Design Issues for an Extensible CMS-Based Document Management System

João de Sousa Saraiva and Alberto Rodrigues da Silva

INESC-ID & SIQuant
Rua Alves Redol, n°9, 1000-029 Lisboa, Portugal
joao.saraiva@inesc-id.pt, alberto.silva@acm.org

Abstract. Content Management Systems (CMS) are usually considered as important software platforms for the creation and maintenance of organizational web sites and intranets. Nevertheless, a simple CMS alone typically does not provide enough support for an organization's more complex requirements, such as document management and storage. More specifically, such requirements usually present a set of design and implementation issues, which need to be addressed in an extensible manner if the system is to be maintained and evolved over time.

This paper presents the design issues that are subjacent to the architecture of WebC-Docs, a highly-customizable and extensible CMS-based web-application that provides document management functionality. Because of this degree of extensibility, the WebC-Docs toolkit can be configured and used in various kinds of scenarios.

Keywords: Document management system, Extensibility, Content management system, WebComfort.

1 Introduction

The worldwide expansion of the Internet in the last years has led to the appearance of many web-oriented CMS (Content Management Systems) [21,4] and ECM (Enterprise Content Management) [1,15,9] 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 [3,17]. These systems typically present some 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 that are oriented towards using Internet-based technologies and workflows to capture, manage, store, preserve, and deliver content and documents in the context of organizational processes [1]. Nevertheless, these two content-management areas are not disjoint [10]. In fact, 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, with problems such as: (1) no verification for duplicate information, (2) no grouping of documents according to a certain logical structure, and (3) no possibility of providing metadata for each document.

A. Fred et al. (Eds.): IC3K 2009, CCIS 128, pp. 323–336, 2011.

Into this context comes the WebC-Docs system. WebC-Docs is a document management toolkit for the WebComfort CMS platform [18], that provides a large set of configuration and extension points, which allows the system to be used in a wide variety of application scenarios.

In this paper we present the architecture of WebC-Docs and discuss its major technical contributions. This paper is structured in five main sections. Section 1 introduces the role of CMS and ECM systems as support platforms for web-applications, as well as the classical problem of document management in an organizational context. Section 2 presents the architectural aspects of the WebC-Docs system. Section 3 provides a brief discussion of this system. Section 4 presents related work that we consider relevant for this project. Finally, section 5 presents the conclusions for this project so far, as well as future work.

2 WebC-Docs

WebC-Docs [19] is a document management toolkit for the WebComfort CMS [17,18] with a component-based architecture, making it easily extensible, highly configurable, and adequate for several kinds of application scenarios.

This section presents some of the components and aspects of WebC-Docs' architecture, namely: (1) its key concepts; (2) its integration with the WebComfort CMS platform; (3) its Explorer-like web interface; and (4) some more technical features, such as: (i) the Document Versioning; (ii) the Dynamic Attributes; (iii) the Indexing and Searching; (iv) the Permissions mechanism; and (v) the facilities for using additional repositories to store and locate documents.

2.1 WebC-Docs' Key Concepts

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

`Document` is the main entity; in a typical 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", without the digital files of 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 "pointers" to regular `Documents`. Thus, although a `Document` must be located in exactly one `Folder`, it is possible for the `Document` 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 a WebComfort role can perform over them (this mechanism is explained further down this section).

2.2 Integration with the WebComfort CMS

One of the important innovations in WebC-Docs is its integration with a CMS platform, in this case WebComfort [17,18,20]. This integration consists mainly of the following

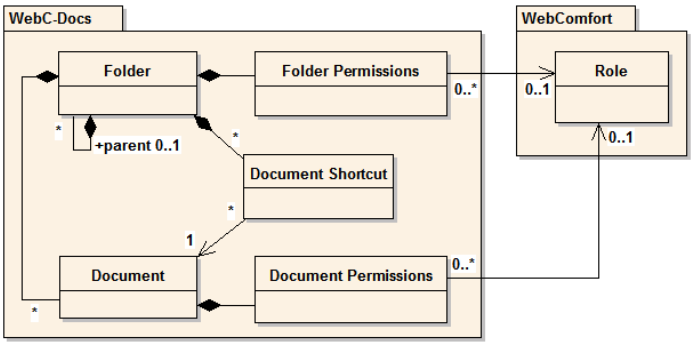

Fig. 1. The main concepts of the WebC-Docs system

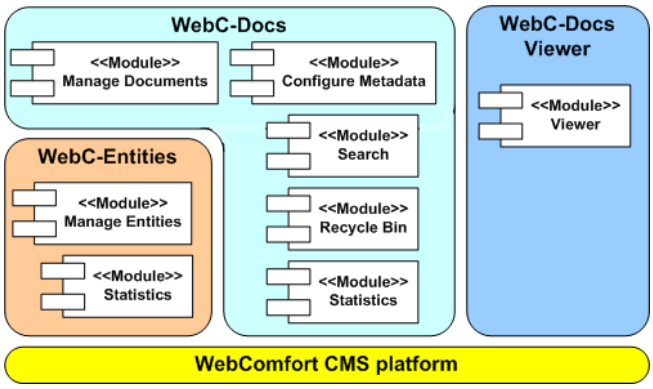

Fig. 2. An overview of WebC-Docs' integration with WebComfort

points: (1) the user's main interaction points with WebC-Docs are implemented as WebComfort modules (e.g., Document Management, Statistics, Configuration) which 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; and (3) WebC-Docs is distributed as a WebComfort toolkit, and so it can installed on any regular WebComfort installation in a simple and automatic manner. Figure 2 shows an overview of WebC-Docs' integration with WebComfort and other toolkits.

WebC-Docs provides a number of WebComfort modules that can be installed in WebComfort tabs (also called Dynamic Pages). These modules are the starting point for each of the system's use-cases. Subsequent steps – such as the visualization of the selected Document's Details – in those use-cases are typically handled by specific WebComfort Pages (not represented in Figure 2 for simplicity), by other WebC-Docs modules (when a Page would not make sense, such as deleting a Document and subsequently showing it in the Recycle Bin), or even by the module itself (when the use-case is very simple, such as moving a set of Documents to a different Folder).

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

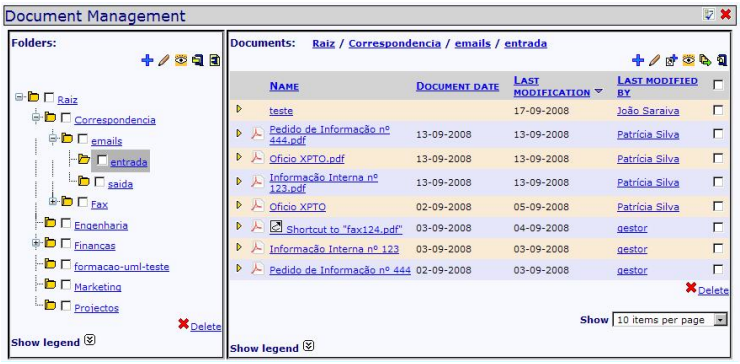

Fig. 3. The main WebC-Docs interface: folders and documents

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, in which each tab contains the document management modules that are adequate for the user's responsibilities within the portal). For a description of WebComfort and its features, we recommend the reading of [17].

2.3 Explorer-Like Web Interface

One of WebC-Docs' main objectives 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 the one of 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 taking into consideration the current user's permissions), while the right side shows the documents contained within the selected folder.

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 can be shown by clicking on the document's name.

WebC-Docs also provides a 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's contents and purge it, or restore documents that have been deleted by mistake. The usage of this Recycle Bin can be disabled, although this is generally not recommended (because deleting a document would become an irreversible action).

2.4 Document Versioning

Document versioning is a fundamental aspect also addressed by WebC-Docs, making it possible to revert a Document back to the state in which it was at a certain point in time.

Each modification (e.g., a change to its description) does not really *alter* the document; instead, a new entry is created containing the new information, and the new file (if any) is stored in the Document's repository.

Reverting to a previous version will not erase all versions since then: a new Document version will be created, which will be identical to the version to which we are reverting. Thus, if a user later decides that the intended version V2 was one that was produced later than the version V1 (in which V2 = V1 + a number of versions) to which we reverted, reverting to that version V2 is still possible.

Thus, a simple Document ID is typically not sufficient to be able to view and/or obtain a certain Document in WebC-Docs: a version number is also necessary. Nevertheless, to simplify the user's interaction with the system (as well as the referral to Documents via regular WWW links and bookmarks), if only a Document ID is specified, WebC-Docs automatically assumes that the user is referring to the latest version of the Document.

Document Versioning currently covers only Documents, their contents and their regular metadata (e.g., name, description, authors). Dynamic Attributes are not yet covered by this feature, but we will address this in the near future.

WebC-Docs offers Document Versioning as a configurable option, enabled by default. However, this option can be disabled if the Administrator so wishes (for reasons such as limited storage space on the document file repository, or no need to be able to revert documents to previous versions). Even if Versioning is disabled, the Document's history (dates of modifications, change-related comments) can still be viewed, as it is independent of Versioning.

2.5 Dynamic Attributes

One of the most powerful features of WebC-Docs is the possibility of specifying metadata at runtime, via the web-based interface, in a customizable manner that can be adjusted to the organization's document information requirements. This mechanism, which we designate as **Dynamic Attributes**, is based on the notions of `Attribute Set`, `Attribute Definition`, and `Attribute Type`, illustrated in Figure 4.

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

Finally, this mechanism can be applied to Documents and Folders by using `Attribute Sets` and `Attributes`. A user can apply a `Attribute Set` to any Document or Folder, any number of times (e.g., to provide various author contacts for a certain Document). `Attributes` are simply used to store the values that are provided by the user.

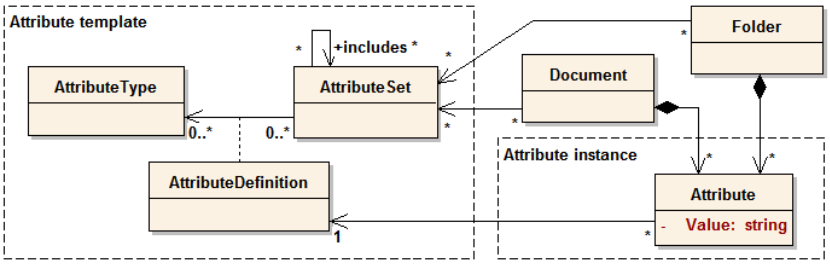

Fig. 4. The main concepts of the "Dynamic Attributes" mechanism

WebC-Docs currently provides out-of-the-box Dynamic Attributes for basic data-types (e.g., integers, strings) and enumerations (sets of strings to be used in selection lists). However, users with the appropriate permissions (e.g., Portal Administrator) can install new Dynamic Attributes by using the mechanism described above. An immediate advantage of this is that it enables the usage of any kind of attribute (e.g., a GIS-oriented Dynamic Attribute that displays a coordinate in a map, obtained via Google Maps).

2.6 Indexing and Searching

For a document management system to be of any practical 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 WebComfort module with search functionality, using the Lucene.Net indexing engine [12], 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 – to refine the search and reduce the number of false positives.

Figure 5 presents the main components of WebC-Docs' indexing. Whenever a Document is created/altered, it is supplied as input to the `Indexer Front Controller`, which is responsible for quickly analyzing the Document (namely its MIME type and file extension) and forwarding it to the adequate `Base Content Indexer` objects (if any). Those objects will then parse the Document's file contents and invoke WebC-Docs' internal API (which, in turn, also provides a wrapper around the Lucene.Net functionality). Document metadata (regular document attributes and dynamic attributes) is always indexed, so the Document can still be found in the system, even if it has no file whatsoever.

The Searching mechanism itself is based on the "Pipes and Filters" design pattern [8]. Each segment/step of a search (filtering, searching) is performed by a single `Search Data Filter`; those search filters are combined into a "search pipeline", at the end of which the search's results are obtained. Although at first sight this approach may appear unnecessary (as the Regular Search actually uses only a single filter, and the Advanced Search uses two filters), the intent was to support new types of search in the

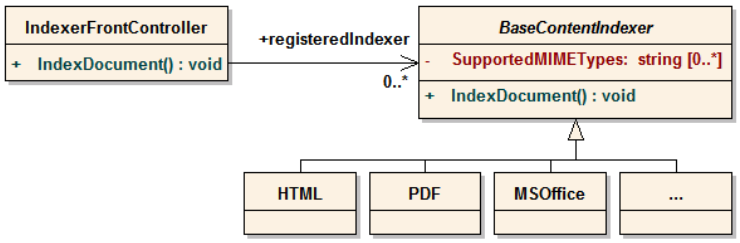

Fig. 5. Indexing can be performed over many file formats

future, likely with user-defined types of search (which, in turn, could be a composition of existing search filters) adjusted for the organization's information needs.

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.

Although WebC-Docs' 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 Google's own search engine), and uses a search term compiler built using CSTools [5]. 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, and use that query to find documents in 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 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 would remain the same.

2.7 Permissions

Permissions specification, 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 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 improve the system's usability, permissions are inherited (from parent folder to child folder/document) by default. For each of the roles defined in the WebComfort installation, a large variety of permission options can be configured (specifying permissions for anonymous users is also allowed, if the organization wishes to make some information publicly available).

The system can be configured to interpret permissions using either a **optimistic** or **strict** perspective. 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. This option allows the system to be easily adapted both to: (1) organizations that deal mainly with internal classified information (i.e., information that should only available to a few select users) and wish to enforce strict security validations regarding document access; and (2) organizations that intend to release most of their information to the public domain (and only some documents, such as works-in-progress, should not be publicly available).

Additionally, and 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-related operations (including modifications) go through it, removing the need to check for old or invalid entries.

2.8 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 repository already exists (e.g., a DSpace repository [6] in an academic context) and should be used by WebC-Docs. All of WebC-Docs' interactions with any repository are done through either `Base Folder Repository` or `Base Document Access` objects, shown in Figure 6.

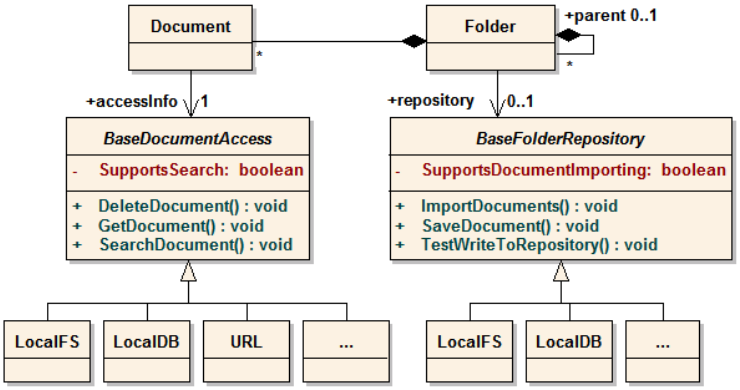

Fig. 6. Support for multiple kinds of repositories

The main reason why repository access has been divided into those two classes, instead of using a single class containing repository access methods, is because all the information needed to access a document's file, or a folder's 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 (that uses a repository R1), is later moved to another folder F2 (which uses a different repository, R2). This avoids moving files between repositories, which could itself present additional issues (e.g., the need to recover if a temporary repository failure is detected). This is also 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 Discussion

Although we have presented some of WebC-Docs' important aspects, there are some key issues that should be mentioned. This section presents a brief discussion of such issues, as well as a few additional notes regarding the 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: (1) 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; and (2) all document creation and modification operations are recorded in WebC-Docs' own historic records, which are not modifiable and can be viewed in the Document's details by anyone that has the required permissions.

WebC-Docs supports the specification of a Document's Authors and Editors. This is done by means of 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 an author of this document", instead of just supplying a string of characters (which usually leads to name variations).

Bootstrapping is a typical problem in any system that involves setting permissions for resources; in WebC-Docs' case, no role has permissions to any Document or Folder, by default. The bootstrapping problem here is in ensuring that, although no particular role has access to a Document or Folder (by default), it is possible to configure Folders and Documents with non-default settings. This has been handled by making the CMS Administrator role (which *always* exists, and should be 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 for the Dynamic Attributes mechanism, 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 [13]; and (2) regarding the "Attribute instance" section, it is not a case of linguistic instantiation, but rather ontological instantiation [2].

It is important to mention that WebC-Docs has already been validated in a number of case studies, namely: (1) the SIQuant [20] and WebComfort.Org [18] web-sites,

which were already using WebComfort, are now using the WebC-Docs toolkit to make available a variety of documents, such as white-papers and user manuals); (2) our own research group's document repository uses WebC-Docs to make our scientific production publicly available to the community [7]; and (3) WebC-Docs is currently being used by a Portuguese real-estate-related company, to organize and catalog their vast paper-based documentation archive.

Finally, it should be noted that, although WebC-Docs has not yet been integrated with other systems (e.g., DSpace [6]), it allows external sources to access search results by means of RSS [16]: 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 (a regular reader such as those included with Microsoft Internet Explorer or Mozilla Firefox 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 already exist which handle some of the issues presented in this paper. In this section, we present some of which we consider most representative of this area, while making a comparison between those systems and WebC-Docs.

OpenDocMan [14] is a free document management system, distributed under the open-source GPL license, that is designed to comply with the ISO 17025 and OIE standard for document management [14]. 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., map coordinates) through its Dynamic Attributes mechanism.

DSpace [6] is an "open-source solution for accessing, managing, and preserving scholarly works" [6], designed to be as standards-compliant as possible. DSpace supports a tree-hierarchy, consisting of: (1) communities; (2) collections; (3) items; and (4) files. Although this hierarchy is fixed and cannot be altered, the user-interface can be adapted to "mask" some aspects of that hierarchy (e.g., to show a community as an aggregation of similar collections). However, this is a limitation that can make DSpace unsuitable for some particular (non-academic) cases – although it should be noted that 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 also takes advantage of the available

metadata (provided with each submitted resource) to provide more accurate search results. It should be noted that, unlike WebC-Docs, some kinds of changes to 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. Additionally, both contents and metadata can be exported at any time, using a XML-encoded file format. It also uses the Handle system [6] to provide unique identifiers for each managed resource, ensuring that document identifiers will be valid for a very long time.

KnowledgeTree's document management system [11] features 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), enabling 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 allows integration with Microsoft Office applications (Outlook, Word, Excel, PowerPoint). On the server, it can index various file types, such as Microsoft Office, PDF, XML, HTML and text.

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

Its metadata-handling mechanism is also very powerful. It allows metadata of various types, which can be marked as "required"; nevertheless, WebC-Docs' Dynamic Attributes mechanism can also easily support these features. It also allows searching over metadata and over documents' 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 (because 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 `Documents`, on a `Folder` basis. Additionally, it allows the usage of tags and tag clouds; WebC-Docs supports tags, but still only at a fairly basic level, because they can be easily replaced by Dynamic Attributes.

The system addresses document and metadata versioning, which allows a document to be reverted back to a previous point in time. As previously mentioned, WebC-Docs also supports document versioning, but only for regular metadata (e.g., authors, comments); Dynamic Attributes are not versioned yet.

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

5 Conclusions and Future Work

The recent expansion of the Internet has originated many CMS and ECM systems that aim to facilitate the management and publication of digital contents. These platforms, which tend to be modular, extensible and versatile, can be used as support web applications for the dynamic management of web sites and respective contents. Nevertheless, 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 that can be provided by 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 typical CMS installations), this system also provides features such as: (1) specifying customizable metadata for each document and/or folder; (2) the indexing and searching of documents using a powerful search engine, or even using additional search engines according to the folders where the search should occur; and (3) specifying fine-grained permissions for each document and folder, in a way inspired by traditional ACL mechanisms.

As 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. One of our priorities is adding 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 (e.g., if an organization wishes to index PDF documents using a particular PDF reading mechanism, then adding such a mechanism would be a matter of adding it through WebC-Docs' web-based interface).

Another aspect to further address is the integration with other document management systems, like DSpace. Although WebC-Docs supports the usage of additional data-stores, connectors have not yet been developed (only for local data-stores, such as the local file-system and a local Microsoft SQL Server database). Also, we intend to define a web-services interface, to allow other systems to interact with WebC-Docs without requiring its web-based interface.

Semantic relationships between documents is another issue that we intend to address. Currently, WebC-Docs does not allow "associating" documents with a certain relationship. Although this issue can be overcome by using folders and document shortcuts, it would be preferable to support user-defined document relationships (e.g., "document D1 is the result of the proposal in document D2"). This would also affect the search functionality, as document relationships could be analyzed in order to obtain potential search results.

Finally, an important topic that will be addressed in the very near future 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 to more complex application scenarios.

Acknowledgements. The authors would like to thank the members of EDM (Empresa de Desenvolvimento Mineiro, S.A.) 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 regarding the system.

References

1. Association for information and image management (March 17, 2009), http://www.aiim.org
2. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Software 20(5), 36–41 (2009), http://doi.ieeecomputersociety.org/10.1109/MS.2003.1231149
3. Carmo, J.L.V.d.: Web Content Management Systems: Experiences and Evaluations with the WebComfort Framework. Master's thesis, Instituto Superior Técnico, Portugal
4. The CMS Matrix, http://www.cmsmatrix.org (retrieved December 9, 2009)
5. CSTools: Malcolm Crowe's Home Page (CSTools) (March 21, 2009), http://cis.paisley.ac.uk/crow-ci0/
6. DSpace: DSpace.org (March 22, 2009), http://www.dspace.org
7. GSI-Documents: INESC-ID, Information Systems Group (GSI) – Documents, http://isg.inesc-id.pt/gsidocs (retrieved March 21, 2009)
8. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, Reading (2003)
9. Jenkins, T.: Enterprise Content Management Solutions: What You Need to Know. Open Text Corporation (April 2005)
10. Kampffmeyer, U.: ECM – Enterprise Content Management (March 17, 2009), http://www.projectconsult.net/Files/ECM_WhitePaper_kff_2006.pdf
11. KnowledgeTree: KnowledgeTree Document Management System (March 22, 2009), http://www.knowledgetree.com
12. LuceneNet: Lucene.Net (March 21, 2009), http://incubator.apache.org/lucene.net/
13. OMG: Object Management Group – Unified Modeling Language: Superstructure – Specification Version 2.0. (March 21, 2009), http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf
14. OpenDocMan: OpenDocMan – Free Document Management Software DMS (March 22, 2009), http://www.opendocman.com
15. Rockley, A.: Managing Enterprise Content: A Unified Content Strategy (VOICES). New Riders Press, Indianapolis
16. RSS: Really Simple Sindication (RSS) 2.0 Specification, http://blogs.law.harvard.edu/tech/rss (retrieved March 22, 2009)
17. Saraiva, J.d.S., Silva, A.R.d.: The WebComfort Framework: An Extensible Platform for the Development of Web Applications. In: IEEE Computer Society (ed.) Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2008), pp. 19–26 (2008)

18. SIQuant: WebComfort.org,
`http://www.webcomfort.org` (retrieved December 9, 2009)
19. SIQuant: WebComfort.org – WebC-Docs,
`http://www.webcomfort.org/WebCDocs` (retrieved December 9, 2009)
20. SIQuant – Engenharia do Território e Sistemas de Informação,
`http://www.siquant.pt` (retrieved December 9, 2009)
21. Suh, P., Addey, D., Thiemecke, D., Ellis, J.: Content Management Systems (Tools of the Trade). Glasshaus (October 2003)