

# Favorite Folders: A Configurable, Scalable File Browser

Bongshin Lee, Benjamin B. Bederson

Computer Science Department

Human-Computer Interaction Lab

University of Maryland

College Park, MD 20742, USA

Tel: 1-301-405-7445

E-mail: {bongshin, bederson}@cs.umd.edu

## ABSTRACT

Microsoft Windows Explorer, the most widely used file browser in Microsoft Windows, shows almost all directories in the file system. However, most users usually access only a subset of the directories in their machine. If the file browser shows only the directories users are interested in, they can select the directory they want more easily and quickly.

This paper introduces a configurable, scalable file system explorer that reduces selection time by showing only the directories users want to see. We give users an easy way to hide directories behind a special *ellipsis node*. In addition, those hidden directories are one click away.

We present a preliminary field study conducted to validate the concept of Favorite Folders and a theoretical model to predict the performance times.

**KEYWORDS:** Windows Explorer, file browser, adaptive interfaces, customizable interfaces

## INTRODUCTION

A typical modern computer has a complex file system structure, involving many directories. As users are added to a multi-user machine and over time, the file system structure becomes more complex. However, a particular user may be interested in only some of these directories. Yet, the most widely used file browser in Microsoft Windows, Windows Explorer (Figure 1), provides the user with a fixed view of the file system structure. It displays nearly all directories in the file system, distracting the user and making the task of selecting a directory cumbersome.

There have been a number of adaptive systems to improve selection performance by providing a small subset of all alternatives as with Microsoft Office's adaptive menus. Researchers advocating adaptive systems believe that they are creating something lifelike and smart and assume users would be attracted to adaptive systems [12]. In addition, adaptive systems have the advantage that they don't require additional work by the user to set it up. However, users may experience a loss of control and frustration because the adaptive systems are unpredictable.

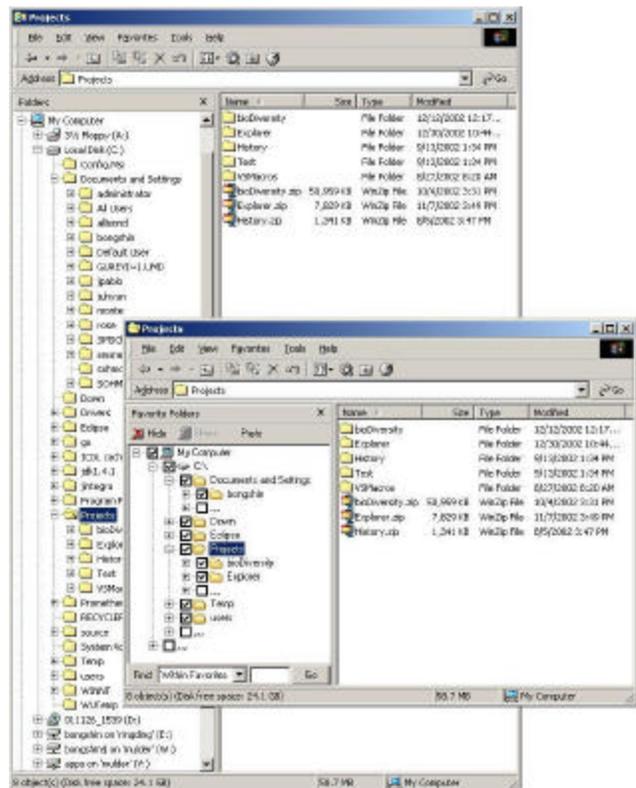


Figure 1. Windows Explorer vs. Favorite Folders

We have developed a technique we call Favorite Folders (Figure 1) that offers similar functionality but keeps the user in control by providing an easy-to-understand adaptable, customizable interface.

In this paper, we describe the design and implementation of Favorite Folders, which allows users to easily and quickly access the directories most relevant to them (typically, they are the ones most frequently accessed). In this paper, we use the term ‘favorite folders’ to denote those directories that a user would like to see. We introduce a manual strategy enabling users to specify themselves whether or not they want to see a directory.

We then explain how we implemented Favorite Folders as an explorer bar for Microsoft Internet Explorer to provide a display area adjacent to the browser pane. It stores necessary information in a relational database and indexes directories by name for fast access. Therefore, it provides fast search capabilities for directories. It watches the file system to update the view properly whenever the file system changes.

We have conducted a preliminary field study intended to validate the concept of Favorite Folders and to get an idea of what users think about Favorite Folders and how users manage and access their file systems in order to inform us future directions. In addition, we describe a model to predict the amount of time saved by the use of Favorite Folders.

## RELATED WORK

There have been a number of research and commercial products to improve performance by making frequent tasks easier or providing small subsets out of all alternatives.

Sears and Shneiderman created split menus, which splits a menu into two sections. Frequently selected items are placed in the top section and infrequently selected items in the bottom section. They described how a logarithmic model applies to high-frequency items, and a linear model to low-frequency items. They found performance times were reduced by 17% to 58% [11].

Debevc, Meyer, and Svecko described an adaptive short list for documents on the web. Their work presents the user with a small set of sites that represents the most commonly used sites based on the user’s history of web use. The adaptive short list is automatically maintained and updated by a decision algorithm. For the most common documents, users get the desired documents quickly because they don’t have to go through the entire list. [2]

Miah, Karageorgou and Knott presented a technique for automatically adapting toolbars to user needs. They calculated a toolbar’s importance from the three variables: time of creation, time elapsed since last interaction, and the frequency of interaction to determine which toolbar would be displayed. [9]

Kaasten and Greenberg developed a revisitation system integrating back, history and bookmarks in web browsers [5]. Their work is based on the following two observations – (1) about 60% of the web pages a person sees are revisits and (2) recency is an excellent predictor of what pages a person is likely to revisit [13]. All visited web pages are organized as a recency-ordered history list, with duplicate pages shown only in their latest position. Therefore, users can expect to find a page they had recently visited near the top of the list.

Microsoft also introduced adaptive interfaces for Office 2000 and the Programs submenu of the Windows Start menu. When users first start any office program, the menus and toolbars display basic commands and buttons. As users work with the program, the commands and buttons that users use most often are stored as personalized settings and displayed on menus and toolbars. Users can look for a menu command that doesn’t appear by clicking the arrows at the bottom of the menu or hovering over them for a few seconds. The menu expands to show more commands. Any command that users click on the expanded menu is added immediately to the personalized (short) version of the menu. The program stops showing a command on the short version of the menu if users stop using it for a while. [15]

Although the adaptive interfaces provide short versions of the menus by showing only basic and frequently used commands, they often cause a loss of control and their behavior is hard to predict because they automatically decide which menu options to show. In fact, users can only guess the underlying algorithms based on the behaviors of the adaptive interfaces. Hence, users are often frustrated when the systems don’t work as they expected. To address these problems, we have developed a technique that offers similar functionality but keeps the user in control by providing an easy-to-understand adaptable, customizable interface.

McGrenere, Baecker, and Booth designed an adaptable interface, which can be personalized by the user for heavily featured productivity software. They point out that menus and toolbars have grown quickly as windows applications such as the word processor and the web browser are getting more complex. They proposed a personal interface adaptable by the user with an easy-to-understand adaptation mechanism. Their system keeps the user in control by providing an adaptable interface, which can be personalized based on the user’s need. [8] Their idea is similar to the manual strategy presented in this paper.

However, menus and toolbars do not grow as users use the application like file systems that get bigger as users use their machines. Menus are no match for file system structure in size. Furthermore, their customization process needs four steps to add a function: 1) Press ‘Modify Interface’ button to open a dialog, 2) Press ‘Add’ button to switch to full interface, which has all the menus and toolbar buttons, 3) Select the toolbar button you wish to add, and 4) Press ‘OK’

button to confirm your selection. Therefore, their approach is not applicable for the file system browser.

## FAVORITE FOLDERS

Favorite Folders is a new customizable Windows file system browser, only showing users the favorite folders by default.

Customization, in general, is a process where users alter a software environment according to individual specifications. Although more and more programs are customizable by the end user, many people do not take advantage of the customization feature if the customization process is complex or if it takes time to learn and perform the customization. Similarly, users are not likely to spend time annotating all directories in the file system for future benefit if it cannot be done easily. We propose a new interface enabling users to customize the program with minimal effort and immediate benefits.

### Hidden Folders

An ellipsis is a mark (...) indicating an omission. To hide irrelevant directories, Favorite Folders introduces a special *ellipsis node* in the directory tree structure. It is a virtual container hiding items that are not shown. The existence of the ellipsis node indicates that there are hidden directories. Opening the ellipsis node redisplay those hidden items. It makes the Favorite Folders design unique because users can see all directories in the file system immediately without switching to the regular interface.

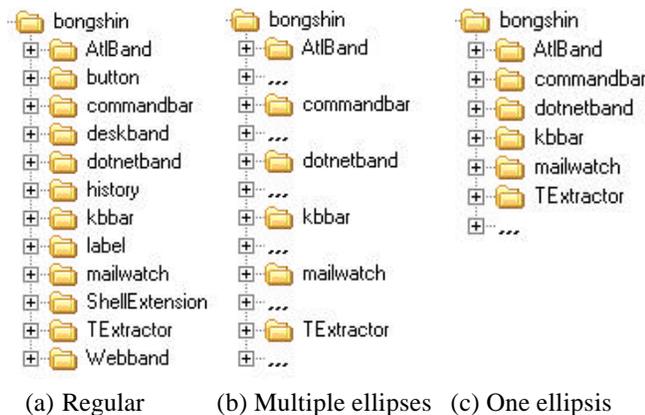


Figure 2. Two styles of ellipsis node

Since directories are displayed in alphabetical order, our system provides two ways to hide irrelevant directories as shown in Figure 2. One way is to preserve the hidden directory's original location. In this case, even though each contiguous sequence of items is hidden in one ellipsis node, hiding multiple directories can end up with several ellipsis nodes. If the directory to be shown and the directory to be hidden alternate, Favorite Folders would be crowded with the ellipsis nodes. Instead, we store all hidden items in one ellipsis node and place the ellipsis node at the bottom of the list as is done in the Microsoft adaptive menus.

## Manual Strategy

One of the key issues with the Favorite Folders design is how to choose the favorite folders to be shown. Since what a user wants is impossible to predict perfectly and it changes over time, we believe that the best way is to enable users to fully control the behavior of the program. In Favorite Folders, users specify themselves whether they want to see a directory or not.

If the customization process is complex or if it takes time to learn and perform the customization, users are unlikely to make use of such features. To minimize user effort, we put a check box at the left of the directory name as shown in Figure 3. With a single click, users can hide the irrelevant directories.

When users first select a directory, the program displays all of its subdirectories. When users uncheck the irrelevant directory, Favorite Folders hides it within the ellipsis node and only displays the directories still checked.

To be a favorite folder, a directory and all of its ancestors must be checked. If users uncheck a directory, then it and all of its descendants are no longer considered a favorite folder.

It is important to note that Favorite Folders doesn't require users to annotate all directories before they use the system. Favorite Folders integrates the annotation process into the user's navigation and supports updating of favorite folders incrementally. Whenever users find an irrelevant directory either by browsing or by searching, they can hide it with a single click and immediately get a better view having a smaller number of alternatives. We expect that some users will be willing to spend some time setting up the favorite folders. We also expect that it will not take much time to reach a stable state in which the favorite folders are almost set up and therefore there will be little need for users to check/uncheck directories after some initial use period.

Users can hide the check boxes to save space after Favorite Folders reaches a stable state. Figure 3 shows Favorite Folders with and without check boxes. We can see that they are much shorter than the Folders view in Windows Explorer in Figure 1.

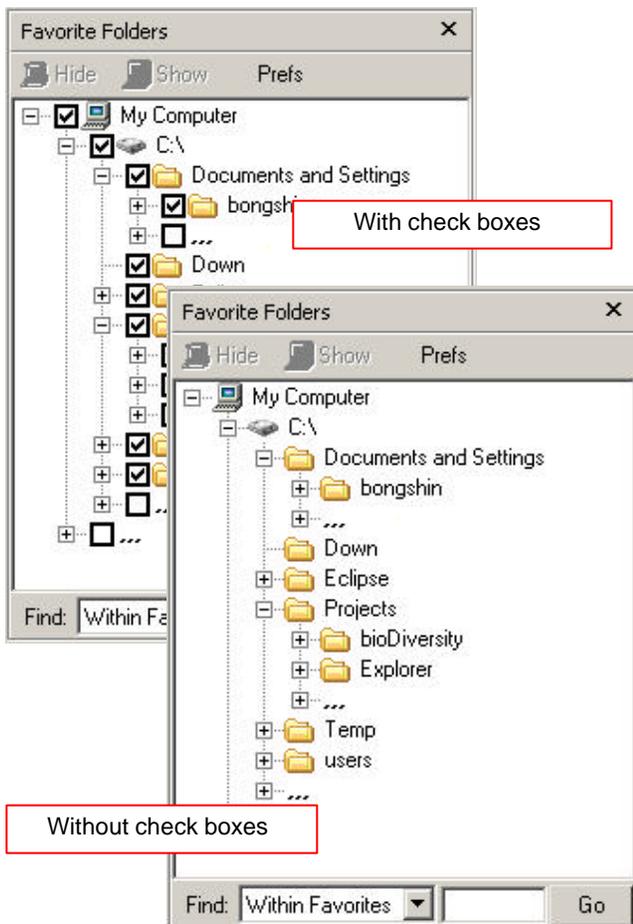


Figure 3. Favorite Folders . Check box visibility is controlled through a context menu.

**IMPLEMENTATION**  
**Explorer Bar**

Favorite Folders was implemented in C# as an *Explorer Bar*, an interface element introduced with Microsoft Internet Explorer 4.0 to provide a display area adjacent to the browser pane. It is basically a child window within the Internet Explorer window, and it can be used to display information and interact with the user [14].

Internet Explorer provides several standard Explorer Bars, including Favorites, History and Search. In particular, the Windows Explorer is a program containing the special 'Folders' explorer bar by default. Favorite Folders is also implemented as an explorer bar to use Windows Explorer's built-in functionalities such as copy and delete. Favorite Folders not only controls the explorer but also catches events from the explorer to synchronize itself with the browser pane. Navigating among folders in the Favorite Folders changes the browser pane and navigating among folders in the browser pane changes the selection in the Favorite Folders window. Therefore, it is quite similar to Windows Explorer. Favorite Folders is launched as shown in Figure 4.

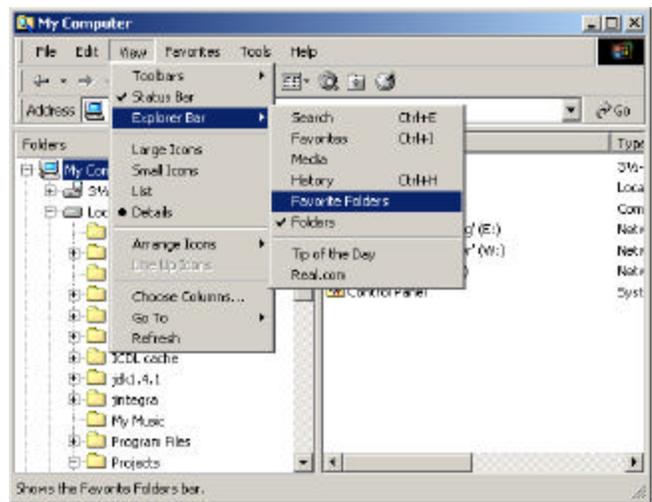


Figure 4. Selecting Favorite Folders in the Windows Explorer

**Index Database**

To decide which directories are shown, Favorite Folders uses a flag associated with each directory. Favorite Folders stores that flag in a relational database and indexes directories by name for fast access and search. Since the file system structure is hierarchical, parent-child relationships are stored, too. Table 1 shows the schema of the directory table.

| Field       | Description             |
|-------------|-------------------------|
| Id          | Directory id            |
| Name        | Directory name          |
| Parent      | Parent directory id     |
| Lastaccess  | Last access time        |
| Favorite    | Favorite flag           |
| Ancestorfav | Ancestors favorite flag |

Table 1. Directory Table Schema

The favorite field represents whether the directory itself is checked or not. As mentioned before, to be a favorite folder, a directory and all of its ancestors should be checked. Since it is slow to check whether a directory is a favorite folder or not by examining its ancestors' favorite fields recursively, we flattened the database with an ancestorfav field, representing whether all of its ancestors are checked or not.

When users check/uncheck a directory, Favorite Folders recursively sets the ancestorfav field of all of its descendents to true/false. Then, it can be determined if a directory is "favorite" by checking both favorite and ancestorfav fields are true, which makes the 'within favorites search' described in the following section fast.

This hierarchy flattening introduces one last complexity. Changing a single check mark can have a big impact on the database as that folder's entire subtree has to be updated. This process can be quite time consuming, so we implemented a separate thread to update the database. This means that when a check box is changed, while the interface reflects the change immediately, it can take several seconds for the database to get updated and if a search is done during this time, the recent changes are not reflected in the search results. This could be fixed by delaying searches until database updates have been completed. A related issue is that we use Microsoft Access to store the database, and Access doesn't support multi-threaded access. We therefore had to synchronize all access to the database – but our implementation does not synchronize across processes which means that only one instance of Favorite Folders can be run at a time. This could be fixed in the future by using a threaded database or by implementing a database access server that synchronizes database access across processes.

We also implemented a small utility program to index the directories in the file system. The program enables users to choose the drives they want to index since users may not want to index some drives such as cd-rom and network drives, which may be temporary or may have many irrelevant directories. This implies that there could be directories which are not indexed yet. Therefore, there are three states for a directory: (1) indexed favorite folder, (2) indexed non-favorite folder, and (3) non-indexed folder. These non-indexed directories can be indexed later when the user browses them or by rerunning the utility program. By default, all indexed directories are favorite folders at first.

If users browse the indexed directories, Favorite Folders retrieves child directories from the database. If users browse the non-indexed directories, Favorite Folders retrieves child directories from the file system and then indexes them into the database so that the next time when the user selects the same directory, necessary information can be retrieved from the database.

### Directory Search

Since every favorite folder is indexed by name in the database, Favorite Folders provides fast search.

The 'within favorites search' is the search restricted to the favorite folders. Users can also search for indexed but non-favorite folders in the file system with 'everywhere search'.

There are a few trade-offs between Favorite Folders search and Windows search. The search results in Favorite Folders highlight the results and shows them in context.. Then, after the search, it automatically selects the first search result directory so that its contents show in the browser pane. It is more useful when the search has only one result. On the other hand, Windows search can search for not only

directories but also files. It also provides advanced features such as searching for files containing specific text or modified recently. Figure 5 shows both search results with "Explorer" keyword done by Favorite Folders search and Windows search.

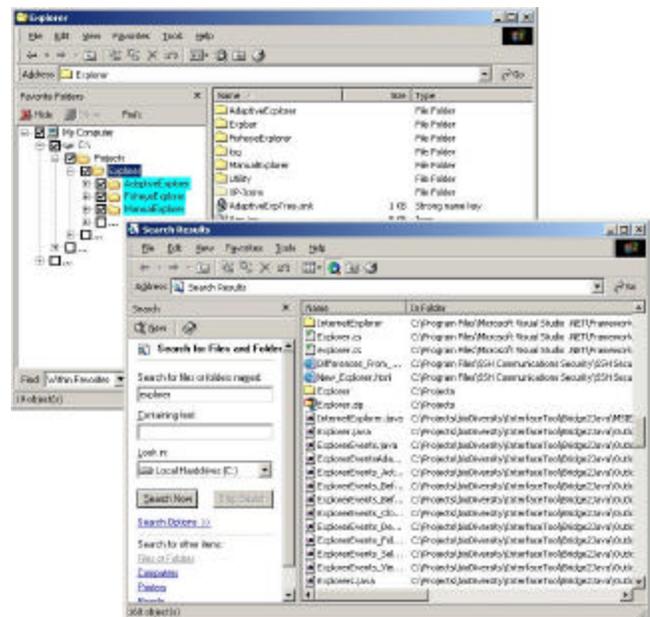


Figure 5. Favorite Folders Search vs. Windows Search

Auto completion can make the search faster. Favorite Folders stores the words users searched for before and provides the first match in the database whenever users type a character in the input box. For example, when users first try to find 'Explorer', they have to type the whole string. However, next time when users want to find it again after they type 'E', the program fills the input box with the keyword 'Explorer'.

Favorite Folders search is especially useful when users already know the location of a directory. Users can access a directory simply by searching for it instead of opening the directories in the path from the root. In other words, Favorite Folders search can serve as a shortcut.

Favorite Folders search also makes it easy to set up the favorite folders. When users perform a search, the search results may include irrelevant directories if their names contain the keyword. Users can exclude them from favorite folders by unchecking them. In addition, if users want to change a non-favorite folder to a favorite folder, they can find it by 'everywhere search'.

### File System Watching

It is necessary to update Favorite Folders properly when users or other programs change the file system structure. For example, directories can be deleted from the command

prompt, new directories can be made when users extract a zip file to a folder, or users can change the directory name in Windows Explorer's browser pane. To catch up with all these changes we implemented a file system watcher. The watcher is launched when the user starts Windows and updates the database when the file system changes.

### **Constraints**

For various reasons, Favorite Folders has some technical limitations.

Because of limitations in the Windows Explorer API, we can not launch Windows Explorer with Favorite Folders opened by default, or add a toolbar button to Windows Explorer's toolbar.

In addition, Windows Explorer intercepts some keyboard messages including backspace. So we disabled renaming since it is so inconvenient to rename directories without the backspace key. Instead, users have to use the browser pane in Windows Explorer.

As described in the previous section, only one instance of Favorite Folders can be run at a time. Finally, Favorite Folders is not fully compatible with samba-mounted Unix drives. First, the file system watcher cannot catch changes in the Unix file system because its underlying mechanism is provided by Microsoft Windows. Second, because directory names are not case-sensitive in Windows, Favorite Folders cannot distinguish Unix folders that are the same except for capitalization.

### **EVALUATION**

While the two authors of this paper use Favorite Folders themselves, we attempted to further validate the concept of Favorite Folders by conducting a preliminary field study. We chose not to run a controlled laboratory study because for Favorite Folders to be effective, users have to care enough about the directory structure to be willing to put in time to customize it. Furthermore, one of the things we wanted to investigate was whether users would in fact be willing to spend time to customize their directory structure in the first place.

We did not expect that the results of this study would provide a definitive understanding of whether Favorite Folders is better than Windows Explorer. Rather, we hoped to get an idea of how users think about the Favorite Folders concept and how they manage and access their file systems in order to plan for future directions.

We recruited 5 volunteers who: 1) primarily use a single Windows PC; and 2) use Windows Explorer regularly. Three of the subjects were computer science graduate students, one of the subjects was a biologist, and one of the subjects was a computer science research scientist. We helped them set up Favorite Folders on their machines and

gave them brief instructions. They were asked to use Favorite Folders in their ordinary setting for one week, from Monday to Friday. We collected participants' usage information, which directories they selected, expanded, searched and so on. We also asked them to fill out a simple questionnaire at the end of the week.

### **Logged Information**

Our main interests were how much time users saved by hiding irrelevant directories when they select a directory, and how much time they were willing to spend customizing Favorite Folders. To compute the expected savings of directory selection time based on the logarithmic model, which will be described later, we recorded the original number of candidate directories in Windows Explorer and the number of candidate directories in Favorite Folders.

However, the model doesn't include the scroll time, which is quite important. To compare the number of scrolls, we also recorded the height of the tree widget, the original location of the target item in Windows Explorer and the location of the target item in Favorite Folders. When a user opens a node in the tree widget, if all children do not fit in the tree widget, the tree widget automatically moves the node to the top to show as many children as possible. Therefore, if we know the target location and the height of the tree widget, then we know whether users need to scroll or not.

### **Results**

After using Favorite Folders for one week, subjects were asked to rate their satisfaction. While the results were not analyzed statistically because the study contained a small number of subjects and one week was not long enough, we learned some important facts and saw some interesting trends.

Mean ratings for Favorite Folders are shown in Table 2. Users indicated that Favorite Folders is easy to learn how to use. Two users indicated that Favorite Folders search is more useful than Windows search but others indicated the opposite. Their preference depends on whether they mainly search for files or directories. Users also indicated that Favorite Folders is easy to use and it is easy to set up the favorite folders by checking/unchecking the check boxes. However, two users stated that they often clicked on the check box by mistake, which caused a folder to be hidden and hence wasted their time repairing the mistake. For the same reason, one user stated that he did not feel in control of this system.

One important thing we found was that some users already had found their own ways to organize the directory structure to support hiding of infrequently accessed folders. For example, one user usually makes directories named "old" and hides segregated directories in them, which reduced the need for Favorite Folders. However, we can see that those folders serve as the ellipsis nodes. Therefore, if we put it in

another way, this is evidence of the necessity of the ellipsis nodes. The other user keeps her most used files in folders with direct shortcuts on her desktop. As the desktop directory is physically nested within the Documents and Settings folder it actually gets harder to get to with Favorite Folders because it is so nested. For these reasons, they did not think that Favorite Folders saved them much directory selection time.

|                                                                               |     |
|-------------------------------------------------------------------------------|-----|
| Favorite Folders was easy to learn how to use                                 | 5.8 |
| Favorite Folders search is more useful than Windows search                    | 4   |
| Favorite Folders was easy to use                                              | 5.2 |
| It was easy to set up favorite folders by checking/unchecking the check boxes | 5.8 |
| It was worth my time to set up favorite folders                               | 5.2 |
| I felt in control of this system when I was using it                          | 5   |
| It seemed like Favorite Folders saved directory selection time                | 5.6 |

Table 2. Average satisfaction ratings. (1=Disagree, 7=Agree)

It is important to note that the number of times the user had to scroll using Favorite Folders was less than that of Windows Explorer as we expected. Table 3 shows the number of scrolls for both. User 5 made Explorer very tall and so never had to scroll. We can see that the number of scrolls for two uses drastically decreased.

| User# | # of selections | # of scrolls in WE | # of scrolls in FF |
|-------|-----------------|--------------------|--------------------|
| 1     | 68              | 13 (19%)           | 0 (0%)             |
| 2     | 237             | 29 (12%)           | 0 (0%)             |
| 3     | 100             | 5 (5%)             | 2 (2%)             |
| 4     | 156             | 8 (5%)             | 3 (2%)             |
| 5     | 81              | 0 (0%)             | 0 (0%)             |

Table 3. Number of Scrolls

### Usability Problems

While we expected the users would use Favorite Folders regularly, several things discouraged users from using Favorite Folders consistently.

Users often access directories and files from open and save dialogs from applications without using Explorer directly. In addition, some users search for files rather than directories and Favorite Folders doesn't support searching for files.

Users frequently forgot to switch to Favorite Folders when they launched Windows Explorer. Unfortunately, there was no way to force Windows Explorer to start with Favorite Folders open, or add a button to Windows Explorer's toolbar.

Favorite Folders is a little slower than Windows Explorer because it has to query the database to get necessary information. One user preferred Windows Explorer because Windows Explorer is faster, he has been using it for a long time, and his folders are well organized, so favorite folders did not offer him much of a benefit.

As we mentioned earlier, Favorite Folders did not provide all the functionality of Windows Explorer. However, most of these technical problems are not indicative of flaws in the concept of Favorite Folders. In fact, one user explicitly stated that he definitely would use Favorite Folders very much, if the idea could be merged into Windows Explorer more seamlessly.

### MODEL

When we introduce a new interface, it is important to predict the benefits of the new interface because users may be reluctant to switch to a new system. Here we describe a model to predict the amount of time saved when users use Favorite Folders. Since we haven't conducted a controlled experiment yet, actual values for constants used to describe the model are not presented.

We can treat the file system hierarchy as an ordered hierarchical menu although the size is usually much bigger. Selecting a folder is the same as users choosing a menu item when they already know the name of the item.

Several models have been developed to predict the amount of time necessary to select an item from a menu [6, 7, 10, 11]. We can classify them into two categories: linear or logarithmic. Landauer and Nachbar's logarithmic model is based on the Hick-Hyman law for choice reaction time and on Fitts' law for movement time [6]. We also describe a model based on those two laws.

The Hick-Hyman law states that

$$t = c + k \log b$$

where  $b$  is the number of response alternatives,  $c$  and  $k$  are constants, and  $t$  is the average response time [3, 4]. This law holds very generally for situations in which people are required to react to any one of many items. The main question with respect to the application of the Hick-Hyman law to folder choice is whether the response time for folder selection is determined by a choice among responses, or by the time for visual scan-and match processes [6]. Since the folder names are sorted in alphabetic order, it is not necessary to search them sequentially to find the target item. We apply the Hick-Hyman law to our model assuming that

the response time for folder selection is determined by a choice among subdirectories.

Fitts' law states that

$$t = c + k \log \frac{d}{w}$$

where  $d$  is the distance moved and  $w$  is the width of the target,  $c$  and  $k$  are constants, and  $t$  is the movement time. For folder selection, since the width of the target is the height of a directory node, it is essentially equal regardless of a directory. The average distance from the parent to the item is proportional to the number of alternatives (subdirectories). Thus, for folder selection, Fitts' law gives

$$t = c' + k' \log b$$

The selection time is the sum of the choice reaction time and the movement time. If we apply both the Hick-Hyman and Fitts' laws, it is a linear function of  $\log b$ .

We need the following values to predict the amount of time to be saved by switching to Favorite Folders from a Windows Explorer. For simplicity, we assume that the hierarchy is symmetric (i.e., a balanced tree), we do not count the ellipsis nodes as subdirectories, and we do not account for scrolling. This last simplification is one which results in a bias in favor of Windows Explorer since Favorite Folders reduced the amount of scrolling needed. So, any benefit found for Favorite Folders would like be larger in actual use.

- $b$ : the number of subdirectories
- $p$ : the percentage of favorite folders
- $l$ : the path length of a directory (e.g. directory  $d = d_1 d_2 \dots d_l$ )
- $f$ : the percentage of favorite folders selections

For Windows Explorer, we always choose one out of  $b$  subdirectories. The average amount of time necessary to choose one directory with Windows Explorer is as follows:

$$t(WE) = c + k \log b$$

where  $c$  and  $k$  are constants.

For Favorite Folders, we have two cases: (1) the target directory is a favorite folder and (2) the target directory is not a favorite folder. If the target directory is a favorite folder, it is already shown in the list and the number of alternatives is  $b * p$ . The average amount of time necessary to choose a favorite folder is as follows:

$$t_{ff}(FF) = c + k \log bp$$

If the target directory is not a favorite folder, it is hidden in

the ellipsis node. Therefore, users first have to figure out that the directory they want is hidden, which takes  $c + k \log bp$ . Then, users have to expand the ellipsis node. Last, they have to select the target item, which takes  $c + k \log b$ . The average amount of time necessary to choose a non-favorite folder is as follows:

$$t_{\wedge ff}(FF) = 2c + k \log bp + k \log b$$

Then, the amount of time necessary to open a directory with Favorite Folders is as follows:

$$\begin{aligned} t(FF) &= f(c + k \log bp) + (1 - f)(2c + k \log bp + k \log b) \\ &= (c + k \log bp) + (1 - f)(c + k \log b) \end{aligned}$$

where  $c$  and  $k$  are same constants above.

To open a directory whose path length is  $l$ , we have to open  $l$  directories included in the path. Hence, the expected savings is

$$Expected\ savings = l * (f(c + k \log b) - (c + k \log bp))$$

Let us compute the expected savings for one user with real data gathered in the field study. The percentage of favorite folders ( $p$ ) was 0.414723, and the percentage of favorite folders selection ( $f$ ) was 0.911765  $\sim$  0.91.

First, we examine the time needed to open a directory with Windows Explorer. The average number of subdirectories ( $b$ ) was 20.17647. Whether we select a directory being a favorite folder or not, the necessary time to select it is  $c + k \log 20.17647 \sim c + 4.33k$ .

Then we examine the time needed to open a directory with Favorite Folders. The average number of subdirectories ( $b$ ) was 8.367647. If we choose a favorite folder, the time needed is  $c + k \log 8.367647 \sim c + 3.06k$ . If we choose a non-favorite folder, the time needed is  $c + k \log 8.367647 + c + k \log 20.17647 \sim 2c + 7.39k$ . The average time needed with Favorite Folders is  $0.91 * (c + 3.06k) + 0.09 * (2c + 7.39k) = 1.09c + 3.45k$ .

The average expected time to be saved or lost to open each directory in the path with the Favorite Folders is  $.88k - 0.09c$ . Since the average path length ( $l$ ) was 2.678571  $\sim$  2.68, the total expected benefits is  $2.36k - 0.24c$ . Table 4 shows the expected savings per selection in percentage depending on the constants  $c$  and  $k$ , based on empirical results of previous studies [4, 11].

|          | $c = .1$ | $c = .2$ |
|----------|----------|----------|
| $k = .2$ | 17%      | 15%      |
| $k = .4$ | 19%      | 17%      |

Table 4. Expected Savings

We see that the total average time saved when accessing the favorite folders is reduced when the non-favorite folders are accessed. However, if we only look at the time saved when accessing the favorite folders, we can save 1.27k amount of time for each access. It means that users can perform frequent things faster.

### Expected Savings

To compute more reasonable expected savings, we take into account scroll time and how much time users spend to customize Favorite Folders. We estimate times to scroll and click on a check box based on the Keystroke-Level Model [1]. One scroll consists of pointing to a target, mental preparation, button press, and drawing line. This operation takes 3.79 seconds. Single clicking on a check box consists of pointing to a target, mental preparation, and button press. It takes 2.73 seconds.

Selecting a directory in the tree widget is more difficult than selecting a menu item because users have to click either on the small plus sign or on the text. We use the slope value  $k = .3$ , which is a little larger than the value from the earlier work by Sears and Shneiderman [11]. We chose the conservative value  $c = .2$  based on the work by Hyman [4]. The number of subdirectories ( $b$ ) was 20.8, the percentage of favorite folders ( $p$ ) was 38%, and the percentage of favorite folders selections ( $f$ ) was 92.5%. On average, users would spend 234.2 seconds with Windows Explorer, to select 128.4 directories. They spent 132.8 seconds to select 118.8 favorite folders and spent 28.9 seconds to select 9.6 non-favorite folders. In many cases, we expect actual time spent could be longer. The average expected savings considering scroll time is 72.5 seconds.

Users clicked the check boxes 97.4 times to set up the favorite folders, which took 265.9 seconds. Assuming users reach a stable state after one week and click 10% as many check boxes in following weeks, they could save 3477.51 seconds in one year. The savings would be even larger if the software was used more often.

### FUTURE WORK

To verify the model we described, a controlled experiment needs to be conducted. Expected savings depends on the constants  $c$  and  $k$ . We can get the constants used to describe the model through the experimental results.

### Fix Usability Problems

Participants showed positive feedbacks about the concept of Favorite Folders. However, they preferred Windows Explorer because Favorite Folders has several usability problems. Some of these are impossible to fix, but most could be addressed. For example, it would make Favorite Folders more usable if we enable users to search for files in addition to folders and show the containing directories in the Favorite Folders view.

### Multiple Views Based on Roles

The main idea of Favorite Folders is that users are usually interested in the same small portion of the whole file system. We also expect that the directories users want to access will vary according to what they are doing. For example, students taking two classes may want to access different directories when they are doing projects for each class. It would be useful to provide different views of the file system based on the user's roles.

### CONCLUSION

Between our own experience using Favorite Folders and the small amount of data we gathered from our user study, we maintain our belief that the idea of letting users manually specify a single bit whether a folder is a "favorite" or not is a good one. At the same time, it is also clear that this approach is not for everyone. To summarize, we think the following elements are necessary to make the favorite folders concept work in practice:

- Interface must be integrated into every place that users access files (explorer, dialog boxes, etc.)
- Interface must be extremely fast to use and require no more than one click to specify whether a folder is a favorite, and no more than one click to access folders that have been identified as non-favorites.
- Interface must be very responsive, and no slower than the alternative interfaces.
- It must be possible to turn off the favorite interface for users that do not want it.

While we have focused on the concept of letting the user specify a favorite bit in the context of file systems, it is clear that the same approach could be generalized to menus and other places where there are many items to be selected from and some are more commonly selected than others. The application to menu selection is particularly interesting because it offers a direct replacement for the adaptive solution currently implemented by Microsoft Office with potential benefits and similar functionality.

### ACKNOWLEDGMENTS

We would like to thank our fellow members of the HCIL. We are especially grateful to volunteer participants for their time and effort.

### REFERENCES

1. Card, S. K., Moran, T. P., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 1980, pp. 396-410
2. Debevc, M., Meyer, B., and Svecko, R. An adaptive short list for documents on the World Wide Web. *Proceedings of the 1997 International Conference on Intelligent User Interface*, 1997, pp. 209-211.

3. Hick, W. E. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4, 1952, pp. 11-26
4. Hyman, R. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, 45, 1953, pp. 423-432
5. Kaasten, S. and Greenberg, S. Integrating Back, History and Bookmarks in Web Browsers. In *Extended Abstracts of the ACM Conference of Human Factors in Computing Systems (CHI'01)*, ACM Press, 2000.
6. Landauer, T. K., and Nachbar, D. W. Selection From Alphabetic and Numeric Menu Trees Using A Touch Screen: Breadth, Depth, and the Width. In *Proceedings of Human Factors in Computing Systems*, ACM, 1985, pp. 73-78
7. Lee, E. and MacGregor, J. Minimizing user search time in menu retrieval systems. *Human Factors*, 27, 1985, pp. 157-162
8. McGrenere, J., Baecker, R., Booth, K. S. An Evaluation of a Multiple Interface Design Solution for Bloated Software. In *Proceedings of Human Factors in Computing Systems (CHI'02)*, 2002, pp. 163-170
9. Miah, T., Karageorgou, M., Knott, R. P. Adaptive Toolbars: An Architectural Overview. *3rd ERCIM Workshop on "User Interfaces for All"*, 1997.
10. Norman, K. (1991). *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*. Ablex Publishing Corporation
11. Sears, A. and Shneiderman, B. Split Menus: Effectively Using Selection Frequency to Organize Menus. *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 1, 1994, pp. 27-51.
12. Shneiderman, B., Maes, P., and Miller, J. Intelligent Software Agents vs. User-Controlled Direct Manipulation: A Debate, CHI 97 Electronic Publications: Panels, 1997, (<http://www.acm.org/sigchi/chi97/proceedings/panel/jrm.htm>)
13. Tauscher, L. and Greenberg, S. (1997). How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *Int. J. Human-Computer Studies*, 47, pp. 97-137
14. Advanced Shell Techniques: Creating Custom Explorer Bars, Tool Bands, and Desk Bands, [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/Shell/programmersguide/shell\\_adv/bands.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/Shell/programmersguide/shell_adv/bands.asp)
15. Microsoft Word Help