AN XML-BASED APPROACH FOR CONTENT MANAGEMENT IN VIRTUAL ORGANISATIONS

Marius Ioan Podean and Lucia Rusu
Faculty of Economics and Business Administration, Babes Bolyai University, Teodor Mihali, Cluj-Napoca, Romania

Keywords: Content management, e-Workspace, Virtual and distributed organisations, XML, Workflow, Access control.

Abstract: An important factor in the efficiency of distributed and virtual organisations is the ability to manage the content involved in every day activities, content that most often resides in documents. A collaborative document editing system that can automate and coordinate content production processes and provide flexible content manipulation, like combining data from multiple sources, can increase team efficiency and allow users to concentrate their efforts on content development. This paper presents a model implementing this approach in order to provide an efficient e-workspace that can be customized for wide variety of virtual organisations. The use of XML proves to be an efficient solution to provide a single source and multi publishing channels approach.

1 INTRODUCTION

Among distributed organisations and teams the ability to work together from different locations (Marca, 2008) and using different devices is an important aspect. As a great amount of information present in distributed groups is stored in documents, a key factor in their efficiency is the ability to manage the content of these documents. The development of web capabilities is significantly changing the way that people work, facilitating content creation and providing easy means for distribution. Taking in consideration the great interest that structured documents gained in the recent period, the aforementioned requirements are rarely placed in this context and at this moment technology offers just partially solutions for these necessities.

As shown by Salminen (2005), structured documents have a range of benefits, well-known even before XML existed, that can significantly reduce the work involved by document creation and publishing and ensure long-term information accessibility. Recently, the use of XML as a basis for document production has been largely adopted in a wide range of organisations and working teams (e.g. research (Yongguo, 2008), legislative system (Heero, 2002)) thanks to its ability to be both a document and a metadata format.

Discussing about e-business innovation, Marca (2008) defines a model based on a highly flexible front-end and a highly standardized back-end (enabling thus cost-effective operations). XML is identified as the required technology in order to implement the concept of flexibility with standardization. In their attempt to provide a greater support for collaboration and content management, new software architectures (e.g. cloud computing, software as a service) take advantage of just a small range of benefits provided by XML technologies. In order to be able to provide a coherent e-workspace that integrates heterogeneous systems and organisations, orchestrate web services and provide rich user interface capabilities, a flexible data interchange format is required.

A system that aims to improve collaboration in distributed organisations should provide efficient and flexible means for content management in order to allow user to spend more time creating valuable content rather than handling technology.

Such a system should try to remove space and time limitations and provide intuitive user interfaces capable of making content creation and management as easy as possible requiring no or minimal knowledge about the technology involved. In the following we will present a framework that tries to handle the aforementioned requirements using XML documents as a basis for content management, independent from the field of application of a

Ioan Podean M. and Rusu L. (2009).
AN XML-BASED APPROACH FOR CONTENT MANAGEMENT IN VIRTUAL ORGANISATIONS.
In *Proceedings of the International Conference on e-Business*, pages 252-257
DOI: 10.5220/0002348025202257

particular type of organisation (e.g. business, legislative, research etc.). The main characteristics of the framework are based on the principle of single source and multiple distribution channels and formats. Documents are regarded as both a data source and a presentation format, enabling thus the use of links between documents and document elements. Combining data from multiple sources is another key concern and our framework tries to impose minimal restrictions regarding the structure of the documents being used.

This work is structured as follows: we start by presenting theoretical aspects concerning document and content management in Section 2. We continue in Section 3 with the design of the **Faust** framework, the model proposed, and present some technical and implementation related aspects in Section 4. Concluding remarks and future work will be presented in Section 5.

2 RATIONALE AND RELATED WORK

In the following we will present some of the theoretical aspects that influenced the design of our framework and briefly highlighting main characteristics of the current technologies approaching this field.

Document management systems (DMS) focus on tracking and storing documents created and exchanged by their authors (Aversano et al. 2001). They provide components for defining metadata for the documents (i.e. date of creation, authors, version etc.), indexing (usually based on metadata), storage and retrieval (based on the unique document identifier). On the other hand, content management systems (CMS) focus on creating, editing and publishing managed content which is most often stored in relational databases. Workflow management systems (WMS) allow the automation of processes within an organization, enabling greater coordination and control among geographically distributed teams (Nallaparaju et al. 2005). Using WMS, the organization can integrate different software technologies, leading to the improvement of the collaborative activities (Aversano et al. 2001).

Structured documents have evolved from basically one simple need: to provide the reader the means to interpret documents in the way that the writer intended. This approach provides a series of important benefits (Salminen, 2005) like consistency and correctness support, rich information retrieval capabilities, information reuse and multichannel publishing, software independence and long-term accessibility of information. The Extensible Markup Language (XML) is a highly flexible tool that supports de definition of custom markup languages and it's possibility to be used as both a document and metadata format has driven researchers to develop advanced management models (Mahboubi, 2009). Models concerning XML data warehousing and XML OLAP have proven XML flexibility and efficiency and taking in consideration that in organisations metadata has sometimes same importance as the data itself has lead to the development of advanced searching techniques like XML data mining (Ding, 2009 and Nayak, 2009).

The **Faust** framework combines key features from the aforementioned technologies (DMS, CMS and WMS) and uses XML documents as a storage support for the managed content. The framework provides DMS functionality in order to handle document creation and retrieval and WMS functionality by automating and coordinating all processes involved in data manipulation. Content management is based on XML technologies and thus implementing a very flexible system for content handling and publishing. This approach enables the definition of documents as sets of layers on top of which application layers can be applied in order to provide intuitive interfaces that allow content interaction. XML technologies can facilitate the single source approach by providing the means to define links between elements residing in different documents, enabling thus the combination of data from multiple sources.

Several solutions concerning some of the aforementioned aspects have already been developed dealing more or less with content management in a collaborative context. Docbook (Walsh, 2005) is a XML vocabulary that allows users to create documents in a presentation neutral form. Using several transformation schemas the content can be published in different formats, dealing especially with technical documentation. The OpenDocument Format is a XML based file format for office productivity applications that defines documents as resource collections based on the separation of content and presentation. It is a complex specification that doesn't place content management in a collaborative context and offers limited support for user defined document definitions. On the other hand, XMetal, a XML based solution, offers support for personalized content using a single source and multiple publishing channels approach, providing though limited workflow capabilities.

3 DESIGN ASPECTS

The **Faust** framework is a web based system designed with the software as a service architecture in mind, although not fully complying with it. It is build around the concept of louse coupling, the main components working as independent units that provide functionality through web services. The main components of the framework handle the management of access, workflow and documents, defining a core functionality that enables the system to be built incrementally.

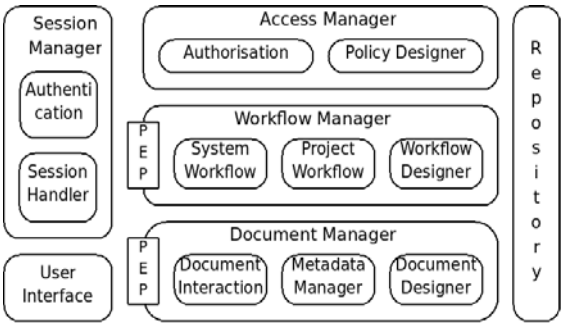

Figure 1: Main architecture.

The *Document Manager* component handles the interaction with the documents, manages metadata attached to them and provides the means to define new document types. The access to this component is controlled by a local policy enforcement point (PEP) and the functionality is provided as web services.

The *Document Designer* component allows users to define new types of documents or extend existing ones. In order to create a new type of document an XML Schema must firstly be defined or imported, and based on this definition the additional layers will tailor the required functionality and presentation. In order to implement this functionality, the framework defines several vocabularies, the most important being represented by the document editing and rendering rules. The document editing rules define the means to create editing stages for a document type, and for each stage, permissions and restrictions regarding viewing and editing are set according to the roles involved in the process. By this mean document level workflow can be defined, enabling thus more flexible project level behaviour to be specified.

The document rendering rules represent a very flexible vocabulary that allows the definition of a set of display rules, independent from the export format, that ensures an uniform representation of the document. These rendering rules allow users to define elements like page set-up and styling in terms close to an usual word editor, and also to associate document structures or elements with certain dynamic editing or data representation elements. The dynamic editing elements represent a binding between the structured documents and rich user interface capabilities that allow users to easily modify the documents inside a web browser concealing the structured nature of the document they are working on (e.g. creating a new paragraph, editing data inside tables or lists is done similar to usual word editors). Data can also have graphical representations (e.g. charts) or represent captions of elements defined in other documents (e.g. link: dynamic caption enabling auto-update when original source is modified; view: static caption, representing the value at a particular moment of time). Based on the property of being independent of export format, elements defined by the document rendering rules are enabled, disabled or enhanced according to user options (e.g. XForms, PDF).

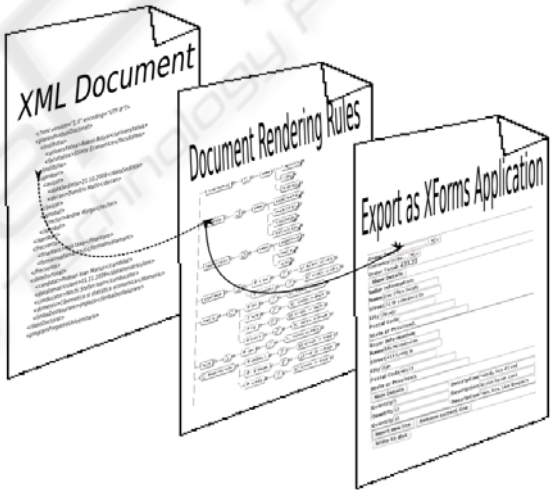

Figure 2: Document layering.

The interaction with documents refers to the actions to be performed on the main documents that act as content containers and, in user interaction terms, it is done through an export document. An export document is regarded as being a collection of layers, the choice of whom depending on the required functionality. The basis layer is represented by an XML document acting as content container that instantiates a particular document definition. The second layer is represented by the associated editing rules, by the means of which (1) the document level workflow is defined and (2) editing permissions and restrictions are applied. According to editing stage, document elements may be disabled

for editing or viewing for certain roles and based on the next layer, represented by the rendering rules, an export format can be generated. From a technical point of view, an export format can be an XForms application, a PDF or a HTML document.

An additional optional layer is implemented in order to provide document level utilities (e.g. notes, comments) that can support creativity and improve collaboration. When using XForms as export format the presence of these utility elements is marked with an appropriate symbol, the content being loaded on request.

The *Document Interaction* component represents the access point to all this functionality, providing the required interface in order to allow the users and the system to interact with the documents. Access is provided in respect to user's role, the component working closely with the local policy enforcement point (PEP). The local PEP manages access based on project policies, being able to grant or deny access to functionality and resources (the local PEP can be regarded as an *Access Manager* delegate that guaranties that the policies defined at project and system level are applied).

In order to be able to manage all layers required to generate an export document, the roles involved in the document level workflow and document type definitions, the *Metadata Manager* component defines a vocabulary that organises all this information concerning a document. In this respect, a document consists mainly in data that describes it's nature and links to files that hold the content of each of it's layers. This component handles the definition of metadata associated with a document and works in close relation with the *Document Interaction* pipe based system. It thus provides all required information in order to process the document layers and apply them the proper transformation, representing at document level an integration component.

The *Workflow Manager* component handles de definition and execution of sequences of operations required by a particular functionality at system or project level. The system and project workflow execution implies orchestrating the functionality provided by the other components, representing the level where the integration of the system as a whole is acquired.

The sequences of operations that aggregate functionality in order to create a process can be defined by the appropriate roles using the *Workflow Designer* component. This component provides the proper means to combine resources, roles and actions (functionality provided by the rest of the system) in a coherent symphony that defines a process either at system or project level. Processes like creating a new account, consulting the list of active projects (system level) or editing and exporting a document (project level) can be defined using a graphical user interface. In order to facilitate the understanding of a process that is executed at a particular moment of time, this component can provide also a visual representation of the workflow involved.

The orchestration of a process represents the automatic coordination and execution of a set of services required in order to obtain a certain output. Each process the system is required to execute (either at system or project level) must have a workflow definition schema in order to be implementable. Taking in consideration that all main components provide a interface in order to communicate one with another, this approach sustains a loose coupled integration of the system. Similar to the *Document Manager*, the access to the functionality provided by these components is controlled by a local PEP.

The framework manages access through the *Access Manager* component which represents the level where decisions regarding the set of permitted operations that can be executed by a particular user during a certain work session are taken. This component implements a role based access control (RBAC) model (Ferraiolo, 2001), relying thus on concepts like users, roles, permissions, objects and sessions. Users are assigned to a particular role and roles are associated with a set of permissions, users being able to execute operations in virtue to the assigned role. Roles represent m-n relationships between users and permissions (taking in consideration the fact that a user can have one or more roles, but just one of them can be activated during a certain work session) and permissions represent the authorization to execute a set of actions upon protected object. A great advantage of this model is that it allows the definition of hierarchic levels inside the system, being able to emphasize the authority and responsibility layers. In order to eliminate issues like interest conflicts deriving from multiple inheritances, the RBAC model introduces the concepts of static and dynamic separation of duty (a user being able to activate just one role during a particular work session, or multiple roles and define conflictual situations).

Using both static and dynamic separation of duty the system gains a great degree of flexibility when applying access policies, being able thus to simulate more realistically real life organizational model.

Together with the local PEP's the *Authorization* component manages access to system resources.

The *Repository* stores all documents required by the aforementioned components and can be accessed exclusively by these. Access policies, workflow schemas, document definitions, user interface elements, project documents etc. are all stored in this repository, being available only to the components delegated to process them (based on system level roles). User created documents are organized in libraries (a library containing all documents defined on a server, site) and projects (a project containing all documents created by users during the execution of a workflow) and components can access them using temporary links provided by path projection.

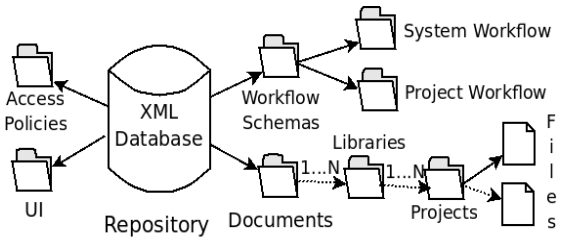

Figure 3: Repository structure.

The two main functions provided by the *Session Manager* refer to authentication and session handling, providing thus the means to identify users and manage the connection that has been established with them. After a user successfully authenticates, the session handler triggers the execution of the authorisation workflow. After the user has been granted with a role at system level, the means to access the functionality associated with that particular role are provided (e.g. create new projects or edit existing ones, define new access policies etc.). The *UI Support* component provides additional graphical elements that enhance usability and user experience, representing extra layers added on top of a normal web page that simulate the looks of a desktop that provides functionality using different applications (e.g. a Start button that provides access to main applications like the document editor, workflow manager etc.). The main purpose of this component is to simplify and enhance the interaction with the system and conceal the structured nature of the documents by emphasizing the flexibility provided by this approach.

The **Faust** framework integrates structured documents with concepts like role based access control and workflow management in order to obtain an efficient user and document centric collaborative system concerned with the management of content. The framework philosophy is build around the concepts of distributed organisations, data integration and interoperability. The following section will present certain technical aspects meant to provide a greater understanding of the model, concentrating only on those elements required by this purpose.

4 TECHNOLOGICAL OVERVIEW

In the following we will present certain implementation related aspects in order to provide a clearer view of the system, confining only to those elements that support this goal.

As mentioned earlier, the **Faust** framework provides functionality using web services following the Representational state transfer (REST) model in defining how resources are addressed and manipulated. This model abstracts application state and functionality into resources that provide a uniform interface to support a stateless client-server interaction. In this respect, the session handler does not rely on sessions and cookies to manage the interaction with the user, instead random ids and time stamps are associated to authenticated users after each transaction (making thus session hijacking more difficult). This approach is possible because the system design allows components to leave data in a consistent state after each process execution. The REST model provides also the means to implement a loose coupled mechanism in order to integrate system components.

Using XML pipelines allows the framework to provide very flexible and powerful capability to process documents. Starting from the metadata that defines a document, the *Document Interaction* component applies several transformations based on the layers required by the export format in order to provide a complete document. For example, when exporting into a XForms application, the pipeline first loads the document's definition in XML Schema format and (1) based on the document metadata and associated document editing rules the editing stage is determined and according to this stage the elements available for viewing/editing are extracted and passed as instance data for the XForms application, (2) based on XML Schema and instance model binding elements are generated in order to associate elements with data types, (3) following the instance model and associated document rendering rules the user interface components are generated. When

deploying the final XForms application the UI Support component is consulted and based on the process that is being executed an editor is provided that wraps the XForms application in order to improve usability.

Processes are orchestrated using Yet Another Workflow Language (YAWL) because of the good support for coordinating both human and computer tasks. YAWL is based on workflow patterns and because of its formal semantics foundation static analysis is available. Another important feature provided by this language is the support to integrate other technologies like XML Schema, XPath and XQuery in the execution of workflows.

Access management is backed by the model of the eXtensible Access Control Markup Language (XACML). Based on this declarative access control policy language the RBAC model can be implemented providing thus the means to simulate more realistically organisation models even in distributed contexts.

5 CONCLUDING REMARKS

The work presented in this paper addressed the problem of integrating document and content management functionality and workflow capabilities into a single system in order to achieve a very efficient solution for collaborative content management. Allowing users to edit collaboratively documents and supporting their needs for time and location independence can result in increased efficiency at organisation level. Users have to be able to concentrate their efforts on content and reduce as much as possible the time used to handle and integrate technologies.

As shown in this paper, using XML technologies as building blocks enables the system to provide great flexibility and cover a wide range of aspects concerning content at organisational level. The proposed framework tackles issues concerning the improvement of efficiency at organisation level using flexible content management solutions and represents a work in progress. As further work, we intend to develop our framework in a stable system in order to support with use-cases aspects presented through this paper.

ACKNOWLEDGEMENTS

This work was supported by Romanian National Authority for Scientific Research under the grant no. 92-100/2008 (PN2).

REFERENCES

Aversano, L., Canfora, G., Lucia, A. & Gallucci, P. (2001). “Integrating Document and Workflow Management Systems”, Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments

Ayed, S., Cuppens-Boulahia, N., Cuppens, F. (2008). “Deploying Access Control in Distributed Workflow. Thirty-One Australasian Computer Science Conference” (ACSC2008)

Ding, Q. (2009). “Data Mining on XML Data”. In J. Wang (Ed.), Encyclopedia of data warehousing and mining - 2nd ed. (pp. 506-510). London: Information Science Reference

Ferraiolo, D., F., Sandhu, R., Gavrila, S. (2001). “Proposed NIST Standard for Role-Based Access Control,” ACM Transactions on Information and System Security 4(3)

Heero, K., Puus, U., Willemson, J. (2002). “XML based document management in Estonian legislative system,” Proceedings of the Baltic Conference, BalticDB&IS 2002 (Vol. 1), Institute of Cybernetics at Tallin Technical University

Mahboubi, H., Hachicha, M., Darmont, J. (2009). “XML Warehousing and OLAP”. In J. Wang (Ed.), Encyclopedia of data warehousing and mining - 2nd ed. (pp. 2109-2116). London: Information Science Reference

Marca, D.A. (2008). “E-business innovation. Surviving the coming decades,” in proc. ICE-B'08

Nallaparaju, V., et. al. (2005). “An Experimental Study of Workflow and Collaborative Document Authoring in Medical Research”, Proceedings of Tenth Australasian Document Computing Symposium

Nayak, R. (2009). “Discovering Knowledge from XML Documents”. In J. Wang (Ed.), Encyclopedia of data warehousing and mining - 2nd ed. (pp. 663-668). London: Information Science Reference

Salminen, A. (2005). “Building digital government by XML,” in proc. 38th Annual Hawaii International Conference on System Sciences

Walsh, N., Muellner, L. (2005). DocBook 5.0: The Definitive Guide, O'Reilly & Associates

Yongguo, J. (2008). “Implementing Marine XML for Observed CTD Data in a Marine Data Exchange Platform,” Journal of Ocean University of China, 7(4)