DOCUMENT MANAGEMENT FOR COLLABORATIVE E-BUSINESS: INTEGRATING EBXML ENVIRONMENT AND LEGACY DMS

Alessio Bechini, Andrea Tomasi and Jacopo Viotto
Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni
University of Pisa, Via Diotisalvi 2, 56122 Pisa, Italy

Keywords: Document Management Systems, ebXML, Interoperability, Web Services.

Abstract: It is widely known that e-business capabilities are a key requirement for a large number of modern enterprises. New B2B technologies can enable companies all around the world to collaborate in more effective and efficient ways, better satisfying the needs of their customers. The ebXML specifications are the standard solution to achieve this kind of interoperability, but they are not as widespread as traditional legacy systems yet. Enterprises typically store their knowledge inside Document Management Systems (DMS), which come in different technologies and handle system-specific metadata models. In this paper, we propose an architecture that enables enterprises to take advantage of the power and flexibility of the ebXML approach to metadata management; this is achieved without affecting in-place working systems, and with no need for a complete repository reconstruction.

1 EBXML AS A SUPPORT FOR BUSINESS INTERACTIONS

ebXML (Electronic Business using eXtensible Markup Language) is a suite of XML-based specifications emerging as the de-facto standard for e-business. It provides "a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes" (ebXML). For example, ebXML can be proficiently used as the principal building block within information systems to support supply chain traceability (Bechini et al., 2005 and 2007).

The ebXML Registry specifications (ebRIM, ebRS) play a crucial role within the standard. An ebXML Registry is defined as "an information system that securely manages any content type and the standardized metadata that describes it", thus enabling the sharing of content and related metadata across organizational entities. Being theFmited number of companies natively store documents and enterprise knowledge according to the ebXML standard, since it is a fairly recent technology; the race for e-business capability has hampered the adoption of one acknowledged standard solution for document management, thus yielding significant interoperability problems. Much (if not all) of the generated information is present inside one or more traditional Document Management Systems (DMS), each implemented upon a different technology and following a proprietary metadata model.

Ideally, interoperability could be achieved moving all enterprise knowledge into an ebXML Registry. In practice, this is hardly ever feasible, due to the strong bindings between the DMS and the rest of the company information system. We propose an architecture to boost up the functionality of in-place DMSs, taking advantage of the power and flexibility of ebXML. The original DMSs, left unchanged, are coupled with an ebXML Registry, used to mirror their metadata. All metadata-related operations (especially searches) can thus be enabled to leverage the ebXML Registry, overcoming the typical restrictions of the back-end legacy module. A direct access to the original DMS is performed only in case an actual document would be involved in the query. Whenever required by a specific application within the upper information system, an additional distinct component can be asked to coordinate the access to the underlying systems, and enforce metadata consistency.

Bechini A., Tomasi A. and Viotto J. (2007).
DOCUMENT MANAGEMENT FOR COLLABORATIVE E-BUSINESS: INTEGRATING EBXML ENVIRONMENT AND LEGACY DMS.
In *Proceedings of the Second International Conference on e-Business*, pages 78-83
DOI: 10.5220/0002112700780083

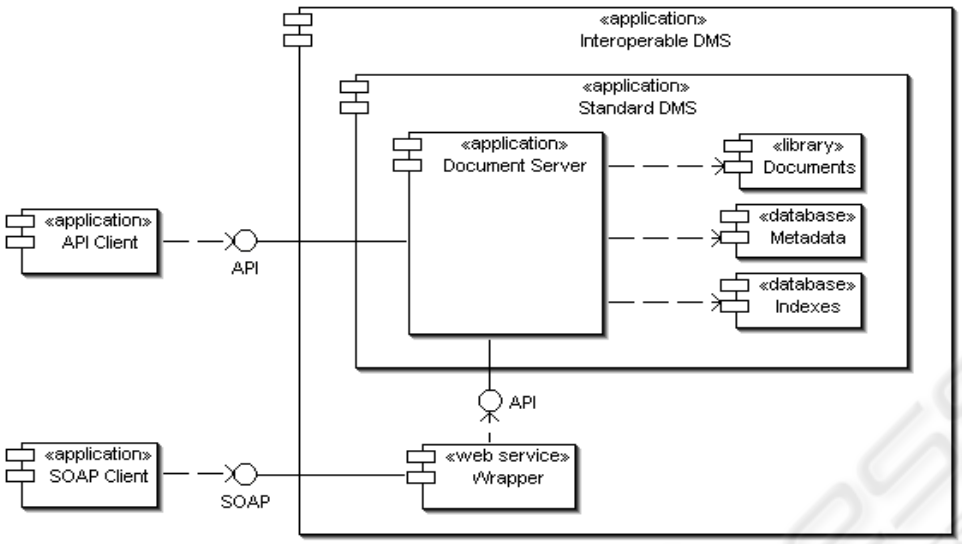

Figure 1: Interoperable DMS.

2 INTEROPERABLE DMS

A common requirement for a DMS is the ability to easily integrate with external systems, in order to provide access to enterprise knowledge from a wide variety of platforms. In spite of this, DMSs typically support a small number of applications, and little or no effort is made towards generalized interoperability. As a partial solution, some systems provide APIs to enable administrators to code adapter applications.

Recently, the adoption of Service Oriented Architecture (SOA) contributed to modify this scenario: the latest versions of the most popular DMSs provide *support* for Web services to ease system integration and extension (FileNet, Documentum, Vignette). Web services run on the server side and wrap API calls to offer system access by means of standard Web technologies. Unfortunately, the claimed *support* often simply relates to a framework to help develop custom services: administrators still need to study the system's details and write the service code accordingly. This is a tedious and error-prone process, and it should be avoided as much as possible. Moreover, no standard way is defined for the implementation of such Web services, hence third-party software need to comply with different interfaces, depending on the actual DMS in use. SOA support in DMSs should definitely be improved, in order to keep up with the current technology trend: "by 2010, 80 percent of application software revenue growth, including licenses and subscription fees, will come from products based on SOA." (Gartner, 2005).

2.1 A Proposal

The proposed solution can be roughly described as a Web service extension to integrate DMS interfaces. The overall system basically takes advantage of DMS-specific APIs, combines them into logically distinct functions, and exposes them as Web services. In this perspective, each specific DMS requires its own specific wrapper software. Figure 1 shows the architecture of a traditional DMS, composed with our extension to obtain an interoperable DMS. As it can be seen, our system acts as an *additional* access point to the DMS, leaving the original system intact. This kind of enrichment in the access point number let clients free to keep on working through the native interface, whenever it is either mandatory or convenient. Clients using the new interface, instead, will be able to communicate with the DMS via SOAP messages, regardless of technology issues and implementation details.

2.2 DMS Interface for Interoperability

In order to achieve true independence from the actual DMS in use, we need to set up a general-purpose interface, able to accommodate typical needs for document management systems.

Table 1: Functions overview. The Profile structure represents a minimal subset of supported metadata (system specific). The ProfileField structure represents a name/value pair for arbitrary metadata fields.

Function	Description
`public string doLogin(string library, string user, string password)`	Login with the given user/password pair.
`public Profile[] search(string dst, string lib, string docnum, string author, string name, string type, string[] words)`	Perform full-text and metadata search. The *advanced* version allows a variable number of name/value pairs in input (the ProfileField structure array). Search criteria can be freely mixed.
`public Profile[] advancedSearch(string dst, string lib, string form, ProfileField[] sc, string[] words, string[] rp, ProfileField[] ord)`	
`public byte[] getDocument(string dst, string lib, string docnum, string ver, string subver)`	Document check-out.
`public string putDocument(string dst, string lib, string docname, string author, string typist, string type, string app, byte[] data)`	Document check-in. The first two functions create a new document. This implies the creation and filling of a new metadata entry, together with actual file transfer. The third function creates a new version, filling the comment and author fields in version metadata (almost a standard). The last one replaces an existing version.
`public string advancedPutDocument(string dst, string lib, string form, ProfileField[] fields, byte[] data)`	
`public void putVersion(string dst, string lib, string docnum, string ver, string author, string typist, string comment, byte[] data)`	
`public void replaceVersion(string dst, string lib, string docnum, string ver, string subver, byte[] data)`	
`public void deleteDocument(string dst, string lib, string docnum, bool delProf)`	Document deletion (including all versions) and version deletion (including all subversions).
`public void deleteVersion(string dst, string lib, string docnum, string ver, string subver)`	
`public void updateProfile(string dst, string lib, string docnum, string docname, string author, string typist, string type, string app)`	Metadata editing, and *advanced* version.
`public void advancedUpdateProfile(string dst, string lib, string form, string docnum, ProfileField[] fields)`	

Unfortunately, no standard interface of this kind has been clearly stated so far, and no standard access method to the DMS document base still exists, despite the fact that typical operations performed over this kind of systems are fairly general. Therefore, our prime concern is to outline a set of core operations that every DMS is required to support, and then standardize the way they are accessed.

We restricted our choice to fundamental services, leaving out any platform-specific feature. This is an explicit design choice: most applications interface to a DMS in terms of simple queries (mainly document upload/download and metadata or content-based searches), whereas advanced features are rarely used and are not guaranteed to be available in all environments. We didn't take into account administration functions, such as user creation, role definition and so on, due to the absence of a standard treatment for access control and user management. However, this might be a significant functionality to add in future versions.

We finally came out with the following list; since these operations are at the heart of document management itself, they are the most commonly used by end-users and the most widely supported by existing systems. Table 1 shows the corresponding function signatures in our test implementation.

Authentication - Obviously, some sort of authentication is needed to access the system. The simplest and most common way to identify a user is asking for a username/password pair. If successful, the login function returns a unique session identifier, which will be needed for every subsequent operation; this identifier encodes user rights and roles, thus allowing access control.

Document/version creation - Creating a brand new document implies uploading the file and creating (and filling) a new entry inside the database to store its metadata. The creation of a new version for an existing document is almost the same process, but slightly more information is needed to identify parent document and version number.

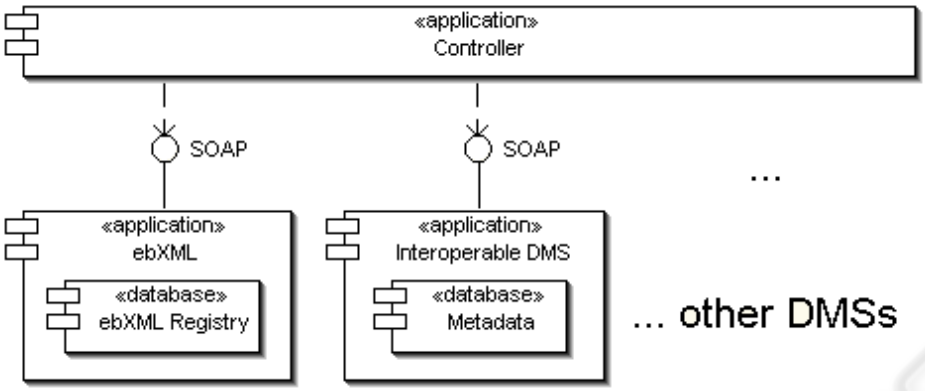

Figure 2: Overall architecture; an ebXML Registry and several legacy DMSs connected to a controller application.

Document/version editing - Since file editing can't be performed in-place, it requires document download (check-out), local editing, and upload of the modified copy (check-in); this typically leads to the creation of a new version, but version replacement is possible as well. Metadata may be directly edited with ad-hoc functions, but it most often changes as part of a file updating process.

Document/version deletion - Deleting one version determines the erasure of all its subversions; deleting an entire document results in the erasure of all its versions. In either case, the deletion includes both physical files and database entries.

Metadata/fulltext search - Users rarely need a specific document, and hardly ever know the exact id; instead, they often look for documents talking about some topic, or containing a few given words. Hence, DMSs support searches over both metadata and file content.

3 SYSTEM ARCHITECTURE

According to our architecture, newly installed and in-place components are arranged in three sub-systems (Figure 2):

- An *interoperable* DMS, containing both documents and related metadata. It is entirely based upon the DMS originally in use inside the enterprise, with the added value of our interoperability component. In the general case, there could be many different systems, each with multiple instances.
- An ebXML Registry, used to store a copy of DMS metadata in an ebXML-compliant fashion. Exploiting its capabilities, we can manage legacy metadata using the same advanced features available for documents natively stored in ebXML repositories.
- A *controller* application intended to coordinate access to the above-mentioned systems. Access to DMSs is performed through the interface we described, while the ebXML Registry already offers standard Web services.

Since there is no direct link between the ebXML Registry and enterprise repositories, every interaction is mediated by the controller. This is necessary to maintain the independence of each individual component: as far as the single sub-system is concerned, no knowledge about the external world is required. It is up to the controller to compose the simple interaction functions provided by each interface into a globally meaningful and consistent operation. In particular, such software level should provide also a semantic mapping among different metadata items on distinct DMS, or maybe a mapping towards some kind of commonly established ontology.

4 METADATA MANAGEMENT

Metadata play a crucial role in the exploitation of functionalities exposed by a typical DMS. In order to be properly managed and to efficiently contribute to information delivery goals, each document must be properly characterized by specifying its coordinates within an adequately rich metadata space. In the archive management and digital libraries community, a standardization effort has led to the definition of a basic set of metadata to be dealt with for each stored/referenced document (Dublin Core); moreover, communication protocols for metadata harvesting have been built up within the

Open Archive Initiative (OAI), taking into account Dublin Core metadata set. Despite this standardization attempt, Dublin Core has not been adopted by the large majority of DMSs, mainly due to its focus on informative documents, instead of business-related ones. It is worth noticing that problems about semantic mapping among different metadata items (or towards a commonly established ontology) arise also in related application fields, e.g. cultural heritage digital libraries (Bechini et al., 2004) and niches search engines (Petinot et al., 2003): in these contexts, it is often reasonable to employ a mediator scheme approach, possibly referring to Dublin Core as a common metadata set.

In the framework of a SOA application, each DMS can be regarded as a completely autonomous entity, and no assumption can be made on its own capabilities in supporting any kind of standard metadata set. Instead, flexibility/extensibility in metadata management can be the crucial feature to enable DMS interoperability at this particular level. Such a feature is typical of an ebXML repository, which can thus be proficiently coupled to other similar legacy modules. For instance, in ebXML repositories custom metadata categories can be easily added, as a simple mechanism to group up sets of related attributes.

With no common set of metadata to be taken as reference, we can think of explicitly working with different metadata sets in a coordinated way. The coordination mechanism, according to the SOA approach, has to be implemented in the software layer that accesses the single services at the different DMS interfaces. This approach to metadata management deeply affects the interface structure of any service involved with metadata. In fact, the service signature must be general enough to allow passing a list of name/value pairs to describe multiple metadata items. On the other hand, the service must be implemented to parse the list and to behave accordingly to what pairs are actually meaningful on the specific DMS platform. In the sample application cited in the next section we have applied a slightly different variant of this approach, choosing to develop each service function in two different flavors: with a static signature, and with a variable set of parameters. The first one is useful to access system-specific metadata, whereas the other can manage any kind of metadata, passed in as name/value pairs.

5 IMPLEMENTED MODULES

We built up a sample implementation of our architecture using popular software modules for the three components. As for the ebXML part, we took into consideration freebXML (freebXML). Moreover, our partner Pivot Consulting s.r.l. provided us with Hummingbird DM, a well-known commercial DMS: we managed to make it an *interoperable* DMS implementing the Web service wrapper we described as a standard interface. For the controller role we developed a demonstrative Web application, in order to allow user interaction (this should be changed in an automated scenario).

6 RELATED WORKS

SoDOCM (Russo et al., 2006) is an example of federated information system for document management. The declared goal of the project is to provide an application that can transparently manage multiple heterogeneous DMSs, while retaining their autonomy. It is based on the mediator architecture, a well-known concept developed for database interoperability, and follows the service-oriented computing paradigm.

With the AquaLogic Data Services Platform (Carey, 2006) the authors propose a declarative approach for accessing data, as opposed to the standard procedural way. The system is intended to integrate heterogeneous data sources in order to support composite applications. This is achieved using XML and XQuery technologies, and introducing the concept of data service. A core functionality is the automatic translation between XQuery and various SQL dialects.

LEBONED (Oldenettel et al., 2003) is a metadata architecture that allows the integration of external knowledge sources into a Learning Management System. The example presented in the paper operates with the eVerlage digital library, which provides a Web service interface to import documents and related metadata.

OAI compliance and metadata re-modeling are a central issue of the eBizSearch engine (Petinot et al., 2003). In this paper, the authors describe how they managed to enable OAI access to the CiteSeer digital library; the proposed solution consists in mirroring the original library and extending this external database to meet OAI specifications. Synchronization between the two systems is performed periodically.

7 CONCLUSIONS AND FUTURE WORKS

Integrating different content management systems within a real-world, modern enterprise presents several interoperability issues, both on the access methods and the metadata models. Despite of the existence of a widely accepted standard, the ebXML specifications, most companies still operate upon legacy systems and pay little attention to interoperable infrastructures. In this paper, we presented an architectural solution for the integration of traditional DMSs with a standard ebXML Registry, allowing advanced metadata management over preexisting legacy systems, while keeping the old information system up and running. As a side achievement, we defined a universal interface for Web service-based access to a generic DMS, which can be used independently from our architecture.

The proposed solution is focused on architectural issues, yet leaving out a few aspects located at the application level and regarding metadata consistency. The first one is the initial integration: the ebXML Registry will need to be synchronized with the DMS in order to replicate the metadata inside it. Afterwards, consistency must also be ensured for every *write* operation; hence, some kind of transaction mechanism should be implemented to allow atomic modifications of the repository and commit/rollback functionalities. Our future work will also involve a more thorough analysis of administration functions, which are excluded from the current version of the DMS interface.

ACKNOWLEDGEMENTS

We wish to thank Pivot Consulting s.r.l. (Navacchio, PI), for the support provided throughout the entire project.

REFERENCES

Gartner Inc., Gartner's Positions on the Five Hottest IT Topics and Trends in 2005, http://www.gartner.com/DisplayDocument?doc_cd=125868

FileNet Corporation, 2007, IBM Filenet P8 Platform, http://www.filenet.com/English/Products/Datasheets/p8brochure.pdf

EMC Corporation, 2003, Developing Web Services with Documentum, http://www.software.emc.com/collateral/content_management/documentum_family/wp_tech_web_svcs.pdf

Vignette Corporation, 2005, Vignette Portal and Vignette Builder, http://www.vignette.com/dafiles/docs/Downloads/ WP0409_VRDCompliance.pdf

Blair, D.C., 2006, The Data-Document Distinction Revisited, *ACM SIGMIS Database for Advances in Information Systems* - Winter 2006 (Vol. 37, No. 1)

Carey, M., 2006, Data delivery in a service-oriented world: the BEA aquaLogic data services platform, *in Proc. of the 2006 ACM SIGMOD international conference on Management of data*

Russo, L., and Chung, 2006, S., A Service Mediator Based Information System: Service-Oriented Federated Multiple Document Management, *10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*

Oldenettel, F., Malachinski, M., and Reil, D., 2003, Integrating Digital Libraries into Learning Environments: The LEBONED Approach, *Proceedings of the 2003 IEEE Joint Conference on Digital Libraries*

Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org

OASIS, 2005, ebXML Registry Information Model specification version 3.0, http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf

OASIS, 2005, ebXML Registry Services specification version 3.0, http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf

Electronic Business using eXtensible Markup Language, http://www.ebxml.org/

United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/

Open Archive Initiative, http://www.openarchives.org/

Petinot, Y., et al. 2003. eBizSearch: an OAI-Compliant Digital Library for eBusiness. *In Proc. of JCDL 2003,* IEEE CS Press, 199-209

Bechini, A., Tomasi, A., and Ceccarelli, G. The Ecumene Experience to Data Integration in Cultural Heritage Web Information Systems. *In Proc. of CAiSE Workshops* (1) 2004: 49-59

Bechini, A., Cimino, M.G.C.A., Marcelloni, F., and Tomasi A., 2007, Patterns and technologies for enabling supply chain traceability through collaborative e-business. *Information & Software Technology*, Elsevier (2007), doi:10.1016/j.infsof.2007.02.017

Bechini, A., Cimino, M.G.C.A., and Tomasi A., 2005, Using ebXML for Supply Chain Traceability - Pitfalls, Solutions and Experiences, *in Proc. of 5th IFIP I3E Conf.*, Springer, Oct. 2005, pp. 497-511