IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 2, 2018, accepted December 23, 2018, date of current version January 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2890165

360° Semantic File System: Augmented Directory Navigation for Nonhierarchical Retrieval of Files

SYED RAHMAN MASHWANI AND SHAH KHUSRO, (Member, IEEE)
Department of Computer Science, University of Peshawar, Peshawar 25000, Pakistan

Corresponding author: Shah Khusro (khusro@uop.edu.pk)

ABSTRACT The organization of files in any desktop computer has been an issue since their inception. The file systems that are available today organize files in a strict hierarchy that facilitates their retrieval either through navigation, clicking directories and sub-directories, in a tree-like structure or by searching (which allows for finding of the desired files using a search tool). Research studies show that the users rarely (4%–15%) use the latter approach, thus leaving navigation as the main mechanism for retrieving files. However, navigation does not allow a user to retrieve files nonhierarchically, which makes it limited in terms of time, human effort, and cognitive overload. To mitigate this issue, several semantic file systems (SFSs) have been periodically proposed that have made the nonhierarchical navigation of files possible by exploiting some basic semantics but no more than that. None of these systems consider aspects such as time, location, file movement, content similarity, and territory together with learning from user file retrieval behaviors in identifying the desired file and accessing it in less time and with minimum human and cognitive efforts. Moreover, most of the available SFSs replace the existing file system metaphor, which is normally not acceptable to users. To mitigate these issues, this research paper proposes 360° SFS that exploits the SFS ontology to capture all the possible relevant file metadata and learns from user browsing behaviors to semantically retrieve the desired files both easily and timely. Based on user studies, the evaluation results show that the proposed 360° SFS outperforms the existing traditional directory navigation and recently open files.

INDEX TERMS Information management, information retrieval, file systems, recommender systems.

I. INTRODUCTION

Dealing with rapidly growing files on our desktop computers is becoming one of the most daunting tasks. The problem worsens due to the hierarchical organization of files into directories and sub-directories. These files are either retrieved through browsing in which a user navigates through directories and subdirectors, or through searching using the search functionality of file systems. For the ease of retrieval, a desired file should be retrieved non-hierarchically, but this is currently possible only through searching. The problem is that people prefer navigation over searching [1]–[5] since it involves less cognitive attention than searching [6]. In addition, searching can be used as an alternative to browsing if a user is unable to retrieve files through navigation. However, existing research [1], [2] has found no evidence that improved desktop search engines have changed peoples' minds and that they still prefer navigation over search. In addition to this significant contribution of navigation towards reducing cognitive attention in locating files, navigation in traditional file structure suffer from several limitations, including the following:

- In large systems, it is not possible for a user to remember the exact path of the stored files.
- If a file is saved at level n in the hierarchy, then the user must navigate to the desired file by taking n navigational steps to retrieve it. There is no way to navigate to the desired file nonhierarchically.
- The importance of the stored files changes according to the context in which the file is accessed. The context may change due to several factors, including the geographical location of the user, the day of the week and the time of the day. For example, the files of interest for a user may be different in normal office hours, in meeting times, in the evening and in weekends with family.

To efficiently retrieve files in a file system, an intelligent file system needs to dynamically organize files per their importance at the current time. In addition, files stored deep inside a hierarchy may be important, and therefore they must be made easily accessible like any other files stored near or in the root directory. Therefore, efforts are needed to reduce the time and the human and cogitative efforts in navigating to a file. In addition, traditional file systems neither provide enough metadata nor a way to cope with this challenge [7]. To mitigate some of these issues in traditional file systems, Semantic File Systems (SFSs) have been devised. A SFS enables file retrieval based on the associated semantics rather than its physical location in the hierarchy. Several SFSs have been proposed to date. However, they are limited since they replace the traditional file system metaphor, require extensive manual annotations, or are unable to learn from user habits. To mitigate these issues, we proposed 360°-SFS, which provides enough metadata to augment navigation in hierarchical file systems by automatically learning from user habits. It allows a user to navigate to the desired files nonhierarchically in less time (total time a user takes to retrieve a file, starting from the first mouse movement made by a participant until they clicked on the desired file), navigational steps (each action a user takes until the target file is opened) and cognitive attention. In addition, it does not replace traditional hierarchical file organization but attempts to overcome some of its limitations, including the following:

- *Remembering exact file location:* Users store information at different locations in the hierarchy according to their preferences at that time. However, they forget the locations as time passes, hence making navigation difficult, especially in large systems.
- *Accessing a file from the exact stored location:* Even if a user knows where the required file is stored in the file system, they must traverse the file system directory tree to reach the exact location. It is a tedious job if the required file is stored deeper in the directory's hierarchy. There is no mechanism to navigate to the files nonhierarchically.
- *Lack of support for recommendations*: To reduce retrieval time and human effort, no such mechanism exists for the recommendation of the desired files to a user while navigating.
- *Cognitive overload:* Although navigation requires less cognitive attention than searching, it still requires enough cognitive attention to navigate to the files in large systems, especially if the file is stored deeper in the hierarchy.

At a specific point in the navigation, a directory only shows information items present at that level in the hierarchy and hides all other information items contained in the directories up or down the hierarchy. The 360°-SFS intelligently finds the important files from subdirectories and shows them in virtual NOW directories. NOWs are special directories that are automatically created in every directory of the file system. The symbolic links to contextually related files are dynamically created in NOW directories, thus making navigation easy and enabling a user to navigate to the desired file in less time, fewer navigational steps and with less cognitive overload.

II. BACKGROUND AND RELATED WORK

SFS is a vision for the future of file systems where information in the file systems is given explicit meaning to be processed by machines automatically and consumed by the users easily. SFSs extend traditional file systems to organize and retrieve information according to their semantics, intentions and relationships with other resources rather than their physical locations. Approaches to SFSs can be broadly divided into two categories: integrated and augmented [8]. The former approach incorporates semantic features directly within the file system by modifying the file system's interface. Since files and directories are manipulated via this modified interface, these systems are not backward compatible with traditional applications. Semantic features can only be accessed through the modified file system's interfaces. Instead of replacing traditional file system interfaces, the later approach augments the traditional file system interface to incorporate semantic features in the file system. Hence, it is backward compatible with traditional file manipulating interfaces and applications. Furthermore, augmented approaches can directly use the improvements that are made to the: operating system related components and the existing traditional file system layers [9]. Researchers focus on augmented file systems because they place fewer demands on the end-user. 360°-SFS falls in the category of augmented SFS. It is directly usable with traditional applications. Our system works in the background without affecting traditional applications. A user can still use traditional applications, such as any of his favorite word processors for documents, file managers for manipulating files, etc.

The MIT-SFS [10] is known as the first SFS that allows retrieval of files based on their attributes without replacing the traditional hierarchical organization. File type specific transducers mine and index the attributes from files. The system maps user's entered path into a query and the result is displayed to the user with the help of the virtual directories and symbolic links to the files. MIT-SFS allows users to access files via their attributes. For instance, a query "*ls -F /sfs/owner:/Smith*"returns all files that are owned by Smith.

The Context Aware File System (CAFS) [11] exploits the time and location information for making data dynamically available to a user. For instance, if a user has to present at 3:00 PM at a conference room, then at that time when he comes to the presentation stage his presentation file will be automatically displayed on the screen. If during the presentation he moves from one location to another, then the same slide is loaded on the screen close to him. In contrast, 360°-SFS recommends the files that a user frequently accesses at a particular time and location. This recommendation is done based on user's previous interactions with file system objects. Furthermore, CAFS is especially designed for ubiquitous computing. Such file systems are designed to support different types of interactions that ubiquitous computing spaces introduce and enable. In our study, we target the file systems of personal computers (i.e., desktop or laptop) with conventional peripherals (i.e., keyboard and mouse) and environments. File systems that are especially designed for ubiquitous computing and distributed computing are out of the scope of this study.

The Semantic Desktop is a step towards addressing the problem of information overload on our desktop computers. Sauermann *et al.* [12] stated that the file system is one of the building blocks for a Semantic Desktop. Ontology-based file systems could enhance the capabilities of file systems [13]. Integrating ontologies into a file system enables one to share file semantics with all stakeholders, thereby facilitating their agreement on the same meaning. To use the features of the ontology based file system, Ngo *et al.* [13] suggested a modification in the file system interface. However, this redesign effort is not a primitive task and may make many existing tools and procedures obsolete.

Tagsistant [14] is a tag-based SFS. It allows for the access of the files from multiple paths with the help of logical OR and AND operators. For instance, the path *"/a/AND/b"* shows all of the files tagged with both "*a*" and "*b*." Similarly, the path *"/a/OR/b"* shows all of the files tagged with either "*a*" or "*b*." Tagsistant interprets Path *"/a/AND/b"* the same as path *"/b/AND/a"*. The early versions use Tag-Manager to manage tags. However, it is discontinued in the later versions. The later versions also support comparison operators and automatic tagging. SemFS [15] is a tag-based SFS that is based on the same author's previous work TagFS [16]. SemFS maps a user's entered path to a query. For example, the path *"/a/b/c"* executes the query and returns objects that are tagged with a, b and c. It interprets path "$/a/b$" the same as *"/b/a/"*. The system allows one to access files via more than one path and recommends tags while navigating on the basis of overlapped tags. However, these methods suffer from the limitations of mandatory manual tagging and the inability of handling a large tag set over an extended period of time. Albadri *et al.* [17] proposed the TreeTags model to overcome some of the problems of hierarchical file systems. They replace names with (multiple) tags and directories with collections. Collections are organized as a tree. They also introduce multiple tags for collections and a basic query language.

LiFS [18] provides support for linking of files internally in file system and provides support for arbitrary annotations. Compared to other systems, it provides high performance because of its in-memory implementation. LiFS is implemented using FUSE API in Linux. To manipulate attributes and links, the authors propose some new system calls. They augment inodes using extent nodes, link nodes and attributes nodes. It does not provide support for external links. Additionally, LiFS is only accessible via a modified POSIX interface.

Soules and Ganger [19] stressed the need for annotating files on the basis of context analysis. They outline different approaches to capture file-related contextual information. They observe that while storing files, users do not like to spend extra time on annotating files. They just choose a name and directory for the file. The name of the file and directory it resides in can be exploited as the context of the file. Furthermore, if a file is accessed after accessing another file, then both files can be declared as related. Such automatic annotations could reduce the burden on users. Similarly, the findings of Okoli and Schandl [20] show that if two files are opened and a third file is modified or created with a specific time frame, then all three files could be related. A user normally retrieves files successively in the same context.

QMDS is a DBMS-based system in which Ames *et al.* [21] propose a graph data model for file system metadata management and a query language. Memsy [22] was proposed to address the problem of information fragmentation since a user's personal files are scattered among different devices and web applications (for instance, Google Drive, Flickr, emails and external portable storage devices, etc.). Memsy crawls and indexes the local file system and cloud storage services and presents different versions of the files in a single desktop interface. However, currently the Memsy virtual file system only considers local files. The Memsy file system allows one to browse the collection of files using a regular Windows 7 file manager.

FindFS [23] provides a special query directory at its root. A directory created in this directory acts as a virtual directory and is treated as a query. The symbolic links to the files are automatically created with this directory as per the user's entered query. The result of the query is accessible until a user deletes the virtual directory. The system exploits the extended attributes for storing users' defined attributes as name value pairs. The system is backward compatible with almost all traditional applications. SileFS [24], [25] is based on the concept of a sile (short for semantic file). A sile replaces the traditional file in that it supports more metadata attributes. However, access to the semantic model's elements is not possible through the normal file system interface. Semplorer, a modified file manager, is used to manipulate a sile's semantic annotations. TripFs [26] links the files stored on file systems with the Linked Data cloud by assigning globally unique dereferenceable HTTP URIs to files.

Mizrachi and Deluca [27] described techniques for displaying a directory icon to assist users to retrieve files easily. Files are uploaded and shared to an external web server to enable other users to annotate the files. The popularity of a file is calculated based on the users' manual annotations. In the local file system, the system displays a file as having the highest popularity as an icon for the directory where the file is stored. If a popular file is an image file, then it is displayed as an icon of the directory. In the case of a video file, the system extracts a suitable image from the video and displays it as an icon for the directory. The authors also suggest modifying the icon size of a directory per its access frequency. If a directory is accessed frequently, its icon needs to be larger in size and vice versa.

The GFS [28] is a tag-based SFS that automatically extracts tags from a file's path. Unlike other SFSs, it also allows users to retrieve files in the traditional way, and it does not replace

TABLE 1. Comparison of 360°-SFS with other file systems.

	User's Habit	Content Similarity	Tags	Logical Operators	RDF
MIT-SFS [10]	-	-	Yes	1	-
Tagsistant [14]	-	-	Yes	1, 2	-
Semfs [15]	-	-	Yes	1	Yes
Tagfs [16]	-	-	Yes	1	Yes
Tagfs [29]	-		Yes	-	Yes
TreeTag [17]	-	-	Yes	1, 3	-
WSFS [30]	-	-	-	-	-
LiFS [18]	-	-	Yes		-
WinFS [31]	-	-	Yes	-	-
SileFS [24]	Yes*	-	Yes	1, 2, 3 (In Filters)	Yes
LODFS [32]	-	-	-	-	Yes
LISFS [33]	-	-		1, 2, 3	-
Presto [34]	-	-	-	-	-
TripFS [26]	-	-	-	-	Yes
QMDS [21]	-	-	-	1, 2, 3	-
FindFS [23]	-	-	Yes	1,2	-
Memsy [22]	-	-	Yes	-	-
File Folder Display [27]	-	-	-	-	-
GFS[28]	-	-	Yes	-	-
360°-SFS	Yes**	Yes	Yes	1, 2, 3	Yes

* file accessed together only
** file movement between directories, file Accessed together, time and geographical location

the file system's metaphor. However, the semantics in GFS are limited to tags only. It does not consider semantics other than tags, and it does not automatically learn a user's habit.

Despite its many contributions, SFSs are yet to be successfully welcomed by common computer users. This is because common computer users are used to and more familiar with the file system metaphor of file organization (i.e., the analogy of files, directories and subdirectories), and they do not want to replace them with a new paradigm [35]–[37]. Extensive manual annotations and replacing file system metaphors with a new paradigm makes it difficult to attract common users. We support approaches that do not replace the file system metaphor and the hierarchical organization, such as MIS-SFS [10] and GFS [28]. However, they do not exploit semantics other than basic file attributes and tags, and they do not learn automatically from a user's interaction with file system resources. Tags are popular on the web for content sharing, but in regard to Personal Information Management, users prefer directories over tags [36].

However, in contrast to the existing SFSs, our system mostly learns automatically from user's interactions, and it shows important files to users in NOW special directories to improve navigation. This is done on the basis of different semantics, such as the peak access time of a file, the previous directory of a file, the user's geographical location when accessing a file, etc., discussed in detail in Section III. The 360°-SFS is backward compatible with existing applications. We do not force users to use the modified file managers, nor have we replaced the file system metaphor for file organization. We try to improve information retrieval via navigation in traditional hierarchical file systems. This makes it easy for a common computer user to understand and start with the 360°-SFS. We also propose the concept of territory, which is a novel approach to address the growing number of elements in other virtual directories. The system also supports tag based navigation, but unlike existing tag-based approaches [14], [15], [29], we do not replace the traditional hierarchical directory navigation. Table 1 compares and evaluates 360°-SFS with the state-of-the-art solutions and highlights our contributions.

III. 360° SEMANTIC FILE SYSTEM

For the ease of access and management, a user groups semantically related files into directories and subdirectories of their choice. We present the idea of using a NOW special directory at each level of the directory's hierarchy, which will intelligently aggregate the important and contextually related files from different locations of the file system down the hierarchy. Our proposed file system enables users to access files semantically using special directories and allows users to access the files from their original locations.

Filtering out and ranking the important files among millions to be displayed in NOW is a challenging and complex task. We perform this task by considering different features such as the peak access time of a file, co-occurring files, previous directories of a file, territory, user's geographical location at the time of accessing a file, etc. The following subsections further elaborate on the 360°-SFS.

A. TEMPORAL

The system keeps a record of file access time stamps. This helps in generating file statistics based on the time of day and the day of the week during which a particular file was frequently accessed. A file becomes a candidate for the NOW directory based on the peak access days and timings of the file. Only higher ranked files are included in the NOW directory from the list of candidate files.

A file is considered as a candidate file only if both the current day (i.e., Monday) and the current hour (i.e., 10:00 AM) are marked as peak. Therefore, the computation is done in two steps.

In the first step, the system determines whether the particular day of the week is among the peak days or not. Let "N" denote the file-access count and "D" denote the total number of days since the file was created or a maximum of 13 weeks (91 days). Then, the threshold "T" that represents the average access frequency of that file per day is defined as given in Equation (1).

$$T = \frac{N}{D} \tag{1}$$

The threshold "T" is used to determine the peak day status of the file. Now, let "c" denote the file-access count of a particular day of the week and "d" denote the total number of occurrences of that day since the creation of the file or a maximum of 13 weeks (91 days). Then, the average access frequency of the file on that day "F_d" is defined as given in Equation (2).

$$F_d = \frac{c}{d} \tag{2}$$

Hence, if "$F_d \geq T''$ for that particular day of the week, then that day is considered as a peak day for the file.

In second step, the system determines whether the current hour is a peak hour or not. This step only occurs if a day is annotated as a peak day.

The threshold "t" that represents the average access frequency of a file per hour on a particular peak day is defined as given in Equation (3).

$$t = \frac{F_d}{24} \tag{3}$$

The threshold "t" is used to determine the peak hour status of a file. Let "m" denote the total file-access count at a particular hour on a particular peak day. Then, the average access frequency of the file at that hour and day "F_h" is defined as given in Equation (4).

$$F_h = \frac{m}{d} \tag{4}$$

where "h" represents any hour from 0-23.

Hence, if "$F_h \geq t$," then the current hour is considered as a peak hour for the file, which makes the file a candidate file for the NOW directory.

If a user opens a file for a shorter amount of time and second file for a longer amount of time, then normally a single file-access is recorded for both files. To include the contribution of the file that remained opened for longer times, the system automatically records a single file-access each time after time interval "I," which is computed by dividing 60 (minutes) by "t."

We restrict the values of "N" and "c" to a maximum of 91 days since a user's priorities change over time and since Agrawal *et al.* [38] showed that the median age of a file ranges between 80 to 160 days. Thus, we only consider the files that are frequently accessed within a recent period of 13 weeks. The system does not consider a file as a candidate if it has not been accessed in the last 4 weeks.

Many applications support access to Recently Opened Files (ROF). Our temporal based recommendation differs from this in terms of the following.

- The ROF recommendation is done at application level, which only restricts its usage for a particular application. Every application maintains its own metadata about the ROF, and other applications cannot take advantage of the ROF list maintained by an application. Since our recommendation is done on the file system level, it can be exploited by any application (even from the command line) without any modification to traditional applications.
- The ROF list only shows the most recently opened files. Different applications restrict the ROF list to approximately 3 to more files. Different applications show the ROF list differently. For instance, some sort them based on recency or frequency, and some group them temporally (i.e., today, yesterday, last week, older, etc.). In our case, the system shows files based on different semantics, such as files that are frequently accessed at the current time and current geographical location, access patterns, previous directories of a file, etc.

B. GEOGRAPHICAL LOCATION OF THE USER

Each time the file is accessed, the system annotates it with the user's current geographical location. This lets the system know which file is frequently accessed at which geographical location.

Let *"$G = \{g_1, g_2, g_3 \ldots g_n\}$"* be a set of geographical locations at which a file was accessed, where "$g_{i''}$" denote a particular location and "F_{g_i}" denote the access frequency of a particular file at that location in last 13 weeks or since its creation. Then the threshold "v" that represents the average file-access frequency per location is defined as given in Equation (5).

$$v = \frac{\sum_{i=1}^{n} F_{g_i}}{n} \tag{5}$$

The threshold "v" is used to determine the peak location status of a file.

Hence, if "$F_{g_i} \geq v$," then the current geographical location "g_i" is considered as a peak location for the file, which makes the file a candidate for the NOW directory.

The geographical location details could be captured from the Wi-Fi/IP address or a laptop's built-in GPS receiver. The IP address-based location is not very accurate compared to hardware-based solutions. Currently, a built-in GPS receiver in laptops is not common, but we are expecting it in near future.

Temporal and geographical location-based recommendations are most effective for multi-profile personalities, which are those that retrieve particular files at particular times and locations. For instance, consider the scenario of a person who works in different portfolios within an organization or with multiple organizations at a time. We can also consider the scenario of a student who is working a part-time job along with his studies. In both scenarios, the system will recommend the appropriate files to the user at the right time and place.

C. FILE MOVEMENT BETWEEN DIRECTORIES

A user groups semantically related files in a directory. With the passage of time, a user moves files between directories. From this file movement, it can be inferred that a file is also related to the directories where it was previously stored.

Let "$M = \{m_1, m_2, m_3 \ldots m_n\}$" be a set of distant directories at which the file was stored in 13 weeks or since its creation, where "m_i" denote a particular directory and "S_{m_i}" denote the total time span of the file in that directory. Then, the threshold "x" that represents the average time span of the file in a directory is defined as given in Equation (6).

$$x = \frac{\sum_{i=1}^{n} S_{m_i}}{n} \tag{6}$$

The threshold "x" is used to determine the candidacy status of a file for the NOW directory.

Hence, if "$S_{m_i} \geq x$," then the current directory "m_i" is considered as a related directory for the file and a candidate directory for the NOW.

Furthermore, when relating a file to its previous directory, two other relationships can also be deduced: directory to directory (i.e., relating the current and previous directories of a file) and file to files (i.e., relating a file to the files that are currently saved in its previous directory).

Therefore, if a user accesses a file in this way, the system can recommend its previous directories and the files stored in previous directories. For instance, the file "*360°-SFS.pdf*" is saved in a directory named as *"File System"*, and a user later moves it to the *"Semantic File System"* directory and then to the *"Research Papers"* directory. Therefore, after retrieving the "*360°-SFS.pdf*" file, the system can recommend its previous directories (i.e., *"File System"* and "*Semantic File System"*). Similarly, all three directories can be declared as related as well. Additionally, the file *"360°-SFS.pdf"* can also be directly related to the files saved within the *"File System"* and "*Semantic File System"* directories.

D. FILE ACCESS PATTERNS

If a file is accessed in a specific time interval, right after accessing another file, then both will be considered contextually related to each other. Similarly, if two files "A" and "B" are open while another file "C" is newly created, then all three are considered to be related, as also suggested in [19] and [20]. In our proof of concept implementation, we relate two files if they are accessed together or created within a predefined temporal duration (also called the context scope) by a human user. The study conducted in [20] states that choosing a minimum value (5 min) for the context scope generates less accurate relationships. Meanwhile, a maximum value (30 min) generates a high number of relationships but will cause an increase in the number false positives. Choosing a medium value (12 min) for the context scope could render a tolerable ratio between accuracy and the number of semantic relationships. Therefore, in our experiment, we decide to set the value of the temporal duration as 12 min.

Let "$P = \{p_1, p_2, p_3, \ldots p_n\}$" be a set of files accessed together with file "A" within duration of 12 minutes. Where "p_i" denote a distinct file and "F_{p_i}" denote the access frequency of that file together with file "A."

Then, the threshold value "u," which is the average access of file "A" with other files, is given in Equation (7).

$$u = \frac{\sum_{i=1}^{n} F_{p_i}}{n} \tag{7}$$

From the value of "u," the system computes whether to relate two files with the *"frequentlyAccessedWith"* property or not.

Hence, if "$F_{p_i} \geq u$," then file "p_i" is declared as *"frequentlyAccessedWith"*file "A," and is a candidate file for the NOW directory.

We declare the *"frequentlyAccessedWith"* property as symmetric. Therefore, if file "p_i" is *"frequentlyAccessedWith"* file "A," then the reasoner will deduce that file "A" is also *"frequentlyAccessedWith"* file "p_i."

E. CONTENT SIMILARITY

Suppose that a user downloads files from the Web and later forgets their location on the storage device. This makes them download the same files again, thus resulting in wasted time and bandwidth along with multiple copies of the same files. In addition, a user sometimes maintains several versions of the same file for different purposes. Therefore, after accessing one file, the system recommends its duplicate copies and earlier versions of the file to the user. The system relates 100% of the identical files via the "*duplicateOf*" relationship, whereas the files with similarity less than 100% and greater than 70%

are related through the "*nearDuplicateOf*" relationship of the ontology.

F. MANUAL ANNOTATIONS AND TAGS

The system also allows the manual setting of the temporal and geographical location based on the alerts on files. When the condition matches, these files are shown in NOW directories independent from their current path (territory). Since manual annotations are more accurate than automatic annotations, the system gives importance to these by showing them in all NOW directories.

Attributes could be manually added by renaming the file and adding double underscore symbols followed by the attribute name and value. After renaming the file, the system adds attributes accordingly and then automatically renames the file back to its original name. For instance, renaming a file as "*FileName__SetManualPeakLocation: University of Peshawar*" and "*FileName__SetTag: Review Meeting*" will set the manual peak location and tag the file accordingly. Similarly, any attribute that is set either manually or automatically could be removed by renaming the file as "*Filename__RemoveManualPeakLocation: Peshawar*," "*Filename__RemovehasTag: Review Meeting*," etc.

The special directory TAGS makes the file retrieval easier. Like in NOW directory, TAGS is also displayed within every directory. A TAGs directory contains subdirectories that are actually the manual tags that the user had assigned to the files via "*FileName__SetTag:*." Renaming and deleting the subdirectory respectively renames and deletes the tag in the RDF triplestore. For instance, the *Review Meeting* subdirectory within the TAGs directory will show the symbolic links to the files tagged with *Review Meeting*. Deleting the *Review Meeting* subdirectory will delete the *Review Meeting* tag from all particular files to which the same tag is assigned. An alternate way to tag a file is the copying of a file to a tag subdirectory within the TAGs directory. If a particular tag does not exist in TAGs, then a user can create a new tag by creating a new subdirectory within TAGs.

As a result, the system automatically keeps track of users' interactions with file system resources and recommends files accordingly. To maintain privacy, if a user deletes the symbolic link of a file in a NOW directory, then all its entries in the RDF triplestore will be deleted. Likewise, if a user wants to permanently stop the automatic recommendation of certain files or directories, then a user can change the status of that particular file or directory to private. Privacy status can be changed by renaming a file or directory as "*FileName__SetStatus: Private*," "*FileName__SetStatus: Normal*" or "*DirectoryName__SetStatus: Private*," among others.

G. TERRITORY

A territory is a limit on the set of files presented in the NOW and TAGs directories, that is given by the path already navigated through by the user. The idea of territory is introduced to control the number of items in the NOW and TAGs directories by adopting the generalization/ specialization approach. The NOW directory on the root shows the candidate files that are aggregated from the entire file system hierarchy. A NOW directory within a directory excludes the candidate files stored in parent directories (at upper levels). It only includes the candidate files stored in that particular directory and in its child directories. This means that if a user navigates down in the hierarchy, then the number of files in the respective NOW directory will shrink, and if a user goes up in the hierarchy, then the number of files in the respective NOW will grow.

For instance, if a user navigates to a directory "*/books/semanticWeb/semanticFileSystems/*," then the NOW directory in the same path will show the files that are saved in the mentioned path and in the child directories of the same path. All files stored in "*/books/semanticWeb/*," "*/books/*" or on the root will be excluded from the NOW directory.

Each time a user opens a directory/subdirectory, it is considered as an input from the user. We give more preference to the user's input than the machine. If a user chooses to go inside a directory, it means the she is narrowing down her search for files, and therefore the system tries to assist her by narrowing that recommendation. However, the restriction of territory only applies to temporal and geographical location based contextually related files. It does not apply to the relationships that are created between files and directories via content similarity, access patterns or file movement.

Like NOW directories, TAGs directories are also territory dependent. A TAGs directory shows all the tags that are within its territory. This enables a user to narrow down the list of candidate items in the NOW and TAGs directories. The dashed rectangles in FIGURE 1 indicate the territory of each NOW and TAGs directory. The candidate files that were excluded from the display in NOW at the upper levels because of the lower ranking scores, might find a place in NOW at the lower level directories down the hierarchy.

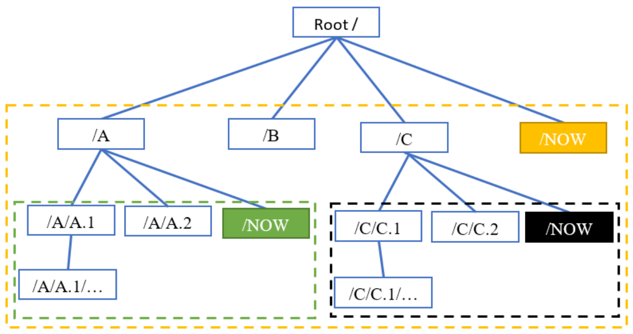


FIGURE 1. Territories.

H. CUSTOM VIRTUAL DIRECTORIES

In traditional file systems, if a user needs to bring together some of the files in a separate directory from different locations of the file system's hierarchy, then they do it manually.

TABLE 2. 360°-SFS operations.

Operation	System Performs
Create new file	Creates an individual of the File class *(nfo:LocalFileDataObject)*and annotates it accordingly…
Create new directory	Creates an individual of the directory class *(nfo:Folder)* and annotates it accordingly…
Delete file A	Deletes the individual of the File class and all its annotations…
Delete directory A	Deletes the individual of the directory class and all of its annotations; Delete all the individuals of the directory and File classes along with their annotations whose parent is A…
Rename directory A	If the new name contains double underscore symbols, then generate the SPARQL query accordingly to generate the contents for directory A …
Delete subdirectory of TAGs directory	Delete the statements where the value of the *hasTag* property is the name of the deleted subdirectory.
Rename subdirectory of TAGs directory	Update the literal value of the *hasTag* property in the statements where the *hasTag* value was the old name of the directory.
Open file	Annotate the respective individual of File class accordingly.
…	…

Algorithm 1 Ranking

```
    For each group of candidates of NOW
Directory
Temporal:
Show top five items based on highest
"$F_h$" value
Geographical:
Show top five items based on highest
"$F_{gi}$" value
File movement:
Show top five items based on highest
"$S_{m_i}$" value
Access pattern:
Show top five items based on highest
"$F_{pi}$" value
End
```

With the help of custom virtual directories, our proposed system allows a user to automatically aggregate files from the entire file system on the basis of any or a combination of file attributes. These are created like normal directories (*Right click > create New Directory*), but their names must be post fixed with a user-friendly query. The name and query parts are separated by double underscore symbols. However, a user can leave the first part blank. The Query Transformer module rewrites these user-friendly queries into SPARQL queries accordingly. The resulting files become the contents of the custom virtual directory.

For instance, a directory named "*DirectoryName__filePeakLocation: University of Peshawar*" aggregates the files with University of Peshawar as the peak geographical location and the "*DirectoryName__hasTag: Review Meeting*" obtains files tagged with Review Meeting.

The "." "+" and "!" symbols are used for AND, OR and NOT arithmetic operators, respectively. For instance, "*DirectoryName__hasTag:Review Meeting + hasTag: Peshawar! filePeakHour: 19*" gets files tagged with either *Review Meeting* or *Peshawar* but whose *filesPeakHour* is not 19:00.

I. RANKING IN THE LIST OF CANDIDATE FILES

From the list of candidate files, the system includes only top-ranked files in the NOW directory. The algorithm for selecting toped ranked files is provided as follows:

The algorithm shows the selection of the top five files in each category, but application developers in the respective interfaces of their applications can increase or decrease the number of files according to their requirements via 360°-SFS API.

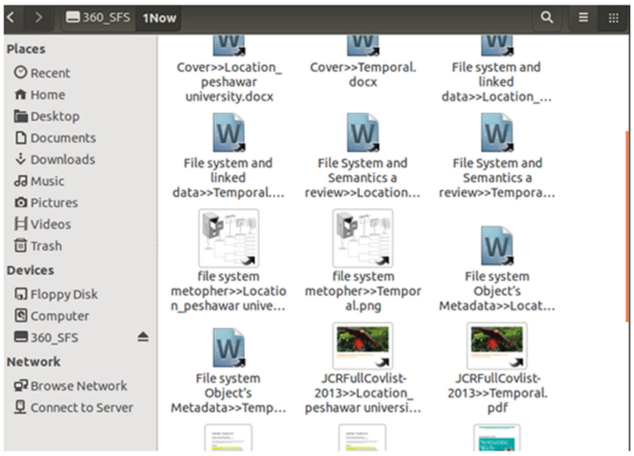
J. 360°-SFS API FOR APPLICATIONS

API based access is also provided for application developers. Through different API calls, applications like file managers, desktop search engines and word processors can show relevant files to users in their respective interfaces. For instance, to show the semantically related files to a user within the word processor interface, a customized word processor plugin can be developed using 360°-SFS API. Additionally, applications can also show the files of NOW along with the ROF list within the application.

IV. IMPLEMENTATION

We develop the ontology based 360°-SFS that automatically creates objects of classes and annotates them while the user interacts with the file system's resources. Table 2 depicts some of the 360°-SFS operations. In our proof of concept implementation, we store the actual files in a directory of the underlying file system and their metadata in an RDF store. The contents of NOW and other virtual directories are symbolic links to the actual files. The problem with symbolic links is that they become dead when the original file is deleted, moved or renamed. In our case, these links are not permanently stored but are created on the fly each time that NOW and other virtual directories are browsed. This solves the problem of dead symbolic links.

The names of the NOW and TAGs directories could be prefixed with a symbol or a number so that these are displayed on top of other directories and files. The names of the files in NOW are affixed with user friendly keywords that indicate the reason for their inclusion. For instance, the file name "*semWeb* ≫ *University of Peshawar.pdf*" indicates that the file "*semWeb*" is included in NOW because a user frequently accesses it at the user's current geographical location (see FIGURE 2).



360_SFS
1Now
Places
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Devices
Floppy Disk
Computer
360_SFS
Network
Browse Network
Connect to Server
Cover>>Location_peshawar university.docx
Cover>>Temporal.docx
File system and linked data>>Location_...
File system and linked data>>Temporal....
File System and Semantics a review>>Location...
File System and Semantics a review>>Tempora...
file system metopher>>Location_peshawar unive...
file system metopher>>Temporal.png
File system Object's Metadata>>Locat...
File system Object's Metadata>>Temp...
JCRFullCovlist-2013>>Location_peshawar universi...
JCRFullCovlist-2013>>Temporal.pdf

FIGURE 2. Contents of a NOW directory in a traditional file manager.

We implement 360°-SFS in the user space using FUSE Java binding, Apache Jena, Protégé and Java API for Google Geocoder. An offline Geocoding database or custom local cache could be used to accelerate the process. In our implementation, we extract the location information from the IP address, but the GPS receiver of a user's personal smartphone can also be used for more accurate results.

FIGURE 3 depicts the architecture of 360°-SFS. Both the SFS-Ontology [39] and prototype of the 360°-SFS are available under a Creative Commons Attribution license from https://w3id.org/sfs-ontology and https://w3id.org/360-sfs respectively.

V. EXPERIMENTAL SETTINGS

Initially, we approached *115* users, most of them were students and employees at the University of Peshawar. Out of these *115*, *7* users were dropped because of losing interest, absences, or biased data. Finally, an evaluation was done with *108* (*25* females, *83* males) highly motivated volunteers. The median age of participants was *32.7*, within the range of *23-38* years. Based on the subjective analysis of the expertise of participants, they were categorized as intermediate, advance and experts. Table 3 summarizes general information about participants along with other indicators that include information related to their education level, age, gender and computer usage experience. The volunteers in this study have given written informed consent. The Institutional Review Board (IRB) of the authors' institution has approved the consent procedure for this study.

The evaluation was done with pre- and post-experiment surveys on the same participants. After orienting users on

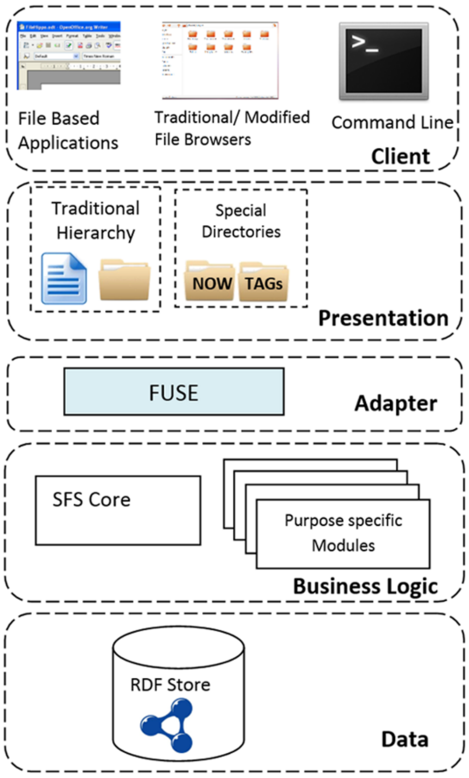


FIGURE 3. 360°-SFS architecture.

TABLE 3. Participants' details.

	Group	Number of Participants	Percentage	SD
Gender	Female	25	23.15	41.01
	Male	83	76.85	
Age	22 to 30 Years	29	26.85	35.36
	31 to 40 Years	79	73.15	
Education Level	Undergraduate	7	6.48	28.51
	Graduate	37	34.26	
	Postgraduate	64	59.26	
Computer Usage Experience	5-10 Years	27	25.00	38.18
	11 and above Years	81	75.00	
Observed Expertise	Intermediate	22	20.37	19.29
	Advance	28	25.93	
	Experts	58	53.70	

360°-SFS, we copied each user's own files to 360°-SFS and asked them to continue their routine work using 360°-SFS. In addition to accessing files, users were also asked to manually annotate and tag files for later retrieval. Two weeks were given to users for their interaction with files so that the system would be amply trained, and enough metadata would be recorded for recommendation purposes. Then the participants were asked to start retrieving files, from the printed list of files that were retrieved during training period, in different context(s) by changing system time and geographical location. For the experimental evaluation, we have provided an option to users to manually set their pseudo geographical location. They were asked to retrieve files of their own choices but in three different ways (i.e., traditional directory navigation, ROF and 360°-SFS). Applications that were mainly used were: Files (File Manager) 3.10, LibreOffice 4.2, GNOME Rhythmbox Audio Player 3.0, GNOME Videos (Totem) Movie Player 4.10, GNOME Image Viewer 3.10, Gedit 3.10, Document Viewer 3.10 and GNOME Terminal 3.6 of the Ubuntu 14 Operating System.

In our experiment, we evaluated the aforementioned modes of file retrieval on the basis of human efforts. While navigating to the desired files, the human effort was calculated based on the time and navigational steps taken. The time taken, and navigational steps followed were recorded manually by the authors for each user for all modes of retrieval. We observed that if exact location of a file is known to a user then she retrieves it within a minute, otherwise she starts brute-force navigation for the desired file. The purpose of experiment was to compare retrieval of the files on which the system was trained and the location of which were known to the participants. Hence, the entry was discarded if a user was taking more than a minute while retrieving or was failing to retrieve a file via any of file retrieval mode.

On average *1,715* files with average depth of *2.9* ± *0.6* directories were copied per user to the 360SFS having average depth of files in directory tree was. On average 61 files were retrieved in training phase and 9.41 files were successfully retrieved per user in testing phase (after the training period).

In the case of file manager, opening a file in the current directory "*./file*" was counted as a single navigational step, and retrieving a file from the path "*./directoryA/directoryB/file"* was counted as three steps. Similarly, each command entered on the command line by a user from root directory was recorded as a single navigational step, retrieving a file from the path "*./directoryA/directoryB/file"*was counted as three steps. In the case of ROF, a single action of a user was recorded as a single navigational step. For example, opening a file from "*file menu > m recent sub menu > file in ROF List*" was counted as three steps. Time access of a file was measured from the point when a user was asked to retrieve a specific file from the printed list of files till the last action on the target file. Based on the evidence collected, Section VI presents a brief analysis regarding the performance of the 360°-SFS.

VI. RESULTS AND DISCUSSIONS

Table 4 compares all the three modes of file retrieval based on the time taken and the navigational steps followed. The results show that on the average traditional directory navigation requires more human effort (*4.1* ± *1.4* steps) and time (*22* ± *14* seconds) than the 360°-SFS (*2.2* ± *0.4* steps, 11 ± 03 seconds) and ROF (*2.8* ± *0.3* steps, *09* ± *04* seconds). On the average, 360°-SFS gets *50%* improvements in retrieval time over traditional navigation. However, it falls *22.22%* behind the ROF. In terms of number of steps, 360°-SFS requires *22.46%* and *46.36%* less steps than ROF and traditional navigation respectively. The main purpose of the 360°-SFS was to improve files retrieval by extending traditional directory navigation, and the results clearly show improvements.

TABLE 4. Results of files that are successfully retrieved via all three modes of file retrieval.

	Time Taken				
	N (files)	Min	Max	Mean	SD
360°-SFS Navigation	1017	0:00:04	0:00:47	0:00:11	0:00:03
ROF		0:00:04	0:00:25	0:00:09	0:00:04
Traditional Navigation		0:00:04	0:00:59	0:00:22	0:00:14
	Steps Taken				
360°-SFS Navigation	1017	2	5	2.21	0.447
ROF		2	3	2.85	0.353
Traditional Navigation		1	9	4.12	1.456

To verify whether these differences are statistically significant we have fed the same data to the SPSS (version 20), using three variables namely *"Time_Taken"*, *"Steps_Taken"* and *"Retrieval_Method"*. For comparison, the null and alternative hypothesis are:

$$H_0 : \mu_{360°-SFS} = \mu_{ROF} = \mu_{Traditional\_Navigation}$$
$$H_1 : \mu_{360°-SFS} \neq \mu_{ROF} \neq \mu_{Traditional\_Navigation}$$

Applying the one-way ANOVA test on confidence interval of *95%* resulted in significance value (p-value) of *<0.001* for both *"Time_Taken"* and *"Steps_Taken"*, which is *<0.05*, hence, rejects the null hypothesis in favor of alternative and shows that the difference among all the three modes of file retrieval is statistically significant. The Tukey post hoc test with p-value of *<0.001* for both the variables (*"Time_Taken"* and *"Steps_Taken"*), which is *<0.05* further reveals that the difference between each two retrieval modes is statistically significant.

In the pre-experiment survey, one question was: "*out of six most recently accessed files, how many of files did you retrieve through ROF?*" The results show that on average *21%* of files are retrieved by ROF. The reason of its lower usage is that the ROF is maintained by every application separately and differently in its own respective interface. Some of the applications sort the ROF list based on recency, some sort them on the basis of frequency and some group them temporally (i.e., today, yesterday, last week, and older). Furthermore, the way of accessing them is not uniform across the applications, and the ROF is generally not accessible from the command line. The study conducted in [1] shows that on average, a user retrieves *56-68%* of the files via directory navigation. Since 360°-SFS augments traditional directory navigation, it is used as a companion to directory navigation and is hence more effective than the ROF. Furthermore, the ROF only shows the recently opened files, while 360°-SFS enables the retrieval of files based on different semantics that were discussed earlier.

One of the objectives of the study was to reduce the cognitive attention while navigating. The study conducted in [6] shows that the deeper in the hierarchy the file is stored, the more cognitive attention that the navigation requires. 360°-SFS augments the traditional navigation and enables the retrieval of the desired file in 2.2 ± 0.4 steps in a location independent way. Hence, it helps in reducing the cognitive attention of a user. To further support this evidence, some statistical tests were performed upon the data collected through the post-experiment survey. The users were asked about the difficulty level because a difficult system requires comparatively more attention than a simpler one. To assess the difficulty, the questions asked include "*Q1: How difficult is it for you to find the file via traditional directory navigation?*"and "*Q2: How difficult is it for you to find the file via 360°-SFS?*." They were answered on a scale from 1-5 (1 low, 2 minimal, 3 neither high nor low, 4 some, and 5 high), which were ultimately fed to the SPSS (version 20), using two parameters (variables) namely *"Traditional_Navigation"* for Q1 and "*SFS*" for Q2. For comparison, the null and alternative hypothesis are:

$$H_0\text{: } \textit{The } 360^\circ - \textit{SFS did not reduce cognitive attention}$$
$$H_1\text{: } \textit{The}360^\circ - \textit{SFS reduced cognitive attention}$$

Applying the Chi-Square test on the two parameters on confidence interval of *95%*, resulted in $\chi^2 = 20.76$ with p-value of *0.002* which is *<0.05*, hence, rejects the null hypothesis and shows that the proposed 360°-SFS significantly reduces the cognitive attention compared to traditional navigation.

However, we observed that participants did not take an interest in manual annotations and tagging. Only *15%* of the files were retrieved by participants via the 360°-SFS Tags directories. Retrieving files via a custom virtual directory was not part of the experiment as the concept was already discussed in [10]. During the experiment, we also noted that the desktop search engine crawlers and antivirus programs were also accessing files for their own purposes. For accurate recommendations, we were only needed to record the human interactions with files instead of machines. On the web, if a bot accesses a page, it can be easily distinguished since a log file is maintained on a server, but there is no such type of log maintained in a desktop. However, the accessing of files by a machine can be distinguished by different techniques. For instance, the file access pattern of a human is different from machine. A human access a file in one directory, takes some time, and then they open another file which is not necessary from the same directory. In contrast, a machine accesses more files in less time, possibly one after another and concurrently. In our implementation, to capture clean metadata, we defined a threshold of three seconds. Metadata was not recorded if the difference between the recently and the last accessed file was less than the threshold.

Because of the freely/opensource availability of the relevant APIs for Linux OS and their rich online support we had to implement the proposed system in Linux OS. But unfortunately, the region where this study was conducted, a common user normally uses Windows OS. Therefore, all the participants were native Windows OS users. So, to minimize the effect of Linux OS environment on the participants to greater extent, we majorly focused on expert computer users (*53.70%)* with above 10 years of computer usage experience (*75.0%)*. However, further user studies would enable better understand the effectiveness of the 360°-SFS.

For the sake of backward compatibility with traditional software applications, we proposed the virtual directories-based solution, which may overburden a user with extra directories. In addition, since we store metadata externally in an RDF store, the system inherits the drawbacks of external metadata. For instance, external metadata does not travel along with a file itself. It is lost if a file is copied to external media or sent via email [40]. Heuristics are needed to make external metadata travel along with a file. Since recommendations are done based on user's previous interactions, the system faces cold start issue.

VII. CONCLUSION AND FUTURE WORK

Previous studies have shown that only *4-15%* of files are accessed via desktop search engines [1]. This shows that people still prefer navigation over searching for file retrieval. This paper highlighted some of the limitations of navigation in traditional hierarchical file systems and presented an ontology-based 360° Semantic File System (SFS) to improve navigation. The 360°-SFS enables file retrieval by exploiting semantics, intentions and relationships to other files instead of its physical location, as in traditional file systems. Unlike existing SFSs, 360°-SFS neither replaces traditional file system metaphors nor does it involve extensive user manual annotations. Rather, it automatically learns from the behavior of a user and predicts files that the user is likely to retrieve next.

To achieve this, 360°-SFS exploits features that were ignored by the existing SFSs, such as file movements between directories, temporal and geographical locations, content similarity and territory. The evaluation of 360°-SFS shows that it outperforms the existing file systems (especially in navigation) in terms of access time, human effort and cognition attention. This enables the users to spend their valuable time in using the file rather than searching for it. The 360°-SFS comes with certain limitations, which we plan to resolve in its future prototypes. These are discussed in the next paragraph.

In the future we will be working to enable the system to recommend relevant directories while organizing local files and linking file system resources to Web of Data. A user often faces difficulties while organizing files. To make the machine assist a user while organizing files, the semantics of directories need to be known to machine. If a machine knows which directory holds what kind of contents, then it can recommend related directories to user while organizing. Similarly, a directory having name Tim Berners-Lee can be linked (i.e., using *owl:sameAs* etc.) to his URI or any other resource having Label "Tim Berners-Lee," or a music album directory in the file system can be linked to the URI of that album in an Linked Open Data (LOD) set. Linking local directories and files to LOD enables a user to explore additional related information, hence, dissolving borders between desktop and Web of Data.

REFERENCES

[1] O. Bergman, R. Beyth-Marom, R. Nachmias, N. Gradovitch, and S. Whittaker, "Improved search engines and navigation preference in personal information management," *ACM Trans. Inf. Syst.*, vol. 26, no. 4, 2008, Art. no. 20.

[2] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger, "The perfect search engine is not enough: A study of orienteering behavior in directed search," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2004, pp. 415–422.

[3] R. Boardman and M. A. Sasse, "'Stuff Goes into the computer and doesn't come out': A cross-tool study of personal information management," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2004, pp. 583–590.

[4] D. Kirk, A. Sellen, C. Rother, and K. Wood, "Understanding photowork," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2006, pp. 761–770.

[5] D. Barreau and B. A. Nardi, "Finding and reminding: File organization from the desktop," *ACM SIGCHI Bull.*, vol. 27, no. 3, pp. 39–43, 1995.

[6] O. Bergman, M. Tene-Rubinstein, and J. Shalom, "The use of attention resources in navigation versus search," *Pers. Ubiquitous Comput.*, vol. 17, no. 3, pp. 583–590, 2013.

[7] M. Rinck, "Document DNA: Distributed content-centered provenance data tracking," Ph.D. dissertation, Univ. Waikato, Hamilton, New Zealand, 2015.

[8] V. Vasudevan and P. Pazandak. (1997). *Semantic File Systems*. [Online]. Available: http://www.objs.com/survey/OFSExt.htm

[9] N. Popitsch, "Building blocks for semantic data organization on the desktop," Ph.D. dissertation, Univ. Vienna, Vienna, Austria, 2011.

[10] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O. James, "Toole, "Semantic file systems," presented at the 13th ACM Symp. Oper. Syst. Princ., Pacific Grove, CA, USA, 1991.

[11] C. K. Hess and R. H. Campbell, "An application of a context-aware file system," *Pers. Ubiquitous Comput.*, vol. 7, no. 6, pp. 339–352, 2003.

[12] L. Sauermann, A. Bernardi, and A. Dengel, "Overview and outlook on the semantic desktop," in *Proc. 1st Workshop Semantic Desktop ISWC Conf.*, 2005, pp. 74–91.

[13] H. B. Ngo, C. Bac, F. Silber-Chaussumier, and T. Q. Le, "Towards ontology-based semantic file systems," in *Proc. IEEE Int. Conf. Res., Innov. Vis. Future*, Mar. 2007, pp. 8–13.

[14] Tx0. (Feb. 2007). *Tagsistant: Semantic File System*. [Online]. Available: http://www.tagsistant.net

[15] S. Bloehdorn. (Feb. 2009). *SemFS—Semantic File System*. [Online]. Available: http://semanticweb.org/wiki/SemFS

[16] S. Bloehdorn, O. Görlitz, S. Schenk, M. Völkel, and F. I. Karlsruhe, "TagFS—Tag semantics for hierarchical file systems," presented at the 6th Int. Conf. Knowl. Manage. (I-KNOW), Graz, Austria, 2006.

[17] N. Albadri, R. Watson, and S. Dekeyser, "TreeTags: Bringing tags to the hierarchical file system," presented at the Australas. Comput. Sci. Week Multiconf., Canberra, ACT, Australia, 2016.

[18] S. Ames *et al.*, "LiFS: An attribute-rich file system for storage class memories," in *Proc. 23rd IEEE/14th NASA Goddard Conf. Mass Storage Syst. Technol.*, 2006, pp. 1–14. [Online]. Available: http://storageconference.us/2006/Papers/2006-006-Ames.pdf and http://www.ssrc.ucsc.edu/Papers/ames-mss06.pdf

[19] C. A. N. Soules and G. R. Ganger, "Why can't I find my files? New methods for automating attribute assignment," in *Proc. HotOS*, 2003, pp. 115–120.

[20] A. Okoli and B. Schandl, "Extraction of contextual metadata from file system interactions," presented at the Workshop Exploitation Usage Attention Metadata, Lübeck, Germany, 2009.

[21] S. Ames, M. Gokhale, and C. Maltzahn, "QMDS: A file system metadata management service supporting a graph data model-based query language," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 28, no. 2, pp. 159–183, 2013.

[22] M. Geel, "Memsy: A personal resource management infrastructure," Ph.D. dissertation, ETH-Zurich, Zürich, Switzerland, 2015.

[23] J. Chou, "FindFS: Adding tag-based views to a hierarchical filesystem," M.S. thesis, Univ. British Columbia, Vancouver, BC, Canada, 2015.

[24] B. Schandl and B. Haslhofer, "Files are siles: Extending file systems with semantic annotations," *Int. J. Semantic Web Inf. Syst.*, vol. 6, no. 3, pp. 1–21, 2010.

[25] B. Schandl and B. Haslhofer, "The sile model: A semantic file system infrastructure for the desktop," presented at the 6th Eur. Semantic Web Conf., Heraklion, Greece, 2009.

[26] B. Schandl and N. Popitsch, "Lifting file systems into the linked data cloud with TripFS," in *Proc. 3rd Int. Workshop Linked Data Web (LDOW)*, Raleigh, NC, USA, 2010, pp. 1–8. [Online]. Available: http://ceur-ws.org/Vol-628/ldow2010_paper02.pdf

[27] B. Mizrachi and L. S. Deluca, "File folder display," U.S. Patent 2017 0 109 010, Apr. 20, 2017.

[28] D. Di Sarli and F. Geraci, "GFS: A graph-based file system enhanced with semantic features," in *Proc. Int. Conf. Inf. Syst. Data Mining*, 2017, pp. 51–55.

[29] S. Schenk, O. Görlitz, and S. Staab, "TagFS: Bringing semantic metadata to the filesystem," in *Proc. 3rd Eur. Semantic Web Conf. (ESWC)*, 2006, pp. 1–2.

[30] D. Garg, V. Mehta, S. Pandit, and M. De Rosa, "A writable semantic file system," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. 4, 2005.

[31] B. Rector. (Mar. 2004). *WinFS*. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479870.aspx

[32] B. Schandl, "Representing linked data as virtual file systems," in *Proc. 2nd Int. Workshop Linked Data Web*, Madrid, Spain, 2009, pp. 1–9. [Online]. Available: http://events.linkeddata.org/ldow2009/papers/ldow2009_paper1.pdf

[33] Y. Padioleau, B. Sigonneau, and O. Ridoux, "LISFS: A logical information system as a file system," in *Proc. 28th Int. Conf. Softw. Eng.*, 2006, pp. 803–806.

[34] P. Dourish, W. K. Edwards, A. LaMarca, and M. Salisbury, "Presto: An experimental architecture for fluid interactive document spaces," *ACM Trans. Comput.-Hum. Interact.*, vol. 6, no. 2, pp. 133–161, 1999.

[35] W. Jones, A. J. Phuwanartnurak, R. Gill, and H. Bruce, "Don't take my folders away!: Organizing personal information to get ghings done," in *Proc. Extended Abstr. Hum. Factors Comput. Syst. (CHI)*, 2005, pp. 1505–1508.

[36] O. Bergman, N. Gradovitch, J. Bar-Ilan, and R. Beyth-Marom, "Folder versus tag preference in personal information management," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 64, no. 10, pp. 1995–2012, 2013.

[37] K. Voit, K. Andrews, and W. Slany, "Why personal information management (PIM) technologies are not widespread," in *Proc. PIM Workshop, ASIS&T*, 2009, pp. 60–64.

[38] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, "A five-year study of file-system metadata," *ACM Trans. Storage*, vol. 3, no. 3, 2007, Art. no. 9.

[39] S. R. Mashwani and S. Khusro, "The design and development of a semantic file system ontology," *J. Eng., Technol. Appl. Sci. Res.*, vol. 8, no. 2, pp. 2827–2833, 2018.

[40] S. Khusro, S. R. Mashwani, A. Rauf, S. Mahfooz, and S. Ali, "A study of file system objects metadata," *Life Sci. J.*, vol. 10, no. 11, pp. 343–348, 2013, Art. no. 63. [Online]. Available: http://www.lifesciencesite.com/lsj/life1011s/ and http://www.lifesciencesite.com/lsj/life1011s/063_21572life1011s_343_348.pdf

SYED RAHMAN MASHWANI received the B.S. and M.S. degrees in information technology from the Institute of Business & Management Sciences, Agricultural University Peshawar, Peshawar, Pakistan, and the Ph.D. degree from the Department of Computer Science, University of Peshawar, Pakistan. His research interests include file systems, information semantics, information retrieval, ontology engineering, and linked open data.

SHAH KHUSRO received the M.Sc. degree (Hons.) from the Department of Computer Science, University of Peshawar, Peshawar, Pakistan, and the Ph.D. degree from the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Vienna, Austria. He is currently a Professor of computer science with the Department of Computer Science, University of Peshawar. His research interests include information semantics, information retrieval, web engineering, recommendation, search engines, ontology engineering, and accessibility. He is a member of the IEEE, IEICE, and ACM.