Using Text Editing Creation Time Meta Data for Document Management

Thomas B. Hodel, Roger Hacmac, and Klaus R. Dittrich

University of Zürich, Department of Informatics,
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
{hodel, dittrich}@ifi.unizh.ch, hacmac@gmx.ch

Abstract. Word processing systems ignore the fact that the history of a text document contains crucial information for its management. In this paper, we present database-based word processing, focusing on the incorporated document management system. During the creation process of a document, meta data are being gathered. This information is generated on the level of the whole document, on sections of a document or even on individual characters and is used for advanced retrieval by so-called dynamic folders, which are superior to advanced hierarchical file systems.

1 Introduction

Text data (documents) are not treated as valuable data (as opposed to business data like customer, product, finance, etc.) even though a lot of companies' knowledge is stored within this structure. For a large-scale document management environment, local copies of remote data sources are often made. However, it is often difficult to monitor the sources in order to check for changes and to download data items to the copies. In many cases, text documents are stored somewhere within a confusing file structure with an inscrutable hierarchy and low security. On the other hand, data, which from an organization's point of view can be classified as crucial, is stored in databases. Here, the infrastructure and the data are highly secure, multi-user capable and available to several other tools for compiling reports, content and knowledge. Reporting and query tools can be specifically defined and applied to such data. Our idea is to make use of such a philosophy for documents. We therefore strive for the native storage of texts in a database.

Most users organize their documents by location in hierarchies onto which they map their own semantic structures. More generally speaking, hierarchies pervade document and information storage systems. The fundamental organizing principle for text documents is based on locations with the restriction of only being able to appear in one location at a time. This forces users to create strict categorizations of documents and organization. In line with these considerations, the choice of how to store and search a document has to be redefined. Users need functionality to store and locate documents without having to specify a location, and without referring to a fixed hierarchy. As a consequence, we propose that text editing creation time meta data to be automatically generated and stored (see part 2.1), which enables a

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 105 – 118, 2005.

sophisticated document management system (see part 2.2) and at the same time enriches text mining functionalities.

Until now, document management and text mining solutions ignored the fact that the history of the creation process of a text document could contain crucial information. Our database-based word processing application supports not only editing but also fine-grained security, versioning, business processes, text structure, layouting, data lineage, and multi-channel publishing - all within a collaborative, real-time and multi-user environment. All of this data, along with every alteration ever made to it since its creation (especially during editing), is therefore captured [6].

In this paper, we focus on our enhanced document management system. For documents, innumerable document management systems exist [6]. According to our information no document management system has integrated automatically generated text editing creation time meta data. Realizing such a document management system involves several aspects. First of all, the word processing application has to be designed in such a way that it is able to capture all such data. This paper presents the text editing creation time meta data concept and prototype for our database-based collaborative editor. With the help of some query examples, we then demonstrate our system; the result is a completely virtualized management of text documents. The implementation of 'dynamic folders' (see part 3) provides the user with an unlimited possibility of views on documents stored in the system.

1.1 Problem Description

With the increasing amount of documents produced it becomes more and more important to find a way of organizing the created documents in an efficient way, so that they can easily be found when required. Currently it is easier to find and access a file created by a kid in New Zealand, than to access a file created on your colleague's desktop [9]. Crucial information contained in the large number of documents in any company or organization is at risk of being doomed to become unachievable.

The emerging market of document management and text mining systems underlines the need for tools which can manage documents in a more sophisticated way than the file system does. These systems, such as 'Documentum', 'FileNet', 'OpenText', 'Autonomy', 'SAS Text Miner', and 'Thunderstone', just to mention a few of them, have a very similar philosophy. First, they import documents, which means that the system either stores the document as it would be stored in a file system (this is the usual way), or it stores it within the database as BLOB (this is the exception). Some systems can even use both methods. Secondly, the tool analyzes the content of the document and creates a full text index. This index is normally stored within the file system.

However, current document management system solutions do not solve the problem of how to organize documents. They index the documents and can generate a response of matching documents for a specific user query using these indexes, but the underlying data organizational structure is not improved.

Even Microsoft has meanwhile recognized the necessity to revise the old-fashioned hierarchical file system with its inherent limitations. In its new Windows Version 'Longhorn', Microsoft will extend the current file system NTFS with the component 'Windows Future Storage' (Win FS) to enable access to the file system in a relational

way, as known from standard SQL databases.1 The difference between Win FS and our dynamic folder concept is that we use primarily automated created metadata and not like Win FS the content of a document. Based on our knowledge, TeNDaX is the only existing database based editor and is the only system which is able to create and store all these metadata.

1.2 Related Work

Several papers have been written emphasizing different aspects of how to leverage document management. Some of them concentrate on possible improvements in the way that meta data could be gained from documents, while others propose new ways of organizing documents, in contrast to file systems. These are also the two main aspects which will guide us through our paper. To the best of our knowledge, no previous research paper proposes to use meta data in the way we do nor our method of organizing it. Next, some of those research papers shall be summarized.

Placeless Documents: The "Placeless Documents" project with its prototype "Presto" addresses the way in which documents are organized. Meta data can be assigned to every document, either manually by users or automatically by applications. The meta data of each document can then be used to create "fluid collections": these are special folders, which automatically include or exclude documents, based on the meta data specified for the folder [2].

KnownSpace[2]: KnownSpace is an open, programmable, computational environment, suitable for arbitrary data management applications so that anyone can create anything. Small, independent programs (called simpletons) are loosely coupled with the data (called entities) and with each other. Programmers can dynamically attach arbitrary computations to arbitrary data. Simpletons parse this unstructured data, building an object-oriented database on it. The frontend may have many faces and is plugged on the KnownSpace kernel.

Lifestreams: A lifestream is a time-ordered stream of documents that acts as a diary of your electronic life; every document you create or receive is stored in your lifestream. The tail of your stream contains documents from the past, whereas the head of it contains more recent documents [3].

TimeScape: A user of TimeScape can spatially arrange information on the desktop. Any desktop item can be removed at any time, and the system supports time travel to the past (to restore desktops) and to the future (to schedule). This allows users to organize and archive electronic information without being bothered by document folders or file classification problems [10].

Semantic File System: A semantic file system is an information storage system that provides flexible associative access to the system's contents by automatically extracting attributes from files with file type specific "transducers". The automatic indexing of files and directories is called 'semantic' because of the use of user programmable transducers, which 'understand' the documents. A semantic file system

[1] http://longhorn.msdn.microsoft.com/

[2] http://hydrogen.knownspace.org

integrates associative access into a tree structured file system via the concept of a virtual directory. These virtual directories are interpreted as queries [4].

Looking at the process of creating meta data, none of these approaches above takes advantage of the way in which documents are created.

Many interesting approaches regarding document organization are described in the papers mentioned above. If, for example, we focus on the Placeless Documents project, fluid collections are used to organize documents. Folders do not contain documents which have been created or moved there, but rather documents whose meta data correspond with the one specified for the folder. This is a very desirable quality for the retrieval of documents, but also has a disadvantage since the content of a folder can change from one second to another. Explicit lists, which the user can specify for a folder, indicate which documents should be included or excluded in a specific folder. This somewhat alleviates the uncomfortable situation of the quickly changing content of folders.

1.3 Underlying Concepts

The concept of our meta data document management system requires an appropriate architectural foundation. Our concept and implementation are based on the TeNDaX [6] collaborative database based editing system.

TeNDaX is a Text Native Database eXtension. It enables the storage of text in databases in a native form so that editing text is finally represented as transactions. Under the term 'text editing' we understand the following: writing and deleting text (characters), copying & pasting text, defining text layout & structure, inserting notes, setting access rights, defining business processes, inserting tables, pictures, and so on i.e. all the actions regularly carried out by word processing users. With 'real-time transaction' we mean that editing text (e.g. writing a character/word, setting the font for a paragraph, or pasting a section of text) invokes one or several database transactions so that everything which is typed appears within the editor as soon as these objects are stored persistently. Instead of creating files and storing them in a file system, the content and all of the meta data belonging to the documents is stored in a special way in the database, which enables very fast real-time transactions for all editing taks [7].

The database schema and the above-mentioned transactions are created in such a way that everything can be done within a multi-user environment, as is usual done database technology. As a consequence, many of the achievements (with respect to data organization and querying, recovery, integrity and security enforcement, multi-user operation, distribution management, uniform tool access, etc.) are now, by means of this approach, also available for word processing. TeNDaX creates an extension of DBMS to manage text. This addition is carried out 'cleanly' and the responding data type represents a 'first-class citizen' [1] of a DBMS (e.g. integers, character strings, etc.).

2 The TeNDaX Document Management Approach

We use the following terminology: whenever the term '**meta data**' is used here, it refers to the text editing creation time data. A '**text editing**' process is a logical entity that represents an editing action on a single character, on a section of a document or on a whole

document. An '**editing action**' is a task a user can do within a word processing application such as insert (write, paste), delete and change a character or characters [6], define structure, security, version, business processes, layout, insert notes, and so on [5]. Under '**creation time**', we understand the date, time, author, roles and specific 'editing action' information. All of these ('text editing', 'editing action' and 'creation time' data) taken together represent the stored 'meta data'. One or more 'editing actions' are combined into a '**editing action type**'. These types can be accessed and interlinked, and as a consequence, can be used for document retrieval. A '**document'** is created through an arbitrary number of 'editing actions'. The combination of all 'editing actions' assigned to a certain document in a specific order defines a current document.

2.1 Collecting Text Editing Creation Time Meta Data

As mentioned above, every editing action invoked by a user in the TeNDaX system is immediately transferred to the database. At the same time, more information about the current transaction is gathered.

As all information is stored in the database, one character can hold a multitude of information, which can later be used for the retrieval of documents. Meta data collected at character level are: Author, roles, date and time, copy-paste references, local and global undo / redo, security settings, version and user defined properties.

Meta data can be gained from structure, template, layout, notes, security and business process definitions. Within each section plenty of information is stored: for example the workflow section [8] contains the business process element name, its category (content, format, structure and process decision), category types (edit, verify cation, comment, translate, and sign) process description, date of creation, author, processors (based on users and roles), due date, time and condition, specific notes, and access rights settings. Meta data collected on the level of a document section are: author, date and time, structure affiliation, template affiliation, layout, business process affiliation, security affiliation, note affiliation, version affiliation, local and global undo / redo and user defined properties.

Last but not least, on the level of the whole document, there is meta data to be gathered during its creation and editing. Meta data we collect at the document level are: Creator, roles, date and time, document object ID, document names, structure affiliation, note affiliation, security settings, size, authors, readers, state, places within static folders and user defined properties.

All of the above-mentioned meta data is crucial information for creating content and knowledge out of word processing documents. We need these meta date for different functions within our collaborative editor, like local and gloabal und / redo, version, data lineage, work flow, security collaborative writing, collaborative multidimensional structuring of text and knowledge management [5], [6], [7], [8]. The next subchapters show how this meta data can be used for document retrieval and how the TeNDaX document management system works.

2.2 The Usage of Text Editing Creation Time Meta Data for Document Retrieval

Using the meta data gained, the following example queries can be asked in the TeNDaX document management system. The concrete form of querying and presentation of results is discussed in the following chapters.

- Show all documents in workflows which have pending tasks for me, or which I have written and which have been rejected in a workflow by another user.
- Show all documents of which more than 50% was written by user “Dittrich” and which haven’t been modified since 1.1.2001.
- Show all documents to which I have write rights and which have been read by more than 100 users with the role “Employee”.
- Show all documents which have character security restrictions for the role “Employee”.
- Show all documents edited by team “A” and read by my boss last week?
- Show all documents which have been written by user “Hodel” or “Hacmac”, with creation date after 1.1.2000 and size more than 1000 characters.
- Show all the documents accessible by the role “Employee”, which have the document name “*proj*”.
- Show all documents written by myself, with similar sentences and phrases to the document “Project TeNDaX”.
- Show the document with ID “1230” and all documents with copy-paste references to this document, created by user “Hodel”.
- Show all documents which are somewhere in a directory called “TeNDaX” and which are marked by the user defined property “to be done”.
- Which documents were read or printed out by our project manager, and when?
- Who wrote this paragraph originally?
- Which parts have been copied and pasted, and from which source?
- As a final example, we can look at the following situation which could occur in a company: four documents containing relevant knowledge for an upcoming project are found. Based on the information about which part was written by which author, and which part was copied from another document, the system can pinpoint suitable employees and teams to discuss the new project.

2.3 Document Organization: Static and Dynamic Folders

The question arises as to whether this meta data can be used to create an alternative storage system for documents. As discussed in part 1.2, several papers have been written on how to improve document management. In any case, it is not an easy task to change users’ familiarity to the well known hierarchical file system.

This is also the main reason why we do not completely disregard the classical file system, but rather enhance it. Folders which correspond to the classical hierarchical file system, will be called “static folders”. Folders where the documents are organized according to meta data, will be called “dynamic folders”. As all information is stored in the database, the file system, too, is based on the database.

2.3.1 Static Folders

Static folders are folders as common. Users can create them, modify their names, delete, copy and move them and assign access rights if they are authorized to do so. Their main function is to store documents users create with the TeNDaX word processor.

Static folders are organized as follows:

- There is one private folder for each user which represents the user's private area. Under no circumstance can other users access this folder. The user may carry out any actions he wants to, e.g. create, modify, copy, move or delete folders and documents. A private folder is quite similar to stored files on a 'local disc'.
- For each role created, a public static folder is set up automatically by the system. A user is only able to see a public static folder if he was assigned to the corresponding role. The rights of the user in this case depend on the rights given to him by the administrator. Example: a new project is started and a new role (i.e. user group) is generated for this project. Next, all the people involved are assigned to this role, so that shared access to this public static folder is granted. These folders are quite similar to sets of files stored on a 'file-server' or within a document management system.

2.3.2 Dynamic Folders

The place where the meta data can be used for document retrieval, are the dynamic folders. The dynamic folders build up sub-trees, which are guided by the meta data selected by the user.

Thus, the first step in using a dynamic folder is the definition of how it should be built. For each level of a dynamic folder, exactly one meta data item is used to. The following example illustrates the steps which have to be taken in order to define a dynamic folder, and the meta data which should be used.

Table 1. Defining dynamic folders (example)

Level	Meta data	Restrictions	Granularity
1	*Creator*	Only show documents which have been created by the users "Hodel" or "Dittrich" or "Hacmac"	One folder per creator
2	*Business process affiliation*	Only show documents with closed tasks for the user group "Manager"	One folder per task status
3	*Business process affiliation*	Only show documents with tasks completed by the user "Meier"	One folder
4	*Authors*	Only show documents where at least 40% was written by user 'Hodel'	Each 20% one folder
5	*Structure affiliation*	Only show documents which have been assigned the template "LNI" or "LYNX"	One folder per template
6	*Copy-paste references*	Only show documents which have no copy-paste references to a document with the name "DKE TeNDaX".	One folder

As a first step, the meta data which will be used for the dynamic folder must be chosen. As we see in Figure 1, the sequence of the meta data influences the structure of the folder. Furthermore, for each meta data used, restrictions and granularity must be defined by the user; if no restrictions are defined, all accessible documents are listed. The granularity therefore influences the number of sub-folders which will be created for the partitioning of the documents. Figure 1 visualizes the dynamic folder defined in Table 1. We named this dynamic folder 'DF_Paper'.

As the user enters the tree structure of the dynamic folder, he can navigate through the branches to arrive at the documents he is looking for. The directory names indicate which meta data determines the content of the sub-folder in question. At each level, the documents, which have so far been found to match the meta data, can be inspected. This is symbolized in figure 1 by the little document icons below each folder. For the folders "Creator_Hodel" and "Creator_Hacmac", the same structure as that which is shown for the folder "Creator_Dittrich" would be constructed.

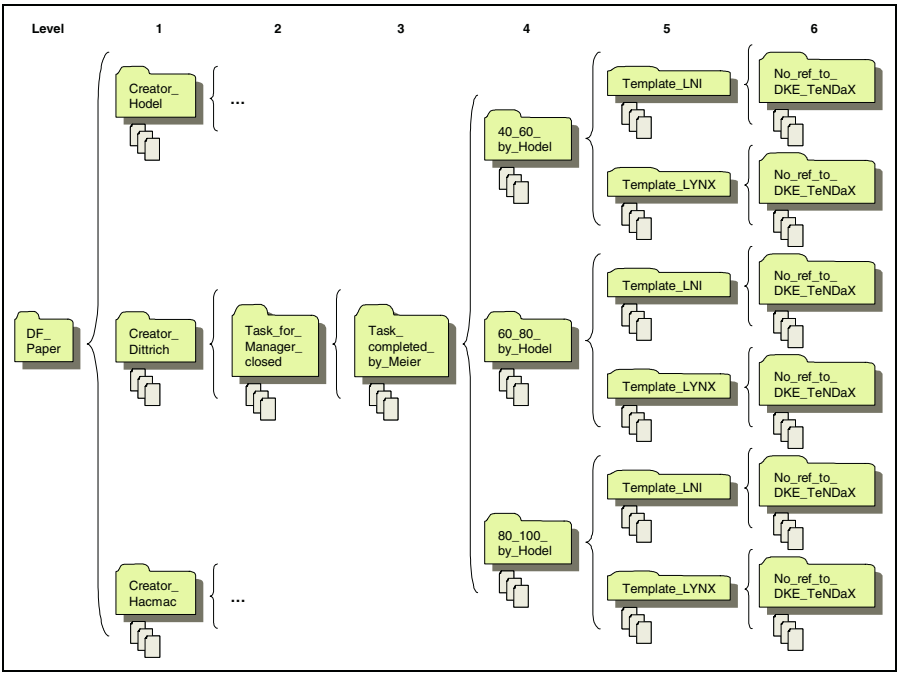

Fig. 1. Representation of a dynamic folder

Ad hoc changes of granularity and restrictions are possible in order to maximize search comfort for the user. It is possible to predefine dynamic folders for frequent use, as well as to create and modify dynamic folders on an ad hoc basis. Furthermore, the content of such dynamic folders can change from one second to another, depending on the changes made by other users at that moment.

3 Prototype

3.1 Collecting and Storing Creation Time Meta Data

Some of the meta data used for document management is deliberately gathered for this purpose only. On the other hand, some of it originates from other functionalities of the word processor. These functionalities are optimized for best performance. This

is the reason why the meta data for document management is not stored in a central area of the database, but is rather widely distributed. Here, the aspects of operational functioning and data warehousing are combined in the sense of a hybrid approach. Figure 2 shows which attributes are located in which database classes. The figure only shows the attributes relevant for document management; other attributes are omitted. The associations between the classes are not symbolized by lines, but instead with attributes in the classes of the type of the associated class.

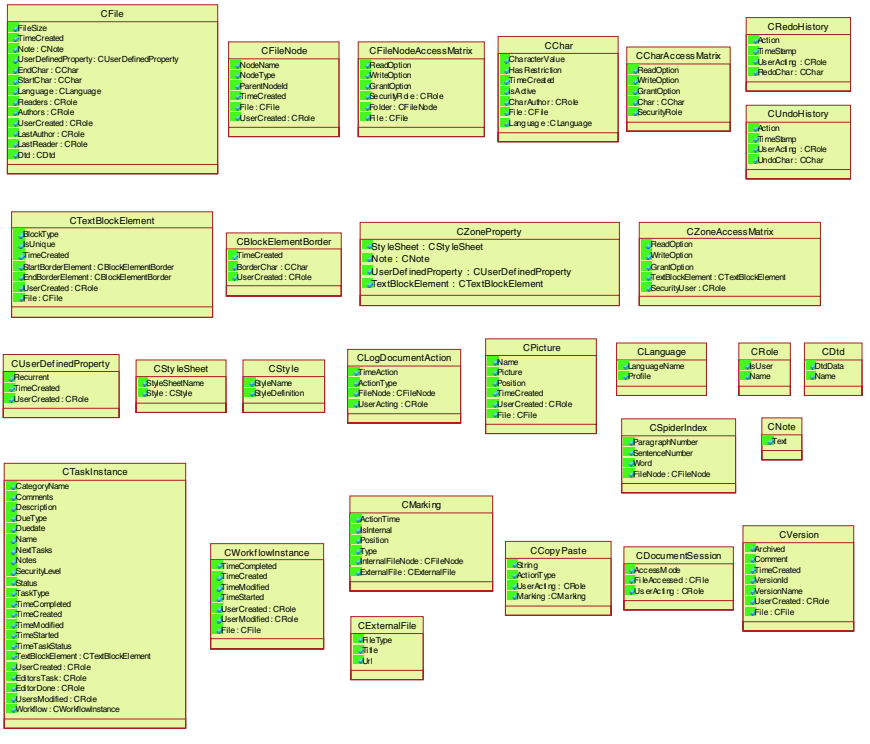

Fig. 2. Database schema showing only attributes relevant for document management

In part 2 we outline which meta data is held. The meaning of the attributes in the classes shown in the database diagram should be explanatory.

Obviously, the different attributes cannot all be of the same type nor can they have the same scopes. Thus for each attribute a separate method handles the scope and granularity. The next chapter shows how these meta data are handled finally in order to create the dynamic folders.

3.2 Generating the Dynamic Folders

As described above, each level of a dynamic folder represents a partitioning of the documents regarding some meta data, each level partitioning the previous one. Thus,

the first step a user must take to be able to use the dynamic folders, is to define the desired dynamic folders. This definition includes which meta data is decisive for which level of a dynamic folder, which conditions regarding this meta data the documents must accomplish, and with which granularity the sub-folders should be created. Remember: only one meta data can be chosen for each level of a dynamic folder. Figure 3 describes the process of opening the first level of a dynamic folder.

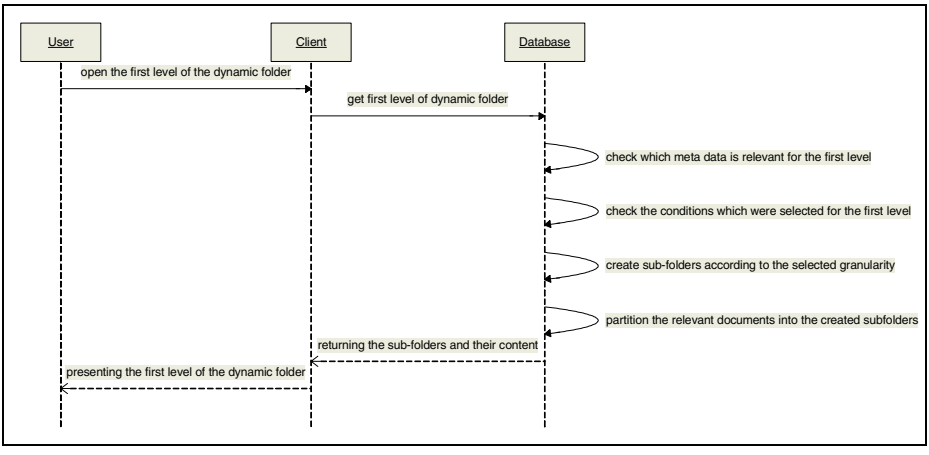

Fig. 3. Sequence diagram 'opening the first level of a dynamic folder'

When the first level of the dynamic folder is presented to the user, he can then choose to open the next level. The processing of this action is equal to the one shown in figure 3, with the difference that the documents which have now to be considered are only those from the sub-folder the user came from, and not from the whole document base.

The following lines describe how the meta data is used to create database queries, i.e. how the relevant documents are selected and filed in the sub-folders of the dynamic folder.

In the following description, the objects and functions used for creating the dynamic folders are portrayed. (The elementary functions are assumed to exist.)

The symbol *d* stands for the object "document".

$$d = document$$

The function *collect* collects all the documents available to and accessible by the acting user *user*. This includes checking the security restrictions, so that documents for which the user has no access rights are not included into the document base *db* for that specific user. The collection of documents *db* is therefore a set of documents *d*. The function *collect* is only called at the first level of a dynamic folder. The method used for the higher levels is described later on.

$$db = \{d_1, d_2, \ldots, d_n\} = collect(user)\ ;\ n = number\ of\ documents\ found$$

The symbol *m* stands for the meta data and symbolizes one of the selectable meta data, as described in part 2.1. For reasons of space, not all the meta data are listed here, but all of these are valid, regardless if they concern the character, section or document level.

$$m = \exists \{Author, Roles, \ldots, User\ defined\ properties\}$$

The function *validate* validates if the documents in the document base *db* accomplish the conditions which were specified for the meta data, which is decisive for the current level of the dynamic folder. The scope of validity is defined in the object *sc* and must correspond to the type of meta data it applies to. The result of the function *validate* is a reduced document base *db'*.

$$db' = validate(db, m, sc)$$

As the relevant documents are now isolated in *db'*, we need a function which builds up the sub-folders in the dynamic folder, as required by the granularity which the user has chosen. The function *partition* takes as its parameters the document base *db'* valid for the current user, the meta data *m* which is decisive for the current level, and the granularity *g* which decides upon the amount of sub-folders *sf* to be created. Also the granularity *g* must correspond to the type of meta data it applies to. The result of this function is a set of sets, i.e. documents filed correctly into the sub-folders.

$$\{sf_1, sf_2, \ldots, sf_x\} = partition(db', m, g)\ ;\ x = number\ of\ sub\text{-}folders\ created$$

$$sf = \{d_1, d_2, d_y\}\ ;\ y = number\ of\ documents\ filed\ into\ this\ sub\text{-}folder$$

The result of the function *partition* is passed to the client. As the user dives into one of the created sub-folders, the same procedure of validating and partitioning takes place. The input for the functions *validates* and *partition* is then:

$$db = sf_z\ ;\ z = sub\text{-}folder\ selected\ by\ the\ user,\ z = \exists \{1, 2, \ldots, x\}$$

$$m = meta\ data\ relevant\ for\ the\ next\ level\ of\ this\ dynamic\ folder,\ as\ defined\ by\ the\ user$$

$$sc = scope\ for\ the\ meta\ data\ m,\ as\ defined\ by\ the\ user$$

$$g = granularity\ for\ the\ meta\ data\ m,\ as\ defined\ by\ the\ user$$

If we consider the example from part 2.3.2, the function *validate* would be called with the following parameters for the first level of the dynamic folder:

$$db' = validate(db, \text{“Creator”}, \text{“Hodel or Dittrich or Hacmac”})$$

The corresponding SQL code is:

```
SELECT CFileNode.ID As fnID, CFile.UserCreated As Creator

FROM CFileNode INNER JOIN CFile ON (CFileNode.File =
CFile.ID) INNER JOIN CRole ON (CFile.UserCreated =
CRole.ID)

WHERE (CFileNode.IsDynamic = 0) AND ((CRole.Name =
"Hodel") OR (CRole.Name = "Dittrich") OR (CRole.Name =
```

```
"Hacmac"))
  IsDynamic = 0: static documents
```

For this purpose, the function *validate* uses the attribute *UserCreated* from the class *CFile* to evaluate the necessary meta data. The document base *db* is generated by the function *collect* with the help of the class *CFileNodeAccessMatrix*, which is responsible for all security issues on the document level. For the second level, *validate* is called as follows. *s* represents the sub-folder selected by the user.

db' = validate(sf_s, "Business Process Affiliation", "Documents with closed tasks to the user group 'Manager'")

The corresponding SQL code is:

```
SELECT CFileNode.ID As fnID

FROM CFileNode INNER JOIN CFile ON (CFileNode.File =
CFile.ID) INNER JOIN CWorkflowInstance ON (CFile.ID =
CWorkflowInstance.File) INNER JOIN CTaskInstance ON
(CWorkflowInstance.ID = CTaskInstance.Workflow) INNER
JOIN CRole ON (CTaskInstance.EditorsTask = CRole.ID)

WHERE (CFileNode.NodeType = 2) AND (CFileNode.IsDynamic
= 1) AND (CRole.Name = "Manager") AND
(CTaskInstance.Status = 1) AND (CFileNode.ParentNodeId
= [Node ID of selected dynamic sub-folder])

CTaskInstance.Status = 1: task closed
CFileNode.IsDynamic = 1: dynamic documents
```

In this case, the meta data used originates from the attribute *EditorsTask,* in the class *CTaskInstance*.

The steps discussed above would also apply to the next levels of the dynamic folder. For each meta data, the functions *validate* and *partition* provide specialized methods which use the necessary classes.

4 Conclusion

The dynamic folder concept is a high-level data model. People use a computer to communicate and to store and organize their personal data. Unfortunately, a computer does a relatively poor job of allowing users to organize the information so that it can be found later. When a user forgot exactly where he puts a file, it can take quite a while to find it again. In the worst case, the entire content of each disk has to be searched.

One reason why to find information on a computer is difficult, is because of the limited ability for the user to organize data. The hierarchical folder structure does not work well when a categorization of data in numerous ways is wanted. Therefore, the first problem is that lots of files have to be stored and no good way to categorize them is available. Another problem is that the same stuff is stored in multiple places in multiple formats. There are a number of problems with current approaches to data

storage, like: Multiple applications cannot share common data, the same information lives in multiple locations, separate copies of data become unsynchronized, and there are no notifications of data change.

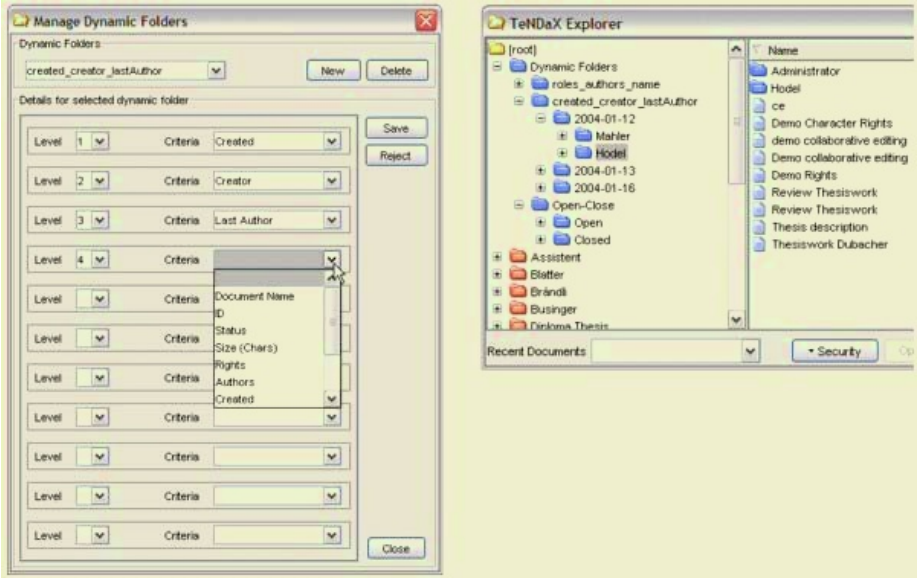

Fig. 4. TeNDaX screenshots

Dynamic folders improve text documents in three ways. First, they store automatically all thinkable meta data and relate one item of information to another. Second, it provides a common storage format for information collected. Third, it promotes data sharing of common information across multiple applications. Dynamic folder is an active storage platform for organizing, searching for, and sharing all kinds of information. This system defines a rich data model that allows using and defining data types that the storage platform can use. All these features together allow four ways to organize documents with dynamic folders: Hierarchical folder-based organization, item property–based organization, relationship-based organization, category-based organization.

TeNDaX reached the status of a quite stable prototype. The system is used in several pilot projects and within some courses at different universities. The described dynamic folders are implemented and running quite well (see Figure 4). Performance is similar to known file-explorer and therefore not an interesting issue. Integration into the Windows-Explorer was programmed too. Further information and a demonstration video clip can be found on the TeNDaX[3] website.

In this paper we have proposed a dynamic document management system environment that represents all documents in a database system, working with the

[3] http://www.tendax.net/

underlying TeNDaX architecture. This architecture enables different views of documents in a structured, reliable and real-time co-operation environment.

References

1. S. Abiteboul, R. Agrawal, P. Bernstein, M. Carey, S. Ceri, B. Croft, D. DeWitt, M. Franklin, H. G. Molina, D. Gawlick, J. Gray, L. Haas, A. Halevy, J. Hellerstein, Y. Ioannidis, M. Kersten, M. Pazzani, M. Lesk, D. Maier, J. Naughton, H. Schek, T. Sellis, A. Silberschatz, M. Stonebraker, R. Snodgrass, J. Ullman, G. Weikum, Widom, and J. Stan Zdonik, "The Lowell Database Research Self Assessment," Massachusetts 2003.
2. P. Dourish, W. K. Edwards, J. Howell, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. Terry, and J. Thornton, "A programming model for active documents," proceedings of the 13th annual ACM symposium on user interface software and technology, New York, USA, 2000.
3. E. T. Freeman, The Lifestreams Software Architecture: Yale University Department of Computer Science, 1997.
4. D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, "Semantic File Systems," proceedings of 13th ACM Symposium on Operating Systems Principles, 1991.
5. T. B. Hodel, D. Businger, and K. R. Dittrich, "Supporting Collaborative Layouting in Word Processing," proceedings of IEEE International Conference on Cooperative Information Systems (CoopIS), Larnaca (Cyprus), 2004.
6. T. B. Hodel and K. R. Dittrich, "Concept and prototype of a collaborative business process environment for document processing," Data & Knowledge Engineering, vol. Special Issue: Collaborative Business Process Technologies, 2004.
7. T. B. Hodel, M. Dubacher, and K. R. Dittrich, "Using Database Management Systems for Collaborative Text Editing," ACM European Conference of Computer-supported Cooperative Work (ECSCW CEW 2003), Helsinki (Finnland), 2003.
8. T. B. Hodel, H. Gall, and K. R. Dittrich, "Dynamic Collaborative Business Processes within Documents," proceedings of ACM Special Interest Group Conference on Design of Communication (SIGDOC) 2004, Memphis (USA), 2004.
9. IBM On Demand Workplace, "How electronics companies can become more resilient and adaptable in a chaotic world," IBM Business Consulting Services, 2003. http://www-1.ibm.com/services/us/igs/pdf/ibm_ondemand_workplace_electronics_18feb04.pdf
10. J. Rekimoto, "TimeScape: A Time Machine for the Desktop Environment," proceedings of CHI'99 late-breaking results, 1999.