WEBC-DOCS: A CMS-BASED DOCUMENT MANAGEMENT SYSTEM

João de Sousa Saraiva, Alberto Rodrigues da Silva
INESC-ID & SIQuant, Rua Alves Redol, 9, 1000-029 Lisboa, Portugal
joao.saraiva@inesc-id.pt, alberto.silva@acm.org

Keywords: Content Management System, Document Management System, WebComfort.

Abstract: Content Management Systems (CMS) are typically regarded as critical software platforms for the success of organizational web sites and intranets. Nevertheless, a simple CMS alone does not provide enough support for a typical organization's requirements, such as document management and storage. On the other hand, Enterprise Content Management (ECM) systems are typically regular web-applications that provide such functionality but without the added advantage of being based on a component-based CMS platform.
This paper presents the architecture and major technical details of WebC-Docs, a highly-customizable toolkit (for the WebComfort CMS platform) that provides document management functionality. Due to this, the WebC-Docs toolkit can be configured and used in various kinds of scenarios.

1 INTRODUCTION

The Internet's recent expansion has led to the appearance of many web-oriented CMS (Content Management Systems) (Boiko, 2001) and ECM (Enterprise Content Management) (Kampffmeyer, 2006; Rockley, 2002) platforms with the objective of facilitating the management and publication of digital contents.

CMS systems can be used as support platforms for web-applications to be used in the dynamic management of web sites and their contents (Carmo, 2006; Saraiva and Silva, 2008). These systems typically present aspects such as extensibility and modularity, independence between content and presentation, support for several types of contents, support for access management and user control, dynamic management of layout and visual appearance, or support for workflow definition and execution. On the other hand, ECM systems are typically regular web-applications oriented towards using Internet-based technologies and workflows to capture, manage, store, preserve, and deliver content and documents in the context of organizational processes (AIIM, 2009). These two content-management areas are not disjoint, and it is not unusual to find a CMS system acting as a repository for an organization's documents and contents, albeit at a very "primitive" level (e.g., no checking for duplicate information, no grouping of documents according to a certain logical structure, and no possibility of providing metadata for each document).

Into this context comes the WebC-Docs system, a document management toolkit for the WebComfort CMS platform (WebComfortOrg, 2009) providing a large set of configuration points that allow the system to be used in a wide variety of application scenarios.

In this paper we present the architecture of WebC-Docs and its major technical contributions. This paper is structured as follows. Section 1 introduces CMS and ECM systems, and the classical problem of document management in an organizational context. Section 2 presents WebC-Docs' architectural aspects. Section 3 provides a brief discussion of some additional issues regarding this system. Section 4 presents related work that we consider relevant for this project. Finally, Section 5 presents the conclusions so far, as well as future work.

2 WEBC-DOCS

WebC-Docs is a document management toolkit for the WebComfort CMS platform (Saraiva and

Silva, 2008; WebComfortOrg, 2009), featuring a component-based architecture that makes it easily extensible, highly configurable, and adequate for several kinds of application scenarios.

This section presents some relevant components and aspects of WebC-Docs' architecture, namely: (1) its main concepts; (2) its integration with the WebComfort CMS platform; (3) its Explorer-like web interface; (4) the Dynamic Attributes mechanism; (5) the Indexing and Searching functionality; (6) the Permissions mechanism; and (7) the facilities for using additional repositories to store and locate documents.

2.1 Main Concepts

The WebC-Docs system consists primarily of the following concepts, also presented in Figure 1: (1) `Document`; (2) `Folder`; (3) `Document Shortcut`; and (4) Document and Folder `Permissions`.

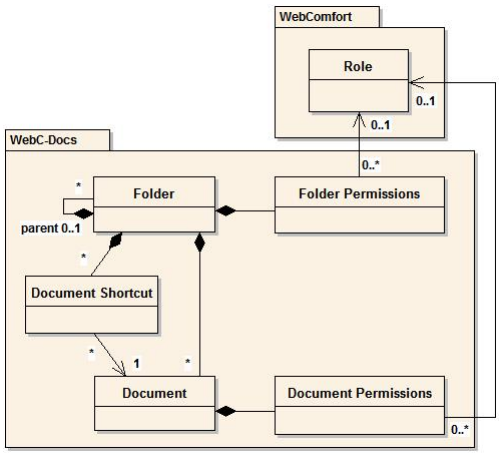

Figure 1: The main concepts of the WebC-Docs system.

A `Document` is the main entity of WebC-Docs; in a typical usage scenario, it represents a "real-world" document (the document's digitized file may or may not be included in the `Document`; WebC-Docs allows the existence of `Documents` as simple "placeholders" for the real documents that they represent). On the other hand, a `Folder` consists of a `Document` container, but it can also contain `Document Shortcuts`, which are mere "pointers" to regular `Documents`. Thus, although a `Document` must be located in exactly one `Folder`, it is possible for it to be accessed from other `Folders`, by using `Document Shortcuts`.

Finally, Document and Folder `Permissions` are associated with `Documents` and `Folders`, respectively, and specify what actions each WebComfort role can perform over them (the Permissions mechanism is explained further down this section).

2.2 Integration with WebComfort

One of the important innovations in WebC-Docs is its integration with a CMS platform, in this case WebComfort (Saraiva and Silva, 2008). This integration consists mainly of the following points: (1) the user's main interaction points with WebC-Docs are implemented as WebComfort modules (e.g., Document Management, Statistics) that take advantage of the facilities provided by WebComfort; (2) the Permissions mechanism (described below) uses WebComfort roles and users, instead of (re)defining those concepts again; and (3) WebC-Docs is distributed as a WebComfort toolkit, and so it can installed on any WebComfort installation in a simple and automatic manner. Figure 2 shows an overview of WebC-Docs' integration with WebComfort and other toolkits (because of space limitations, this integration will not be presented in detail in this paper).

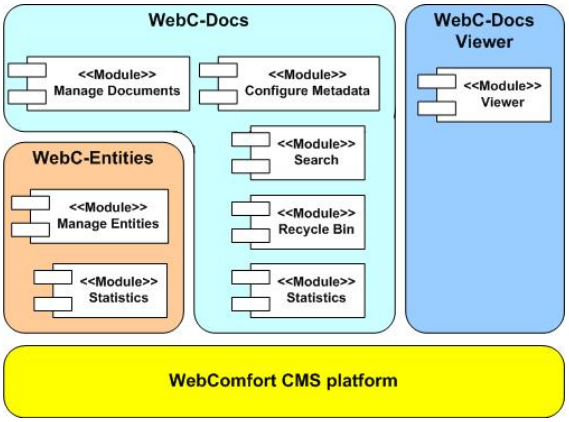

Figure 2: An overview of the integration with WebComfort.

Besides all this out-of-the-box functionality provided by the CMS, an added advantage of such an integration is that the system can be easily adapted to requirements such as those derived from an organization's structure or size (e.g., an organization's document management portal can consist of a select number of tabs – searching, management, configuration –, or it can provide a tab for each of its users, each tab containing the document management modules that are adequate for the user's responsibilities).

It is important to note that, although WebC-Docs' current implementation is based on the WebComfort CMS platform, it does not use concepts exclusive to WebComfort (e.g., tabs, modules, and roles are concepts that can be found in many CMS platforms). Thus, WebC-Docs could easily be migrated to other CMS platforms, as long as they provide adequate support for CMS-based web-applications.

2.3 Explorer-like Web Interface

One of WebC-Docs' main goals is to be intuitive to the "average" user. To achieve this goal, we designed the main Document Management module to be similar to the typical file explorer interface (such as Microsoft's Windows Explorer), as shown in the screenshot in Figure 3: the left side of the module displays the system's folder structure (always considering the current user's permissions), while the right side shows the documents within the selected folder (if any).

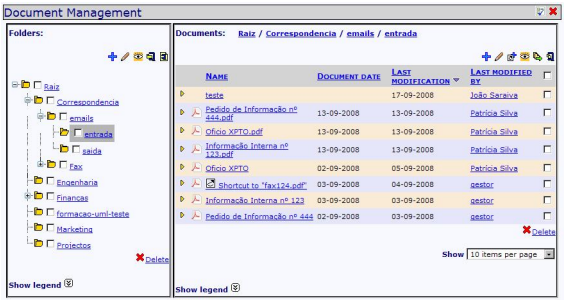

Figure 3: The main interface: folders and documents.

The user can perform various actions over each folder and document (such as "Delete", "Move to", "Create shortcut to", and "Export to ZIP file"), which are displayed over the folder and document listings, respectively. Documents can also be downloaded immediately (by clicking on the file type's icon), or the document's details page – illustrated in Figure 4 – can be shown (by clicking on the document's name).

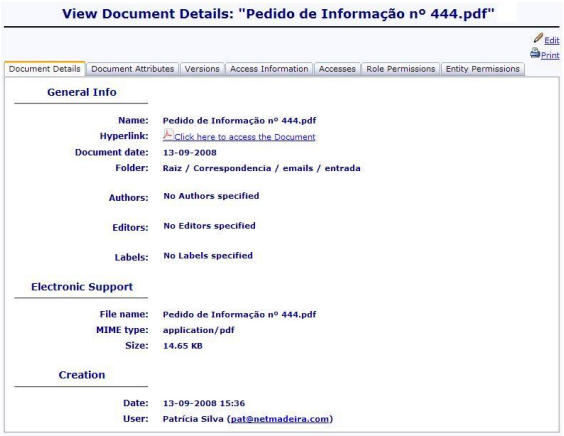

Figure 4: The Document Details page.

WebC-Docs also provides the Recycle Bin (not to be confused with Microsoft Windows' own Recycle Bin), to which deleted documents will be moved. It will be up to a user with the "Documental Manager" role (which can be configured to be any one of WebComfort's roles) to regularly check the Recycle Bin and purge it, or restore documents that have been deleted by mistake. The Recycle Bin can be disabled, although this is generally not recommended, as deleting a document would become an irreversible action.

2.4 Dynamic Attributes

One of WebC-Docs' most powerful features is the possibility of specifying metadata, in a customizable manner that can be adjusted to the organization's document information requirements. This mechanism, designated *Dynamic Attributes*, is based on the notions of `Attribute Set`, `Attribute Definition`, and `Attribute Type`, illustrated in Figure 5.

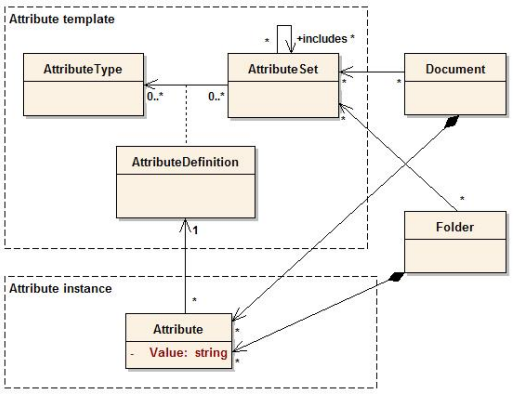

Figure 5: The "Dynamic Attributes" mechanism.

An `Attribute Set` consists of a group of `Attribute Types`, and possibly even other `Attribute Sets`, allowing the specification of metadata using a tree-like structure. `Attribute Type` is responsible for defining the various kinds of attributes that be used (e.g., integers, dates, enumerations). This is done by pointing the attribute type to a class that implements an interface with which WebC-Docs will communicate: this class will provide the controls for viewing and editing attribute values, as well as conversion to/from the type's string representation. The associations between `Attribute Sets` and `Attribute Types` are called `Attribute Definitions`, and they configure the `Attribute Types` in the context of the `Attribute Set` (e.g., name, default value, whether it is read-only).

A user can apply any `Attribute Set` to any Document or Folder, any number of times (e.g., to provide various author contacts for a certain Document). `Attributes` are used to store user-provided values.

2.5 Indexing and Searching

For a document management system to be of any use, it must allow its users to find documents given only some information about them (typically a string that is a part of the wanted document). WebC-Docs provides a module with search functionality, using the Lucene.Net indexing engine (LuceneNet, 2009), over all document information (i.e., regular document attributes, dynamic attributes, and document file contents). There are two types of search: (1) "regular search", which operates like common search engines (if any part of a document matches the user's search terms, the document is part of the search results); and (2) "advanced search", in which the user can specify a variety of options (e.g., filter by dates, find all documents with a certain string in their file contents) in order to refine the search and reduce the number of false positives that will be returned by it.

Figure 6 presents the main components of WebC-Docs' file indexing. Whenever a Document is created/altered, it is supplied as input to the `Indexer Front Controller`, which quickly analyzes the Document (namely its MIME type and file extension) and forwards it to any adequate `Base Content Indexers`. Those indexers will then parse the Document's file contents and invoke WebC-Docs' internal API (which, in turn, provides a wrapper around Lucene.Net). Document metadata (regular document attributes and dynamic attributes) is always indexed, so the Document can still be found in the system.

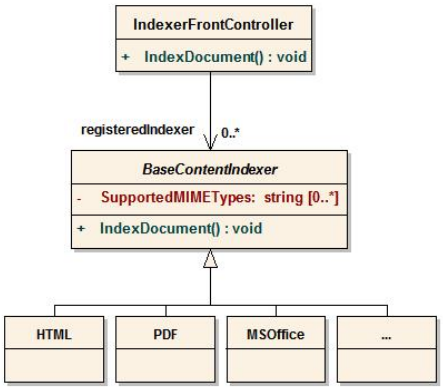

Figure 6: Indexing support for many file formats.

WebC-Docs also allows searching in the Document's own repository (if the repository allows it, of course), by means of the *Additional repositories* mechanism, described further down this section.

We consider it relevant to note that, although this indexing and searching functionality uses Lucene.Net, the user's search terms are not directly provided as input to Lucene.Net. Instead, WebC-Docs defines its own textual search language (similar to Lucene.Net's and Google's own search engine), and uses a search term compiler built using CSTools (CSTools, 2009). This compiler is used to both validate the user search terms (e.g., too many parenthesis) and generate an AST (Abstract Syntax Tree) that will be provided as input to WebC-Docs' internal searching API (which will, in turn, transform that AST into a Lucene.Net query string, and use that string to query the various indexes that WebC-Docs uses).

This intermediate search step provides two main advantages. First, it allows us to overcome some limitations of Lucene.Net, such as: (1) some efficiency issues when performing certain kinds of search (e.g., search without knowledge of the term's prefix characters), or (2) its inability to search for all documents *except* those that match a certain criteria (e.g., a search string like "NOT document" in WebC-Docs would return all Documents that do not contain the word "document" in them, but this search string would be considered invalid in Lucene.NET). Second it provides the added advantage of additional pre-processing over the user search terms (e.g., to remove potentially dangerous words). Another possible advantage would be that, if we later decided to use a different indexing engine, the search language used by WebC-Docs's users would not need to be changed.

2.6 Permissions

Permissions specification, which is perhaps the most important requirement in this kind of application, is addressed by WebC-Docs through the concepts of `Document Permissions` and `Folder Permissions`. This permissions mechanism follows the typical ACL (Access Control List) philosophy, in which each of a user's operations (even viewing) performed over a document/folder are first checked for the user's permissions, to ensure that the user can perform that operation over the specified document/folder. To simplify the system's usability, permissions are inherited (from parent folder to child folder/document) by default.

The system can be configured to interpret permissions in either the *optimistic* or *strict* perspectives. In the optimistic perspective, access is granted to a user over a certain folder/document if any of the user's roles has an explicit "allow" permission to it. On the other hand, the strict perspective follows a more traditional approach, by blocking access if *any* of the user's roles has permissions explicitly stating that access to the specified folder/document is blocked.

Figure 7 presents a folder's permissions editing screen: for each of the roles defined in the WebComfort installation, a large variety of permission options can be configured (permissions for anonymous users are also supported, if the organization wishes to make some information publicly available).

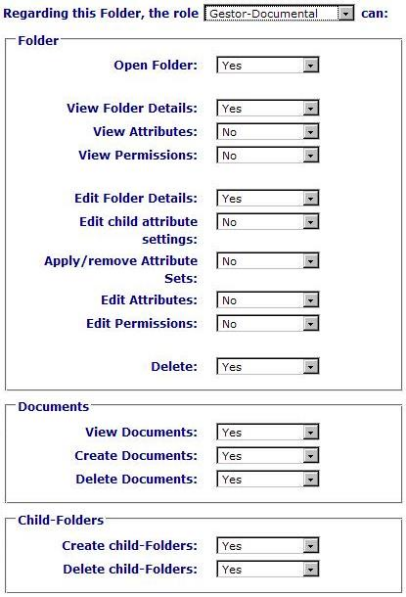

Figure 7: The permission options for a folder.

Also, considering that these permissions are likely to be accessed on a very frequent basis, WebC-Docs uses a "permissions proxy" mechanism. This proxy stores information about accessed permissions in the local server's memory, which accelerates future lookups of those permissions (e.g., for a user's access to a certain popular document). We say that this is a proxy (and not just a simple cache) because all permission modifications go through this proxy, which removes the need to invalidate/remove entries (we just need to update the proxy's information).

2.7 Additional Repositories

Although the majority of WebC-Docs' usage scenarios will likely be based on the server's local file-system for document storage, and a local database server for additional info (e.g., metadata), WebC-Docs can nevertheless use additional types of repositories for document storage and search; this can be useful for scenarios in which an external document source already exists (e.g., a DSpace repository (DSpace, 2009) in an academic context) and that source repository should be used by WebC-Docs (instead of using just a local server repository). All of WebC-Docs' interactions with any repository (whether local or remote) are done through either `Base Folder Repository` or `Base Document Access` objects, shown in Figure 8.

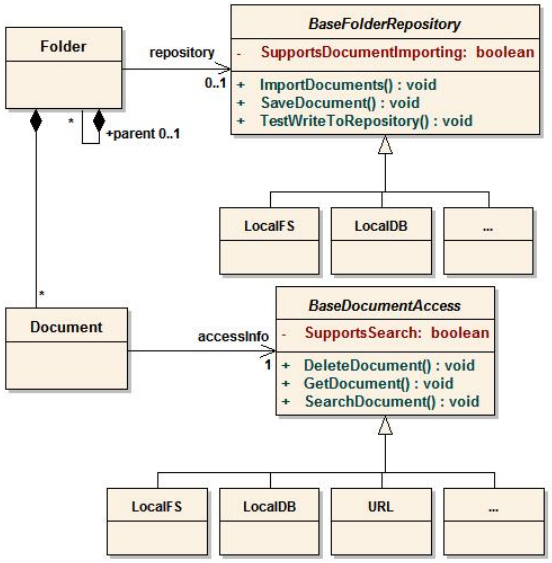

Figure 8: Support for multiple kinds of repositories.

The reason why repository access has been divided into these classes, instead of using a single class with repository access methods, is because all the information needed to access a document file, or a folder repository, is recorded both in the `Document` and `Folder` classes. This allows us to correctly handle cases in which a document, created in a folder F1 (using repository R1), is later moved to another folder F2 (using a different repository, R2). This avoids moving files between repositories, which could present additional issues (e.g., temporary repository failure). This is also the reason why search methods are found in the `Base Document Access`, instead of `Base Folder Repository`: to avoid contacting a repository regarding a search for documents that are not really there.

3 ADDITIONAL ISSUES AND VALIDATION

Although the previous section has presented some important aspects of WebC-Docs, it has not presented some key issues. This section presents a brief discussion of those issues, as well as a few additional notes, and some case-studies that were used to validate this system.

A very important aspect in document management systems (in fact, in any kind of collaborative system) is *traceability*. WebC-Docs addresses this aspect in two complementary ways. Any action performed by a user (even an anonymous user) is recorded in WebComfort's own logging mechanism, along with all information that is relevant for that action, so that the CMS administrator can analyze that log. Also, all document-related operations are recorded in WebC-Docs' own historic records (which are not modifiable, and can be viewed in the Document's details by anyone with the required permissions).

Document versioning is another aspect addressed by WebC-Docs (i.e., it is possible to revert a Document back to the state in which it was at a certain point in time), but this is not described here due to space limitations.

WebC-Docs supports the specification of a Document's Authors and Editors through the functionality provided by another WebComfort toolkit, "WebC-Entities", that provides concepts such as `Entity`, `Person`, and `Group`. This allows WebC-Docs users to specify that "Person X is one of the authors of this document", instead of just supplying a meaningless string of characters (which, in practice, usually do not really convey much useful information).

Bootstrapping is a typical problem in any system that involves configuring permissions; in WebC-Docs, by default no role has permissions to any Document or Folder. The bootstrapping problem here is in ensuring that, although no particular role has access to a Document or Folder (by default), it is still possible to configure Folders and Documents. This has been handled by making the CMS Administrator role (which *always* exists, and is granted only to a few select users) be automatically granted access to any Document or Folder; thus, it is the CMS Administrator's responsibility to configure initial settings and permissions for WebC-Docs.

Regarding the Dynamic Attributes mechanism (shown in Figure 5), we consider the following notes relevant: (1) regarding the "Attribute template" section, it is not unlike the relations that the UML metamodel itself establishes between Class, Property, and Type (OMG, 2005); and (2) regarding the "Attribute instance" section, it is not a case of linguistic instantiation, but rather ontological instantiation (which is explained in (Atkinson and Kühne, 2003)).

It is important to note that WebC-Docs has already been validated in a number of case studies, such as:

- SIQuant's institutional web-site (at `http://www.siquant.pt`), as well as (WebComfortOrg, 2009), which were already using the WebComfort CMS platform, are now using the WebC-Docs toolkit to make available a variety of documents (e.g., white-papers, user manuals);
- Our own research group's document repository (`http://isg.inesc-id.pt/gsidocs`) uses WebC-Docs to make our scientific production publicly available to the community;
- WebC-Docs is also being used by a Portuguese real-estate-related company, to organize and catalog their vast paper-based documentation archive.

Finally, it should be mentioned that, although WebC-Docs has not yet been integrated with other systems (e.g., DSpace (DSpace, 2009)), it already allows external sources to access search results by means of RSS (Really Simple Syndication): a WebC-Docs search module can make its results available as a RSS feed, which can be consumed by an additional (light-weight) WebComfort toolkit called "WebC-Docs Viewer", or by any RSS feed reader (even a regular reader, such as those provided by most current browsers, can view those feeds). Document details are made available in the feed by using the RSS extension mechanism, and the generated feeds are totally compliant to the RSS specification.

4 RELATED WORK

Besides WebC-Docs, other document management systems exist which handle some of the issues presented in this paper. In this section, we present some of which we consider most relevant, while making a comparison between those systems and WebC-Docs.

OpenDocMan (OpenDocMan, 2009) is a free, open-source, document management system that is designed to comply with the ISO 17025 and OIE standard for document management (OpenDocMan, 2009). Like WebC-Docs, it features a fully-web-based access mechanism (through a regular web browser), a fine-grained access control to files, and automated install and upgrades. OpenDocMan supports the concept of "transaction" because any document is in either the "checked-in" or "checked-out" states; WebC-Docs does not use this philosophy on purpose, because it would make it impossible for users to even *view* a document if it was checked out by a different user (e.g., because the user forgot to check the document back in). Also, like in WebC-Docs, OpenDocMan allows its administrator to add additional fields besides "document category" to further describe the document; however, those OpenDocMan fields are only strings, while WebC-Docs supports additional fields of any type (e.g., maps) through its Dynamic Attributes mechanism.

DSpace (DSpace, 2009) is an "open-source solution for accessing, managing, and preserving scholarly works" (DSpace, 2009). It was designed to be as standards-compliant as possible (e.g., it supports the Open Archives Initiative's Protocol for Metadata Harvesting (OAI-PMH) v2.0, and its metadata schema currently supports the Dublin Core specification, although its developers hope to support an IMS/SCORM subset in the near future).

DSpace supports a fixed tree-hierarchy: (1) communities; (2) collections; (3) items; and (4) files. Although this hierarchy is immutable, the user-interface can be adapted to "mask" some of its aspects (e.g., show a community as an aggregation of similar collections). However, this limitation can make DSpace unsuitable for some (non-academic) cases – although DSpace's objective is to support the scholar community, and not a wider enterprise-like community.

It supports the addition of metadata fields to a resource-submission form, by adding a "name"–"type of HTML element to use for input" element to a textual file. Additionally, like WebC-Docs, its searching mechanism takes advantage of the available metadata (provided with each submitted resource) to provide more accurate search results. It should be noted that, unlike in WebC-Docs, some kinds of changes to the DSpace can only be done through the editing of textual files by an experienced administrator.

One of DSpace's primary goals is to handle the *archive and preservation* of documents. To this end, DSpace supports two different kinds of preservation: (1) *bit preservation*, which is like the typical file-storage mechanism (the document's bits are always the same); and (2) *functional preservation*, in which the document's contents are continuously adapted to the new formats, accompanying the passage of time (and subsequent evolution of formats). Also, both contents and metadata can be exported at any time, using a XML-encoded file format. DSpace uses the Handle system (DSpace, 2009) to provide unique identifiers for each managed resource, ensuring that its document identifiers will be valid for a long time.

KnowledgeTree's document management system (KnowledgeTree, 2009) is a commercial system featuring functionalities very similar to what can be found in current CMS systems (e.g., widgets); in fact, it could even be considered by some as a "document management system-oriented CMS". However, unlike other web-based document management systems, KnowledgeTree's solution uses a rich-client (Windows-based) application combined with a web-application (installed on a web-server, and also providing a web-services communication layer), which enables a variety of useful operations (e.g., it allows drag-and-drop operations between local machines and remote document repositories). This rich-client perspective also makes it able to integrate with Microsoft Office applications. On the server-side, it can index files of various types, namely: (1) Microsoft Office; (2) PDF; (3) XML; (4) HTML; (5) RTF; and (6) text.

Its underlying concepts are very similar to WebC-Docs', namely the concepts of Folder and Document, which we believe proves that the "Document and Folder" metaphor used in WebC-Docs is adequate for this kind of system. It can have forums associated to documents, to enable discussions about certain documents (or even other subjects), a feature that could easily be found in CMS-based systems (forums are typical examples of functionality offered by a CMS).

Its metadata-handling mechanism is also very powerful, allowing metadata of various types, which can be marked as "required"; nevertheless, WebC-Docs' Dynamic Attributes mechanism also supports these features. The system also allows searching over metadata and over document contents (if they are indexed), like WebC-Docs. An interesting feature of this system is its concept of *document type*, which can be used to specify document categories, with associated metadata fields; while WebC-Docs does not support this kind of concept (as we believe that there can be many types of document, which are not mutually exclusive among themselves), it does support specifying the automatic application of an `Attribute Set` to new `Document`s, on a `Folder` basis. Also, it allows the usage of tags and tag clouds; WebC-Docs supports tags, but only at a fairly basic level, because they can also be easily replaced by using Dynamic Attributes.

The system does address versioning of documents and metadata, allowing for a document to be reverted back to a previous point in time. WebC-Docs also supports document versioning, but only for regular metadata (e.g., Authors and Editors, comments); Dynamic Attributes are not versioned yet.

Like OpenDocMan, the system allows a "check-in"/"check-out" philosophy. Nevertheless, this feature can be regarded as irrelevant, considering that the system supports the specification of workflows (this philosophy can easily be implemented by a system that supports workflows), combined with its support for document permissions.

Finally, we believe that one of WebC-Docs' greatest advantages over KnowledgeTree's system is its explicit usage of a CMS extensible platform (WebComfort): functionalities added to WebComfort (such as an additional toolkit) can easily be integrated with WebC-Docs (e.g., using forums), while "built-from-scratch" systems must have such functionality created/adapted to their specific platform.

5 CONCLUSION

The expansion of the Internet has originated many CMS and ECM systems that aim to facilitate the management and publication of digital contents. However, most CMS platforms still do not offer functionality of a more complex nature (such as document management), while ECM platforms tend to address such complex functionality but without taking advantage of the possibilities inherent to a CMS platform.

In this paper we have presented WebC-Docs, a document management system for the WebComfort CMS platform. Besides typical file storage functionality (which can be found in many regular CMS installations), this system also provides features such as: (1) specifying customizable metadata for each document and/or folder; (2) indexing and searching documents using a powerful search-engine based on Lucene.NET, or even using additional search-engines according to the searched folders; and (3) specifying fine-grained permissions for each document and folder, in a way inspired by typical ACL mechanisms.

For future work, we plan to introduce additional extensibility and configuration points to the system, in order to improve its adaptability to different organizational contexts. Among others, we intend to add an explicit configuration point to the system regarding the document indexers that are installed/used in the system; this will allow us to explicitly configure which indexers to use for each installation.

Another aspect to address is the integration with other document management systems (e.g., DSpace). Although WebC-Docs supports the usage of additional data-stores (on a folder basis), connectors to such data-stores have not yet been developed (only for local data-stores, such as the local file-system and a local relational database). Also, we intend to add a web-services-based interface to WebC-Docs, allowing other systems to interact with it without explicitly requiring the usage of its web-based interface.

Finally, another important topic is the specification of document management workflows. Currently, WebC-Docs only allows the management of Documents and Folders, according to the current user's permissions. It would be desirable to specify the various steps of a workflow, in order to adapt WebC-Docs to more complex application scenarios.

ACKNOWLEDGEMENTS

The authors would like to thank the members of EDM (Empresa de Desenvolvimento Mineiro, S.A.), as well as all the other testers of WebC-Docs, for all their hard work, both in assisting with the testing of WebC-Docs in a real organizational environment, and in providing very helpful suggestions for the system.

REFERENCES

AIIM (2009). Association for Information and Image Management. Retrieved March 17, 2009 from http://www.aiim.org.

Atkinson, C. and Kühne, T. (2003). Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):36–41. Retrieved March 21, 2009 from http://doi.ieeecomputersociety.org/10.1109/MS.2003.1231149.

Boiko, B. (2001). *Content Management Bible*. John Wiley & Sons, Hoboken, New Jersey, U.S.A.

Carmo, J. L. V. d. (2006). Web Content Management Systems: Experiences and Evaluations with the WebComfort Framework. Master's thesis, Instituto Superior Técnico, Portugal.

CSTools (2009). Malcolm Crowe's Home Page (CSTools). Retrieved March 21, 2009 from http://cis.paisley.ac.uk/crow-ci0/.

DSpace (2009). DSpace.org. Retrieved March 22, 2009 from http://www.dspace.org.

Kampffmeyer, U. (2006). ECM – Enterprise Content Management. Retrieved March 17, 2009 from http://www.project-consult.net/Files/ECM_WhitePaper_kff_2006.pdf.

KnowledgeTree (2009). KnowledgeTree Document Management System. Retrieved March 22, 2009 from http://www.knowledgetree.com.

LuceneNet (2009). Lucene.Net. Retrieved March 21, 2009 from http://incubator.apache.org/lucene.net/.

OMG (2005). Object Management Group – Unified Modeling Language: Superstructure – Specification Version 2.0. Retrieved March 21, 2009 from http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf.

OpenDocMan (2009). OpenDocMan – Free Document Management Software DMS. Retrieved March 22, 2009 from http://www.opendocman.com.

Rockley, A. (2002). *Managing Enterprise Content: A Unified Content Strategy (VOICES)*. New Riders Press.

Saraiva, J. d. S. and Silva, A. R. d. (2008). The WebComfort Framework: An Extensible Platform for the Development of Web Applications. In IEEE Computer Society, editor, *Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2008)*, pages 19–26.

WebComfortOrg (2009). WebComfort.org. Retrieved June 8, 2009 from http://www.webcomfort.org.