges the feasibility of Semantic Web development [8].

In this paper we are extending a leading Enterprise CMS, called *Alfresco* [9], with semantic capabilities and, more specifically, the capacity of automatically importing and exporting RDF/S ontologies. The implemented Alfresco plug-in parses the input ontologies and automatically updates the corresponding Alfresco core files that are related with the creation of categories, properties, metadata and their appearance in the Alfresco front-end. The reverse process, namely, exporting content from the Alfresco repository into an RDF/S ontology, is also integrated in the plug-in, allowing third-party applications to use and process Alfresco content. In order to test the implemented extensions, the system was deployed for managing the large volumes of documents for a state university (as in e.g. [10]), also offering semantic interoperability between the university and other education-related organizations.

The rest of this paper is structured as follows: Section 2 gives a brief overview of related work paradigms, followed by a more thorough presentation of the Alfresco CMS. Section 4 comprises the backbone of this work, presenting the test application domain and the proposed semantic extensions implemented in Alfresco, while the paper is concluded with the final remarks and directions for future work.

2. Related Work

SCORE (*Semantic Content Organization and Retrieval Engine*) [11] is a system that

offers semantic content organization and recuperation of information. The software has the capability of processing information, by interrelating different pieces of information existing in its repository with the knowledge that a person can possess over a specific subject. These semantics-based information processing capabilities offer a significantly superior performance over syntactic information searching methods. Other vital advantages of SCORE include its high output and extensibility.

Caravela [12] is another CMS that provides dynamic unification and automatic categorization and is mainly used for categorizing published scientific researches and for the on-line creation of bibliographies on issues related to schema evolution and data cleaning. Content and documents of various types from diverse (semi-structured) sources can be unified and categorized in different dimensions. The system offers: (a) dynamic information unification with simultaneous attachment of related sources, (b) automatic interconnection that offers dynamic categorization without any time- or effort-related cost to the user, (c) semi-automatic categorization at length of multiple dimensions, and, (d) flexible and functional points that make search tasks easier.

The *Rhizomer* platform [13] is a CMS based on REST and Semantic Web technologies, offering advanced content management services. All content is described using semantic metadata that is semi-automatically extracted from multimedia content, facilitating publishing, querying, browsing, editing and interacting with semantic data. Rhizomer supports search via semantic queries and the user can navigate through the graph of data, retrieving fragments and rendering them as interactive HTML. Additionally, two or more pieces of metadata about common or similar resources can also be mixed and simple mappings between concepts from different ontologies can be defined. Finally, the user can annotate new semantic metadata describing a resource, or edit existing annotations via user-friendly HTML forms.

A more e-Gov-oriented approach is featured in [14], where a system for multimedia management is proposed. The system supports: automatic information extraction, retrieval, semantic interpretation of the relevant information presented in the document, storing and long term preservation. Since an e-Gov document is typically composed by various multimedia data types (images, text, graphic objects, audio, video etc.), the textual processing phase involves the use of legal ontologies.

The systems described above are semantic-enabled CMSs with possibly more related capabilities than the "semantically enhanced" Alfresco presented here. However, the aim of this work is to convert a "traditional" CMS into a semantically-enabled system, offering the capability of interoperation with other applications via importing/exporting ontologies, which is a feature none of the other systems possesses.

3. Alfresco

Alfresco [9] is an open-source enterprise CMS and is considered a pioneer in its domain. It comprises a client/server application that provides a powerful *Application Programming Interface* (*API*) for web services integration. The server-side of the software resides inside an application server, where the Alfresco application and repository are kept. On the other hand, the client-side does not require installation of additional packages, as it is launched by the user's web browser and runs like a typi-

cal web application. This modular architecture provides the advantage that existing components can be replaced, providing better implementation and optimum integration with existing environments/applications. Additionally, unneeded features can be removed, offering a lighter and potentially more flexible version of the system.

Alfresco integrates a variety of cutting-edge open-source technologies and can cooperate with any popular data base repository. Furthermore, Alfresco is fully compatible with *XML*, supporting importation and exportation of content in any XML format (e.g. *Dublin Core*). Moreover, the system is strongly supported by a wide developer community. In order to encourage further development of the system, the Alfresco Company consistently offers the latest Alfresco versions openly to the community and has translated the software in more than 20 languages.

The two most important factors for choosing Alfresco as the development platform were its distributed architecture and open source code. The former is based on a scalable *Service-Oriented Architecture* (*SOA*), while the latter allows for incorporation of further technologies into the system as well as the extension and interoperability with external applications. Additional advantages involve Alfresco being easy to install and use, without requiring any additional third-party applications and its support by well-written documentation as well as an enthusiastic developer community.

4. Implementation: Alfresco Semantic Extensions

A running example will be used throughout section 4, for illustrating the usability of the presented implementation. The *International Hellenic University* (*IHU*) was chosen as the application domain, since it is a newly founded state organization that inevitably faces the challenge of managing large volumes of administrative documents and is currently developing its ICT infrastructure. In addition to improving content management and information search and retrieval, the deployment of a semantically-enabled CMS in organizing IHU will also contribute in the *semantic interoperability* between the institute and other organizations (e.g. Ministry of Education, other universities, international education-related organizations etc.).

4.1. Source Ontology Specifications

The Alfresco extension is designed to accept as input all RDF/S ontologies that comply with the following specifications:

- The input ontology can contain any number of classes, subclasses and properties.
- Only data type properties are allowed in the ontology, since Alfresco cannot handle object properties.
- Each class should have a label that will enclose a human-readable identifier for the class. The tag content is used for representing the class in the Alfresco front-end.
- Hierarchy correlations among classes and subclasses are allowed, but their property counterparts (properties/subproperties) are not, as Alfresco does not support them.

As a sample model, an RDF/S ontology was deployed that was used for testing the Alfresco semantic extensions and represents the organizational chart of IHU, contain-

ing 25 classes and 20. The root class in the hierarchy is "*University Branch*" that declares a categorization of the various operational branches of IHU and has four subclasses, "*Administrative Board*", "*Administration*", "*Secretary*", "*Department*", which are further subdivided.

4.2. Importing Ontologies into Alfresco

Initially, the *Ontology Loader* module reads the input RDF/S ontology that conforms to the above specifications and detects all included classes, subclasses and properties, as well as their interrelations. For accomplishing these tasks, the software deploys *the Jena Semantic Web Framework* [15] and the *Pellet* reasoner [16]. Since Alfresco treats classes (represented as "*categories*" in Alfresco) and properties (represented as "*aspects*") separately, the module also follows this principle, generating two separate vectors that include classes (vector V_C) and properties (vector V_P), respectively. Each vector member "is aware" of all relevant information, e.g. a member class $c_i \in V_C$ is aware of all its predecessor and descendant classes, while a member property $p_j \in V_P$ is aware of its domain and range. Eventually, the Ontology Loader passes vector V_C to the *Categories Manager* module and vector V_P to the *Aspects Manager* module, which are the components responsible for updating repository content.

The *Categories Manager* module modifies file *categories.xml* (a core Alfresco XML file responsible for classifying repository content), adding the categories/classes that were extracted from the ontology (vector V_C). The new categories are appended, starting with the root superclass (i.e. the class that has no further superclasses) and gradually proceeding to its subclasses etc. In case of categories with more than one parent category, the corresponding child category is appended more than once into the document. The process terminates, when all members of vector V_C have been added.

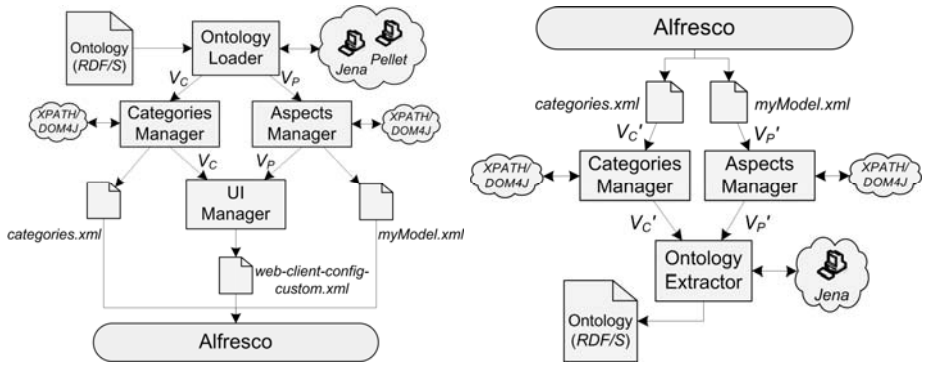

Fig. 1. Ontology import plug-in architecture (left); Ontology export plug-in architecture (right).

The *Aspects Manager* module, on the other hand, is responsible for adding the extracted properties (vector V_P) into *myModel.xml* (another core file responsible for properties, rules and metadata). For every class that appears as domain in (at least)

one of the detected properties, a new aspect with the same name is created. The range of the property determines the type of the aspect and the process terminates, when all members of vector V_P have been added.

Finally, the *UI Manager* module is responsible for modifying file *web-client-config-custom.xml* that is responsible for displaying the aspects and categories in the Alfresco web client front-end. The module receives the two vectors V_C and V_P from the previous two modules and proceeds to three basic modifications to the file. Firstly, the classes and their properties are declared. Secondly, the module has to define which classes should appear in the Alfresco wizard for creating rules on the content. The third modification involves the aspects that will be included in the Alfresco search window. Fig. 1 (left) displays the architecture of the implemented plug-in responsible for importing an ontology into Alfresco.

4.3. Exporting Ontologies from Alfresco

The process of exporting ontologies from the repository was integrated as a second plug-in to the system, which implements a similar but functionally inverse approach. Again, the two Alfresco core files that play a major role are *categories.xml* and *myModel.xml*, the former provides the set of RDF/S classes, while the latter is responsible for generating the corresponding set of properties. This process is realized via two of the modules presented previously: *Categories Manager* and *Aspects Manager*.

The *Categories Manager* module initially parses the Alfresco file *categories.xml*, creating a vector of categories/classes V_C'. Each member category $c_i' \in V_C'$ is aware of its direct predecessor, which will be included as the corresponding superclass in the resulting RDF/S ontology. This procedure excludes any root classes, which are considered subclasses of `rdfs:Resource`, the top RDF/S class. The *Aspects Manager* module, on the other hand, parses file *myModel.xml*, searching for properties. A corresponding property vector V_P' is created that contains properties; each member property $p_j' \in V_P'$ is characterized by its name, the property range and the property domain.

The two vectors (V_C' and V_P') are then passed to the *Ontology Extractor* module that actually creates the resulting RDF/S ontology through Jena. Fig. 1 (right) displays the architecture of the ontology export plug-in.

4.4. Integrating into Alfresco

After implementing the two plug-ins, the final task was to integrate the application into Alfresco. This process involves modifying the following Alfresco files: (a) *faces-config-custom.xml*, where the JSF-managed listener beans reside, (b) *web-client-config-custom.xml*, which is responsible for customizations to the Alfresco front-end, and (c) *titlebar.jsp* that contains the statement declarations for the newly developed front-end buttons that will appear on the Alfresco GUI toolbar.

Two frames were developed for assisting the user in importing and exporting an ontology, respectively. The *Ontology Import Frame* (*OIF*) (Fig. 2 - left) requires from the user to enter the path of the input ontology as well as the locations of the three Al-

fresco core files (*categories.xml*, *myModel.xml*, *web-client-config-custom.xml*), as described previously. The plug-in automatically detects the locations of the respective Alfresco files, but the user also has the option of browsing for a different location. Next to each text field in *OIF* there is an "*Import*" button, for modifying each of the three files separately, but there also exists a choice for mass modification ("*Import All*"), through which the user can merge all three ontology importing steps into one.

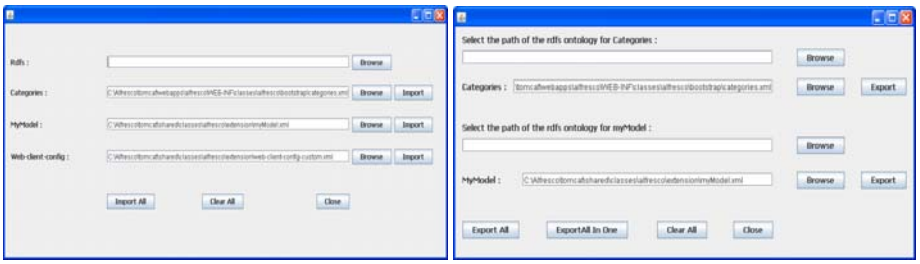

Fig. 2. Developed frames for importing (left) and exporting (right) an ontology.

Similarly to *OIF*, the *Ontology Export Frame* (*OEF*) (Fig. 2 - right) consists of text fields, where the user again has to provide file paths. *OEF* operates in two modes: the user can either export the repository content into two separate ontology files (one for classes and one for properties), or, alternatively, a merged export can be performed, exporting the content into one single file. This dual functionality was considered useful, since Alfresco stores representations for classes/categories and properties/aspects in different locations, without ensuring that consistency is maintained. Buttons "*Export All*" and "*Export All In One*" perform these the two distinct operations.

5. Conclusions and Future Work

This paper argued that CMSs offer various advantages to enterprises and organizations for managing the large volumes of information, but also display certain disadvantages. The application of Semantic Web technologies in content management offers various benefits, like semantic interoperability, dramatic cost reductions, revenue improvements and competitive advantages. This paper presented a semantic extension to a leading Enterprise CMS, called *Alfresco* for automatically importing and exporting RDF/S ontologies. The added functionalities enable re-use of Alfresco repository content, semantically-enabled search and interoperability by a variety of other third-party applications. IHU was chosen as the test domain for the implemented system, which was used for managing the subset of managerial and administrative documents.

As for future work directions, an interesting improvement would be to have the system import and export ontologies encoded in a more expressive language, like OWL. Another appealing extension would be the integration of a reasoning engine, which would allow Alfresco to automatically classify content classification. Further operations that Alfresco can cover can be discovered by its application in different environments, such as educational institutes, enterprises and, generally, any organiza-

tion that deals with voluminous collections of documents. Finally, it would be interesting to study whether the "semantic-oriented" methodologies developed for Alfresco can also be applied in other "traditional" CMSs, like e.g. *Plone* [17].

References

[1] Auffret, M.: Content Management Makes Sense. Journal of Knowledge Management Practice, Vol. 2, December (2001)

[2] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Web Semantics: Science, Services and Agents on the World Wide Web, 4(1):14-28 (2006)

[3] Lytras, M., Garcia, R.: Semantic Web Applications: A Framework for Industry and Business Exploitation-What is needed for the Adoption of the Semantic Web from the Market and Industry. Int. Journal of Knowledge and Learning, 4(1):93-108 (2008)

[4] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, 284(5):34-43 (2001)

[5] Guarino, N.: Formal Ontology, Conceptual Analysis and Knowledge Representation. Int. J. Hum.-Comput. Stud. 43(5-6):625-640 (1995)

[6] Brickley, D., Guha, R. V.: RDF Vocabulary Description Language 1.0: RDF Schema. `http://www.w3.org/TR/rdf-schema/` (2004)

[7] Fass, L. F.: The Semantic Web, E-Government and the Digital Divide. In: Abecker, A., A. Sheth, G. Mentzas and L. Stojanovic (Eds.), Semantic Web Meets eGovernment, Stanford University, pp. 14-17, AAAI Press SS-06-06 (2006)

[8] Wagner, C. Cheung, K., Ip, R., Bottcher, S.: Building Semantic Webs for e-government with Wiki technology. Electronic Government, 3(1):36-55, Inderscience (2006)

[9] Alfresco Enterprise Content Management System, `http://www.alfresco.com/`

[10] Wales, T.: Library Subject Guides: a Content Management Case Study at the Open University, UK. Program-Electronic Library and Information Systems, 39(2):112−121 (2005)

[11] Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., Warke, Y.: Semantic Content Management for Enterprises and the Web. IEEE Internet Computing, 6(4):80-87 (2002)

[12] Aumueller, D., Rahm, E.: Caravela: Semantic Content Management with Automatic Information Integration and Categorization. Franconi, E., Kifer, M., May, W. (Eds.): ESWC 2007, LNCS 4519, pp. 729-738 (2007)

[13] García, R., Gimeno, J. M., Perdrix, F., Gil, R., Oliva, M.: The Rhizomer Semantic Content Management System. In: Lytras, M. D., Carroll, J. M., Damiani, E., Tennyson, R. D. (Eds.) 1st World Summit on the Knowledge Society: Emerging Technologies and Information Systems for the Knowledge Society, LNAI, 5288, pp. 385-394, Springer (2008)

[14] Amato, F., Mazzeo, A., Moscato, V., Picariello, A.: Semantic Management of Multimedia Documents for E-Government Activity. In: Int. Conference on Complex, Intelligent and Software Intensive Systems, pp. 1193-1198 (2009)

[15] McBride, B.: Jena: Implementing the RDF Model and Syntax Specification. Technical report (2001)

[16] Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Web Semantics: Science, Services and Agents on the WWW, 5(2):51-53 (2007)

[17] Clark, A., Parker, C., Hanning, D., Convent, C., DeStefano, J., Stahl, J., Aspeli, M., Bowen, M., Newbery, R., Knox, S., McMahon, S., Conklin, T., Williams, V.: Practical Plone 3: A Beginner's Guide to Building Powerful Websites. Packt Publishing (2009)