Conceptual File Management: Revising the Structure of Classificationbased Information Retrieval

Ali Sajedi Badashian[1], Seyyed Hamidreza Afzali[2], Morteza Ashurzad Delcheh[3]
Mehregan Mahdavi[4], Mahdi Alipour[5]
[1,3]Software Engineering Department,
Azad University
Lahijan Branch, Lahijan, Iran
sajedi@iau-lahijan.ac.ir, Mor.delcheh@gmail.com

[2]School of Information and Communication Technology
(ICT), Royal Institute of Technology (KTH), Sweden
Hr.afzali@gmail.com

[4]Department of Computer Eng.,
University of Guilan, Iran
mahdavi@guilan.ac.ir

[5]Electrical, Computer, IT and
Biomedical Eng. Department, Azad
University – Qazvin Branch, Iran
m.alipour@qiau.ac.ir

ABSTRACT: *This paper addresses the Information Retrieval issues by today's hierarchical systems such as file systems. They usually contain substantial amount of redundant items. Maintaining the structure becomes difficult when large amount of items exist and ambiguity occurs in the structure. In this study, a conceptual method is presented that replaces the "containment" principle involved in current systems by "membership" principle. The proposed Conceptual File Management (CFM) method allows a file to be accessed from multiple folders; furthermore, it keeps the current hierarchical structure with minor changes. CFM reduces the ambiguity and redundancy, therefore the quality of information retrieval is improved. A Conceptual File Manager based on the above-mentioned principle has been developed. Experiments show that it improves the quality of the system in terms of maintaining the structure and retrieving the desired items easier and faster based on users' viewpoints. CFM is applicable to not only traditional file systems on computers, but also file systems on cell phones (and message storage systems), site maps in Web sites, Content Management Systems, taxonomies, and in general, classification based structures.*

Keywords: File management, Hierarchical tree, Taxonomy, File classification, Conceptual file management

Received: 27 August 2010, Revised 30 September 2010, Accepted 7 October 2010

1. Introduction

Current view about the file management is based on traditional hierarchical (tree) structure of folders. The files are located

somewhere in this structure. The users store their files in categorized folders based on their concepts.

Lansdale believed that the goal of organizing files is to facilitate their retrieval [1]. As a result, proper retrieval cannot be achieved in absence of suitable categorization tools that help to place a document in all related categories [2]. As Barreau indicates, this does not mean the inappropriateness of hierarchical file systems; nevertheless, hierarchy is the primary paradigm by which the users organize and retrieve their original content [3].

In this paper, improvements to this paradigm are suggested in order to take advantage of multiple categorization (not much different from current hierarchical file managers in User Interface), allowing users to have different views at the same time. As a result, the user can play more effective role in classifying documents. The classification of files in different periods of time can be enhanced regarding the user's management style. Access time for retrieving documents will also be improved.

This paper is organized as follows: Section II describes previous research on information retrieval including Personal Information Management (PIM) techniques and tools. Moreover, it describes tree based structures. Section III discusses some issues associated with current file managers, such as ambiguity and confusion in information storage/retrieval. Section IV describes the proposed solution with seven primitive operations followed by implementation details in Section V. Experiment results are presented in section VI. Finally, Section VII concludes the paper.

2. Related Work

Every computer user spends time and effort to create his/her filing scheme and have better understanding of placement of files on disk [3]. Barreau and Nardi defined two basic strategies for finding files [3];

• *Location-based search* in which the user takes a guess at the directory/path where he/she thinks a file may be located, browses that location, and looks for the desired file. The process can be iterated as needed.

• *Logical search* in which a text search is applied within division(s) of file system looking for the file name (or some keywords to be searched in contents of the files).

In location-based search the users' memory is judged when scanning files at a location and the file name is used as a behavioral trigger for reminding him/her to take appropriate action instead of searching for a particular name in a logical search. Barreau and Nardi concluded that the users prefer location-based search instead of logical search. In fact, they prefer browsing a list of files and folders rather than trying to remember exact file names. The text searching techniques were used very infrequently. They concluded that location-based search is preferred because it more actively engages the mind and body and inspires a greater sense of control. The user likes the active search manner instead of waiting for the computer to return a list of possibly relevant items. In other words, in the shared categories, the more adopted manner may be the logical search [3].

Often users want to access the file in their first attempt; as a result, the folder structure is the most important aspect of their search process. There are several information retrieval systems such as: Presto [4], WinCuts [5], Workspace Mirror [6] and Stuff I've Seen [7] that aid location-based search and Google Desktop [8], Mac OS Spotlight [9] and Phlat that provide logical search [10].

For example, Phlat [10] is a lookup and tagging system in which searching is done for personal information including file system volumes, e-mail messages, web histories, etc. Cutrel, et al. merged search and browsing through a variety of contextual cues to form an intuitive lookup interface for Phlat. They used whatever the user remembers about the desired information in order to fetch them. The achievement in Phlat was focused on logical search based on multiple cues, not the hierarchical structure of the objects.

Boardman, et al. have considered the need for several hierarchies in three different personal information collections; (1) e-mail folders in MS Outlook; (2) the user's document area in the file system; and (3) bookmark folders stored under Favorites. They considered the folder overlap between multiple hierarchies in different PIM tools. They found that the user is bewildered when retrieving from the hierarchies. As a result, they solved the problem by replicating changes to folder structures in their PIM tools. They produced three unified hierarchies [6]; however, their work was similar to developing one comprehensive hierarchy for all PIM tools and they just synchronized multiple hierarchies resulting in more consistency and convenience.

Nevertheless, neither solves the ambiguity problem for file storage/retrieval studied in this paper.

In some other studies,the importance of categorization in web-based IR is considered [11]. However, the focus here is of file management, but with other applications as well.

3. Motivation and Exploring the Problem

The folder paradigm and hierarchical file system implemented in UNIX-based systems in the 1970s has dominated existing user interfaces as the primary mechanism for organizing files and folders with one-to-many relationships between a folder and the files [2]. Quan, et al. report that thought initially the mentioned hierarchical structure was sufficient, due to extensive advances in disk capacity and dayto- day used applications and file types it encounters problems in organization/retrieval of information [2]. As a result, they pointed to Lansdale's statement that classification into a single category is cognitively hard [1]. They also implemented a system to automatically categorize web bookmarks based on tags as categories. However, their system implemented too many links between categories and the bookmarks, even for the minor links. As a result, it cannot be used for large numbers of categories (such as folders in the file systems).

There is a retrieval problem when you do not know where your stored files are located. The problem is intensified when you are not certain about the file name or don't even know anything about its name. In these cases, you look for the respected file in the existing folders (based on the concept of the file).

The worst problem occurs when there are several files with the same or similar name, but different contents or several versions of a single file. Here, if the hierarchy of folders cannot help finding the respected file, one should test all of them by opening and reviewing them. Similar ambiguity problems occur in storage of the files. Let us consider the problems by simple examples.

Consider the structure of Fig. 1 as a part of a directory hierarchy. The paper "tree.doc" that has presented in SigMod09 is also a result of R2-Taxonomy research grant. So it is duplicated in two folders and their relative addresses are available via the following paths:

Study\Conferences\sigMod09\tree.doc

Study\ResearchGrants\R2-Taxonomy\Results\ tree.doc

It is the same for "simulate.doc" that is presented in InfoVis08. It is a result of your research grants named "R1- Visualization", and is a resource of the R2-Taxonomy research grant. It is also copied once in the folder "ToSendToFriends" because you want to remember to send it for your friends. So it is quadrupled and the versions are accessible via the following relative addresses:

Study\Conferences\InfoVis08\simulate.doc

Study\ResearchGrants\R1-Visualization\Results\
simulate.doc

Study\ResearchGrants\R2-Taxonomy\Resources\
simulate.doc

Study\ToSendToFriends\simulate.doc

Other file repetitions are occurred for "simulate.avi" and "validation.avi".

The redundancy causes growth in file system size and ambiguity in both storage and retrieval:

• When you want to save a file, its actual path is not obvious. For example, in the directory structure of Figure 1, saving a new presentation video "tree.avi" for the SigMod09 conference can be performed in both the following paths:

Study\Conferences\SigMod09

Video\ConferencePresentations

However, you put it in one of the mentioned folders based on your own preferences and the similarity between the new file

and the existing folders. As a result, you save the new files in the nearest folder to its concept. However, it is possible to save a file in a new path while another copy of the same file is located in another path before. The arrangement of folders is sometimes so intricate that numerous copies of a file exits in a computer. This is verified as an experiment in subsection "A" of section "VI". Thus, redundancy is inevitable in a hierarchical structure, especially in the file system.

• When you want to retrieve a file, the actual place of the file is not obvious. It may exist in all the matching paths, several, or none of them. Thus, you should browse all the possibilities one by one. The more critical problem occures when you updated one of duplicate files spread in your computer. Identifying the most recent version between duplicate files arises serious difficulties.

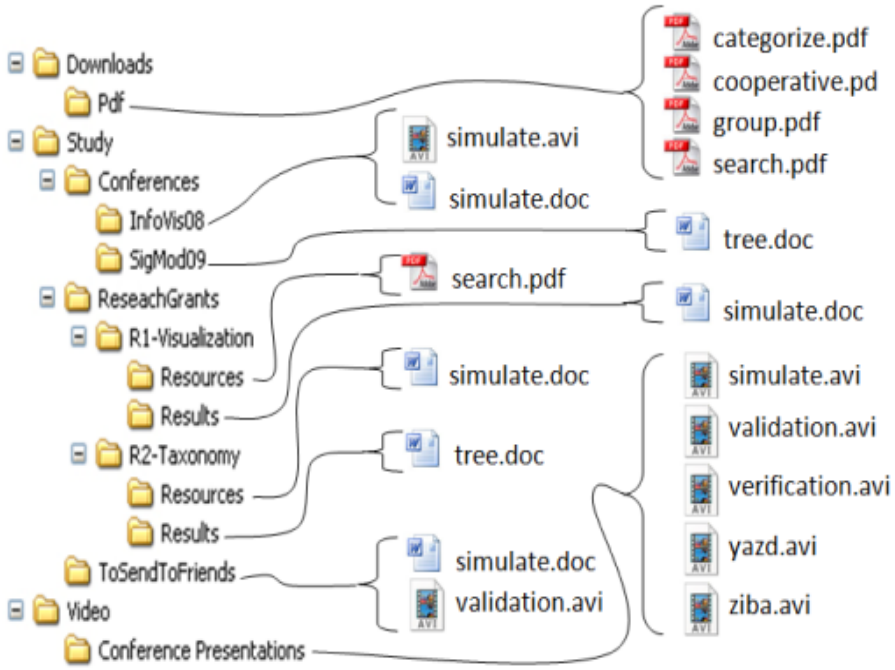

Figure 1. A part of a hierarchy containing folders and files

A means of multiple categorization is needed to solve the problem. If the user links a file to all significant paths, both the storage and retrieval phases enhance. Note that using optional links such as Shortcut, Symbolic Link (Soft Link), Hard Link and Junction Point doesn't solve the problem.

The mentioned links do not offer comprehensive view over files and their access methods. They only made trivial links to handle multiple addressing of the files. They enhance the shortcuts, but have several drawbacks. For example, Hard Links of a file fail to work when changing the place of the file. They save the attributes of the target file separately, so all of them change due to redundant information in the Hard Links. Symbolic links do not support tracking of addressing. They remain orphan when deleting the target file. Besides, changing the place of the target file makes its symbolic links useless. Traditional shortcuts act more restricted from symbolic links. Finally, the junction point is a means of mounting a file (or folder) to another partition to be appeared here, but actually being stored somewhere else. It is used due to space limitations in partitions of disk. There are also more limitations and drawbacks that are not mentioned here. On the other hand, they do not have user simple visualization. They are hard to make use of, keep track of a pursuit, even if all the limitations resolved.

Also note that some third party software such as Microsoft media player media indexer that are designed to manage media files with dynamic auto-classifying ability have limitations such as single purpose usage [12], [13]. As a result, they do not solve the mentioned problems.

4. proposed Solution

Let us consider the solution in a plain context eliminating the containment rule, and replacing it by membership rule. Everything (representing a file) is in a plain space, named "Root".

All the files are stored in this Root directory and supposedly there are no folders. Consider the structure of the previous example. In the CFM structure, the notion of “folder” is changed to “concept”. Also “membership” relation connects files to the concepts. In this view, each item can be a member of several concepts, so there are several ways (as several paths) to access a file. The file “simulate.doc”, for example, is accessible via the following paths as before:

Study\Conferences\InfoVis08\simulate.doc

Study\ResearchGrants\R1-Visualization\Results\
simulate.doc

Study\ResearchGrants\R2-Taxonomy\Resources\
simulate.doc

Study\ToSendToFriends\simulate.doc

Note that in these paths, the file names are not separate copies of the file. They are just shadows of the file indicated in the UI to represent the file’s role. File shadows are obtained from the relation between the file and concepts and act as handles to the files. They do not exist as separate objects in the real world, but the Conceptual File Manager creates them as temporary interface objects.

In fact, by changing the “folder” material to “concept”, there will be a shell of concepts pointing to the files instead of a mass of folders containing files. Each file actually matches with several concepts, from the major ones to the less important ones. This perspective is the fundamental of CFM, but in the current file systems, each file is related to only one folder that is contained in. In addition to CFM, the “File” itself is going to be separated from the interface. Interaction with the users also remains somewhat unchanged, but the basic operations in the file system need modifications and are discussed in this section

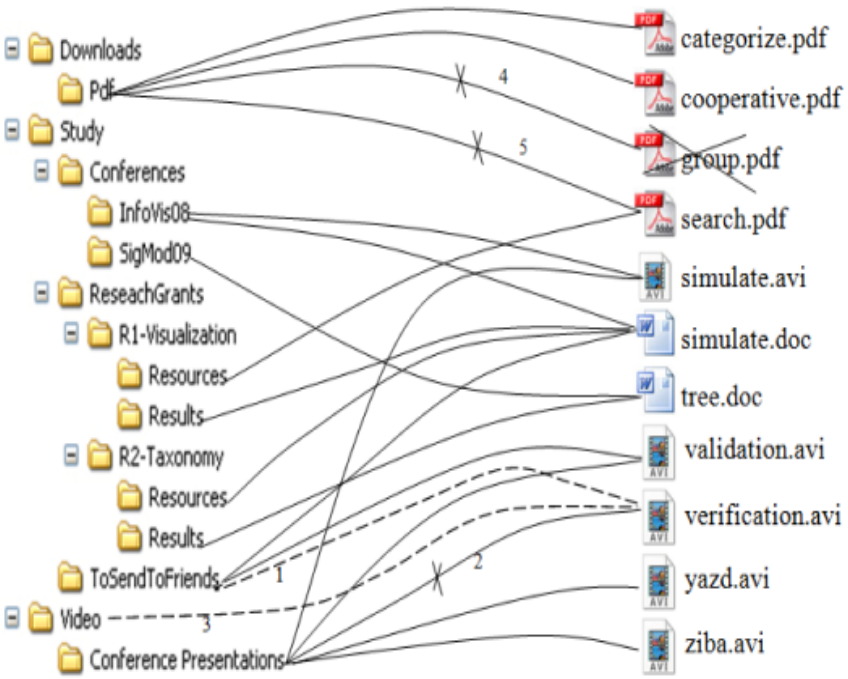

Figure 2. The proposed (conceptual) structure for the same files

4.1 New File

when creating a new file in a concept, in fact, no file is created there. Instead, the file is created in the “Root”, and a shadow is created in the concept (the desired path) from the user’s view, but actually a relation between the file and the path is created.

4.2 Copy – paste

When a file is copied to another concept, in fact, no file is copied. Instead, a shadow is created in the new path from the users

view, but actually, a new relation between the file and the destination path is added. For example, copying the file "verification.avi" to "ToSendToFriends" concept only makes a relation between them (see the dashed line numbered as "1" in Figure 2)

4.3 Cut – Paste

When a file is cut from a concept and is paste into another, in fact no file is displaced. Instead, the shadow is moved from the source path to the destination from the users' view, but actually, only a relation (between the file and the source path) is removed and another between the file and anotherm destination path is created. For example, moving the file "verification.avi" from "Video\ConferencePresentations" to "Video" only removes the previous relation with "ConferencePresentations" (the relation labeled "2") and creates another with "Video" (the dashed line labeled "3")

4.4 Delete

When a file representing by a shadow is deleted, in fact, there are two cases:

• First, the shadow is the only shadow of the file. In this case, the shadow is deleted from the user's view, and the file is also deleted (as deleting "group.pdf" from "Downloads\Pdf"; both the file "group.pdf" and the relation labeled "4" are deleted).

• Second, there are other shadows for the file. In this case, in fact, no file is deleted. Instead, the shadow is deleted from the users view, but actually the relation between the file and the path is deleted (as deleting "search.pdf" from "Downloads\Pdf"; the relation labeled "4" is deleted, but the file "search.pdf" remains unchanged in the files list).

4.5 Rename

When renaming a file, in fact, both the name of the file and all its shadows are changed from the users view, but only the name of the file is actually changed. All the shadows reflex the changes when are obtained from the exiting relations with the file.

4.6 Properties

When showing the properties of a file, in fact, the properties of the shadow are not shown. Instead the properties of the target file are shown. A supplementary functionality is to show all the shadows of the file in this case, because the user wants the information about the file; hence one or more paths are shown.

4.7 Edit

When modifying (editing) a file accessed by a shadow, in fact, the changes can be seen when accessing the file from within all other shadows. However, a supplementary activity is to offer a choice to the user to make a duplicate copy from the file, to keep the other shadows point to the previous version of the file, but the current shadow points to a new file (representing "Save as").

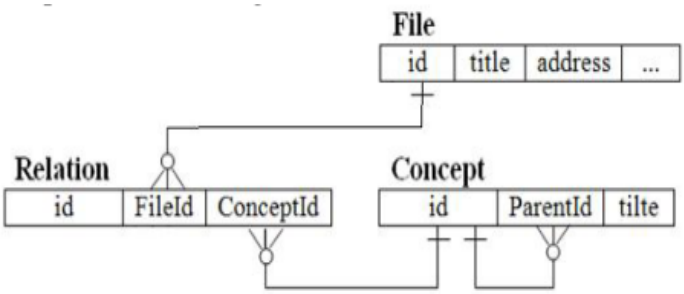

Figure 3. The basic tables needed for CFM in database manner

5. Implementation Issues

There are considerations that were achieved during the analysis, design and implementation argumentations of the developer team. These are related to the problems, rules and examinations of running the program under different situations, especially the seven actions of the previous section. Here, some of them are mentioned:

As mentioned before, all the files are located in a single path named "Root". Nevertheless, there may be several files with the same name; thus, in order to separate these files, a hidden comment including the file name and an automatic sequence

number is assigned to each file. The comment of a file guarantees clarity and uniqueness of the files in the system, even with similar names. The comment is not shown unless there are two different files with the same name in a path. However, this occurs rarely in the reality.

• The CFM can be achieved in a database approach that stores the concepts, file names and relations in three separate tables (see Figure 3). The innermost table indicates the many-to-many relationship between the files and the concepts.

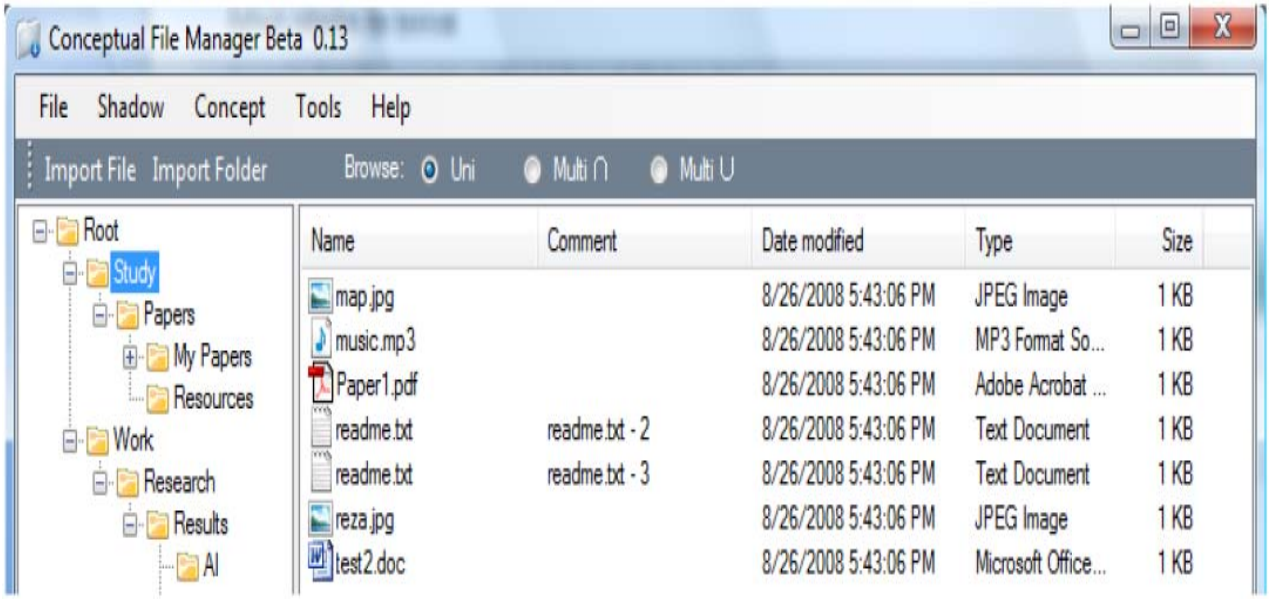

Figure 4. The general view of CFM Environment

6. Experimental Results

Two experiments have been done in order to consider the effects of CFM. First, several computers have been examined to determine the redundancy rate. Then, a prototype was developed in order to see the overall users' feedback.

6.1 PC Analysis

Ten separate drives have been examined by directory Opus 9.0 [14] to find duplicate files. The results show that in average17,000 files out of 70,000 containing 2GB to 10GB out of 40GB to 80GB files available on the hard disks were redundant items. In other words between 5 to 10 percent of the hard disk space is copies of the existing files from one to 10 extra copies. Considering redundant items discovered that they are produced by both the system (automatically) and the user. However, a massive amount of the redundant files are related to the user.

6.2 Prototype analysis

A prototype was developed to consider both the challenges of the CFM systems and the users' reflections against the conceptual approach. The UI for ordinary usages is designed similar to Windows Explorer (See Figure 4). In the experiment, nine Students including three female (numbers 2, 5 and 9) and six males participated in a two-day experiment to store and retrieve documents using both paradigms (i.e., Folder paradigm with "MS Windows Explorer" or "My Computer" and CFM paradigm with our implemented prototype system in two separate examinations. The users were familiar with neither CFM software nor the CFM approach before; hence the chance was given to them to familiarize themselves with the CFM prototype for at most one hour after demonstrating the idea and the program.

Although the time is not sufficient to be proficient CFM users, they all finished the practice after 15 to 45 minutes. According to their opinion, the UI was easy-to-learn and it resembled the UI and the operations of Windows Explorer. Therefore, they were able to make hierarchies of concepts and categorize the documents easily.

A competition was set up for making best categorizations in either paradigm. The examination for each of the two paradigms was conducted in two phases; Storage and Retrieval. The users were divided into two groups; the first four users examined

the folder paradigm first and the other five examined the CFM paradigm first. Interestingly, there was no significant difference between the average results of two groups. Hence the final results are studied together.

1) Storage Phase

In this phase, 88 documents in a single folder were given to the users. They had to open each document, create hierarchies regarding the examination (i.e. folder or CFM) and categorize them in the folders/concepts by cut/copy and paste operations provided in CFM and MS Windows. In MS Windows examination, they could use drag and drop operations too.

They created an average of 15 folders and 25 concepts in either experiment. The categorization time was varied from 1320 to 5700 seconds using folder paradigm (average=3177,σ =1505) and from 1900 to 6700 seconds using CFM paradigm (average=3622, σ =1704) (see Figure 5).

A 14% increase in CFM was very good during the nature of the problem; in other words, a file should be connected to more than one concept; as a result, the time slightly increases (not so much) because the user does not have to place a file in its best location. He/she can paste a file in several concepts to increase retrieval efficiency without considering size or redundancy matters. On the other hand, they were unfamiliar with the CFM previously, but with lots of folder experiences in several years. These facts might offset the lower performance of CFM in long term.

2) Retrieval Phase

In the second phase, 19 randomly selected files were demanded one by one and the time to find the files were recorded. The selected files were equal for the two experiments (i.e. folder and CFM paradigms) to achieve comparable results, but the users did not know this before finishing their categorization. Each file search was started by giving the name of the document or some clues from its contents.

In order to control the experiment synchronously and preventing exhaustion and give up, a maximum threshold of two minutes was applied for each test case. The test case result was recorded as "failure" if the user was not able to retrieve the file in this time limit. Trial experiments showed that in such cases the user usually gives up or spends a long time (longer than three minutes in average) for retrieving the file. As a result, the time 160 seconds was chosen for these cases.

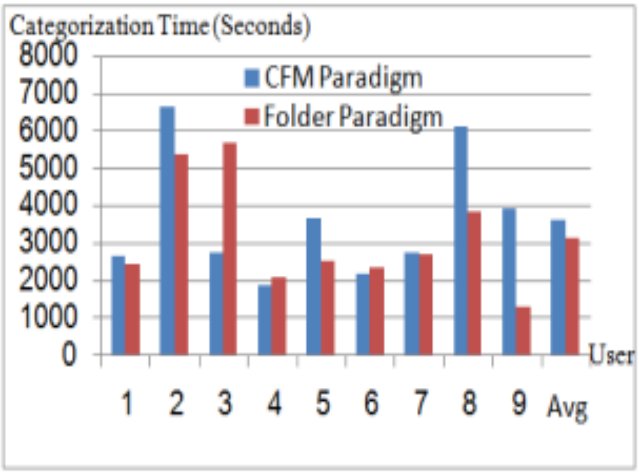

Figure 5. The categorization times using CFM and folder paradigms per user

The average access time in CFM was 57.0 seconds (σ =23.6) against 62 seconds (σ =18.6) in folder paradigm (7.9% faster in CFM) (see Figure 6). However, comparing the "failure rate" of the users is far more interesting; On average, 2.22 cases out of 19 test cases were failed to be found in CFM (σ =1.2) against 4.33 cases in folder paradigm (σ =2.3). 47.8% decrease in failed attempts as shown in Fig. 7.

In the real situations the user gives up when he/she fails finding the file by examining several possible locations and creates an extra copy of the file often with a different name. Our "PC Analysis" supports this claim. However, better results can be expected for CFM when the experience time is increased. In other words, more practice with the software will generate better results.

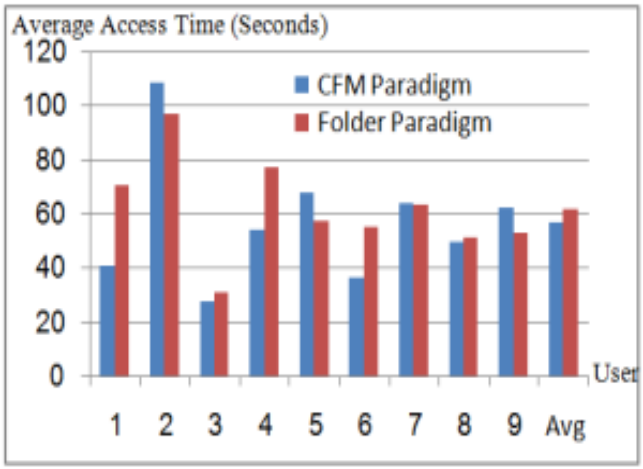

Figure 6. The average access times using CFM and folder paradigms for 19 test cases per user

7. Conclusion and Futute work

A method has been proposed to gain a flexible, flat visualization to the file system. The advantages of CFM are described (but not limited) to:

• Clarity of the system; the user can save a file in the system with as much access methods (paths) as he/she wants.

• The system offers less ambiguity in both storage and retrieval.

• There will be no redundant items while the user places a file in several concepts without duplicating the files.

• Quick access to the files (i.e., finding the files in the first attempt).

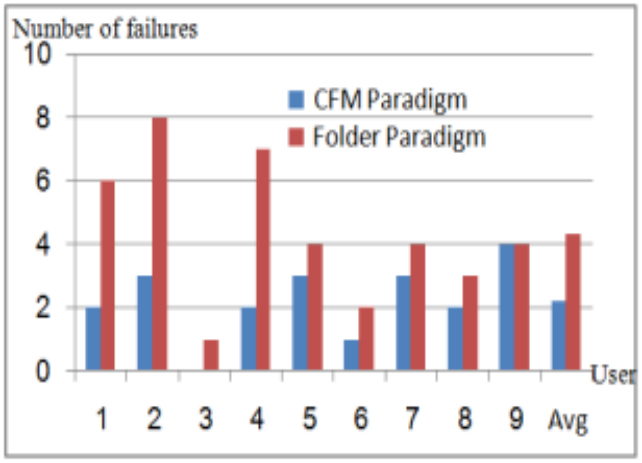

Figure 7. The number of failures using CFM and folder paradigms for 19 test cases per user

• Size reduction (due to eliminating redundancy).

However, creating the completely conceptual tree needs concentration. Using the CFM manner, the user keeps both benefits of the manual hierarchical categorization in nowadays file systems and membership in more than one category (also associating to more than one filter or tag) simultaneously. However, it needs much user interface support (to be embedded in different software) in order to effectively evaluate the usefulness of dynamic categorization scheme.

The CFM approach eliminates the “containment” rule considered previously in not only the file managers, but also file

systems. It is even applicable in other classification based structures such as text categorization, taxonomies, web mail management and site maps. In all applications, there are a mass of data to be visualized as categories based on different conceptual hierarchies.

References

[1] Lansdale, M.(1988). The Psychology of Personal Information Management, *Applied Ergonomics*, 19 (1) 55–66.

[2] Quan, D., Bakshi, K., Huynh, D., Karger, D. R. (2003) . User Interfaces for Supporting Multiple Categorizations, *In*: Proceedings of INTERACT 2003, 228-235.

[3] Barreau, D., Nardi, B. (1995). Finding and Reminding: File Organization from the Desktop, *SIGCHI Bulletin*, 27 (3) 39 43.

[4] Dourish, P., Edwards, W. K., LaMarca, A., Salisbury, M.(1999). Presto: An Experimental Architecture for Fluid Interactive Document Spaces, *ACM Trans. Comput.-Hum. Interact*., (2) 133–161.

[5] Tan, D. S., Meyers, B., Czerwinski, M. (2004). Wincuts: manipulating arbitrary window regions for more effective use of screen space, *In*: CHI '04: CHI '04 extended abstracts on Human factors in computing systems, 1525–1528.

[6] Boardman, R., Spence, R., Sasse, M. A. (2003). Too many hierarchies? The daily struggle for control of the workspace, *In:* Proc. HCI International.

[7] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., Robbins, D. C.(2003). Stuff i've seen: a system for personal information retrieval and reuse, *In*: SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, New York, NY, USA. ACM Press, 72–79.

[8] Google, Inc.(2004). Google Desktop. http://desktop.google.com/

[9] Apple, Inc. (2004). Mac OS X Spotlight. http://www.apple.com/macosx/features/spotlight/

[10] Cutrell, E., Robbins, D. C, Dumais, S. T, Sarin, R. (2006). Fast, Flexible Filtering with Phlat — Personal Search and Organization Made Easy, CHI 2006 Proceedings, April 22–27, Montréal, Québec, Canada.

[11] Seig, A., Mobasher, B., Lytinen, S., Burke, R (2004). Using Concept Hierarchies to Enhance User Queries in Web-Based Information Retrieval, *In*: Proceedings of IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria.

[12] Li, M., Claypool, M., Kinicki, R. (2002). MediaPlayer™ versus RealPlayer™: a comparison of network turbulence, *In*: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement.ACM Special Interest Group on Data Communication. 131 - 136. DOI=http://doi.acm.org/10.1145/637201.637221

[13] Windows Media Tools Components http://technet.microsoft.com/enus/ library/bb676146.aspx

[14] GP Software, http://www.gpsoft.com.au/