A Visual Adaptive Interface to File Systems

Rosario De Chiara
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno
84081, Baronissi (Salerno), Italy
dechiara@dia.unisa.it

Ugo Erra
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno
84081, Baronissi (Salerno), Italy
ugoerr@dia.unisa.it

Vittorio Scarano
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno
84081, Baronissi (Salerno), Italy
vitsca@dia.unisa.it

ABSTRACT

In this paper we present our experience in building a visual file manager, VennFS2, that offers to users an adaptive interface toward access to files. Our file manager was originally designed to overcome some of limitations of hierarchical file systems, since it allows users to categorize files in such a way that files may belong multiple categories at once. Based on the past history of the files that were opened and modified by the user, VennFS2 graphically presents the user a small number of choices of the next file the user will modify. Some preliminary testing with interesting hints are also reported.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Interaction styles

General Terms

Venn diagram

Keywords

Set, Venn, User Interfaces, Adaptivity

1. INTRODUCTION

VennFS2 is proposed as a novel interface for the most common interaction all the users perform on a computer: access of the files stored on their computer.

Single-inheritance. As stated in [6] "The desktop and file&folder metaphor were created so that users could relate their computer-based systems to the paper-based systems they were used to". The heaviest heritage of this origin is the "single-inheritance" structure of filesystem: a file can be in one place at a time. A solution to this problem is provided in [5], multiple inheritance is given by the attributes associated to documents. Documents attributes capture the multiple different roles that a single document might play. In VennFS2 a simpler and more intuitive way of multi-inheritance has been implemented.

Adaptivity. As far as we know, there are no specific adaptive visual interface that provides the same capabilities of VennFS2. Nevertheless, some works on related topics show a particular interest in trying to improve and adapt the presentation and interaction with users. A work that is germane to our research, though not visual, is the Adaptive Command Line interface [2] where previous commands for a UNIX shell are used to provide indication to the next, most probable command to be typed. Some simple "adaptive" mechanism are also provided by Microsoft Windows. One of them is extremely silent and, still, very effective. Windows users that put to good use the ALT-TAB keys to switch among currently active tasks would have probably noted that the tasks are presented always in order of most recently accessed (i.e. that had most recently the focus) to least recently used. This consists in a consistent time saving in repeatedly switching between few relevant task among all the active ones.

Our proposal. The prototype that we present here is a tool that can help the user during daily activities. Our work follows the research line that started by developing VennFS [3] that allowed users to place documents and categories on a plane where files may belong to multiple categories at once, by using well-known and intuitive Venn diagrams to represent graphically each category. Our tool, VennFS2, is able to adaptively suggest the user what the next action (in term of access to files) can be. In fact, the files that are placed in the Venn diagrams are constantly monitored to recognize actions by the user. The actions performed up to that point are, then, used by VennFS2 to provide a certain number of suggestions. The suggestions are visual since they are based on the graphical representation with Venn diagrams, in such a way to present the file within its context, i.e. the categories and the files that are "close" to it in the plane. Our interface provides suggestions that are only few clicks (or keys typed) away of our normal activity. Therefore, we placed particular care in designing an interface that, though fully visual, asks only few and quick interactions by the user. The adaptive interface is not intrusive: the user is silently observed during the common activities and discreetly presented with the suggestions.

2. MAIN FEATURES OF OUR TOOL

VennFS (see [3]) can extend the traditional desktop metaphor based on hierarchical file systems by graphically organizing files on a plane in such a way that *(1)* each files can belong to multiple categories, *(2)* proximity on the plane of files/categories can be used by the user to relate information and *(3)* filtering by time is possible. The goal, here, is to solve the problem of documents that belong to several aspects of our work and that, therefore, should be placed into several folders at once. The technique we used is based on Venn diagrams. Categories are natural extension of folders and are represented by ellipses on the plane. The intersection of two or more ellipses is the logical place where to place files that belongs to different categories.

A remark is needed about symbolic links, the only technique, offered by file systems, to possibly address the problem of multiple categorization. They do not represent a suitable solution to our problem since they provide a solution for a single document and the approach is neither systematic nor visual (i.e. unintuitive at all). Moreover, links are difficult to trace back from the original document and therefore it is complex to change the folder which the document belongs to: all the links pointing at the document should be changed at once. The problem is well known in the Hypertext area: it is similar to the broken links that can be found on the Web, and for which solutions, like Open Hypermedia, were originally designed by the community.

Several advantages can be found on the representation of files on the plane, since we push the user to set data relationships as spatial relations by representing closeness of argument (between categories as well as between documents) by means of proximity in the plane. The user is suggested, implicitly, to draw related categories close to each other, since it allows partial overlap of them, while totally unrelated categories are intuitively placed far away. An interesting visual consequence of this usage of distance to relate documents is that filtering by arguments is implicitly obtained by zooming on the region and zooming in/out to get at the desired level. Our objective is that the user is given an instrument to easily draw the environment represented by the (unstructured) corpus of his/her own documents, place documents into (possibly multiple) categories and relate categories by proximity: the task of information retrieval for the user is made easier by building a cognitive map [10] of this environment. Since the environment is created by the user himself, the internalized analogy in the human mind of the physical layout (created by using our tool) becomes easier to grasp. In a sense, one may see VennFS 2 as a way to make a cognitive map of its documents explicit and browseable. In order to facilitate the navigation we provide the capability to place landmarks, since it well known that their identification helps in navigating [4] as well as learning and memorizing [7]. Landmarks do not correspond to files. A list of them is shown to the user and each one can be easily reached by double-clicking on landmark's label. A first attempt to help the user in recognizing recently accessed file was originally present in VennFS. In fact, an indication of how recent is the file is shown by using an easily interpretable metaphor: recent (i.e. recently accessed) files are "hot" (i.e. red) while old files are "cold" (i.e. blue) with intermediate colors to represent intermediate age. Then, it is important to allow filtering over the time (by using a slider) in such a way to show only the files whose last modification date is below a given date.

2.1 Prediction of the next file to access

The adaptive mechanism of VennFS 2 is based on Markov chains. The way of training a Markov chain and the way of use it are two critic choices, so we have also performed a preliminary testing sessions of the system that have been done on two users with different behavior.

Markov chain. As long as the user modifies files and documents kept in the sets diagram, VennFS 2 keeps track of modifications by probing last write time for every file. The variation of last modification time causes two effects: the color of the file label in the diagram goes to red (meaning a "hot" file) and a list of modified files (considering consecutive writes too) is kept up to date. This list is used to train a Markov chain. A Markov chain is a stochastic process with what is called the *Markov property* : the process consists of a sequence $X_1, X_2, X_3, \ldots$ of random variables taking values in a "state space", the value of X_n being "the state of the system at time n". The discrete-time Markov property says that the conditional distribution of the "future" : $X_{n+1}, X_{n+2}, X_{n+3}, \ldots$ given the "past", $X_1, ..., X_n$, depends on it only through X_n. Each particular Markov chain may be identified with its matrix of "transition probabilities", often called simply its transition matrix. The entry i, j in the transition matrix are given by $p_{ij} = P(X_{n+1} = j \mid X_n = i)$ that is the probability of moving to state j from the state i. Our Markov chain is labeled using file identifiers because we are interested in predicting the sequence of files modified by the user. We have tested two kinds of Markov chain.

- One label state memory: the next file that will be modified depends on the last file modified by the user. Every state in the chain is, therefore, labeled with one file identifier.
- Two labels state memory : the next file that will be modified depends on the last two files modified by the user. Every state in the chain is labeled with a couple of identifiers from the last two modified files.

Another critical decision to be taken how many time we query the Markov chain in trying to guess the next file the user will modify. Experimental results shows that these parameters are critical in terms of percentage of correct previsions. Preliminary experiments seem to suggest that presenting 3 choices is enough to make the tool useful.

3. PRELIMINARY EXPERIMENTS

3.1 Users characterization.

One user, the "focused", was working on an article in the days just before the deadline. The second user, the "discontinuous", was working on an article with no deadline and a software package. The focused user had a profile characterized by a sequence of a short patterns of modifications often repeated (e.g. `article.tex` $\longmapsto$ `article.log` $\longmapsto$ `article.dvi`) interleaved by sequence of modifications of a small number of files (e.g. figures for the article, charts, ...). The discontinuous user busy with a paper and an application, was characterized by a discontinuous pattern of

modifications with various length. This difference of behavior can be justified thinking about the way the two users work : the focused one was completing a paper so his modification session were short and frequent: corrections of the english or the fonts etc. . . ; the discontinuous user had long modification sessions interleaved by off line work like readying a paper. This difference is emphasized by the fact that the focused user was near a deadline, so he was in a hurry. It should be noticed that the experiments were run by users that were using their traditional desktop (MS Windows) and whose activity was simply monitored by a non invasive application that simply records the history of files modifications. This application is the same probing core implemented in VennFS 2.

Tests results. Here we show the results that driven us in fixing the parameters of the file prediction. It may be worth a remark that our tool is also adaptable, since we just provide the default settings but also let users free to specify them on his/her own. The first question we had to answer was how much memory had to be used in Markov model. We have used the focused user samples and we have queried our model verifying the correctness of the forecast. That is to say, at each step we built the model with the history of file access up to the point, queried (probabilistically) the model and, then, compared the prediction with the actual choice of the user. The results are that the chain with one state memory forecasts correctly the next file to open in about the 22% of cases while the two states memory chain forecasts correctly in the 26% of the cases. Having, consequently, decided that appealing to two states memory was worth the (moderate) increase in the computation needed for each suggestion, the second question is about the number of suggestions to present to user. Of course, the number of suggestion that are simultaneously presented to the user is a crucial parameter: it can (of course) increase the success rate but can, as well, induce disorientation if too many choices are presented. In Table 1 we show the performances comparing different number of suggestions proposed by the system. For every group of suggestions the test verifies if the next file is correctly predicted. In Figure 1 we show how the performance of our model, expressed in percentage of correct predictions, increase linearly with the number of suggestions obtained by querying the Markov chain.

Figure 1: Forecasting performance on increasing number of suggestions.

# Launches	Focused user		Not focused user	
	Average	Std. Dev.	Average	Std. Dev.
1	26.76%	4.23	15.58%	0.31
2	38.81%	6.94	16.67%	0.30
3	49.76%	8.40	17.18%	0.99
4	60.07%	5.64	17.05%	0.45
5	65.93%	5.52	17.82%	0.73

Table 1: Prediction correctness on varying number of launches. For both typologies of users.

The two data set we used (for the focused and discontinuous user) seems to show that the tool can really be effective for users that are continuously using files, with a tight and determined schedule while it does not seem useful for erratic usage of the computer. We also tested how quickly the model developed was able to provide a reasonable success rate in the suggestions. Results were good for the focused user: after roughly 100 file modifications, the model had already reached the success rate (1 and 3 launches) that is reached at the end (as reported in Table 1). Results were not as good (which could be expected) about the discontinuous user given the erratic behavior in accessing files. Considering the limited amount of data used in these experiments, it is rather premature to draw definitive conclusions and we believe that larger data sets (as well as a user study) are needed.

Figure 2: Forecasting performance on increasing number of suggestions.

4. THE GRAPHIC INTERFACE

As VennFS 2 starts, two windows appear, the main window shows an empty "desktop" on the right, a left pane with three group boxes and few simple menus, and a secondary window 'named 'VennFS Suggestions" (see Figure 2). In the main window the tabbed pane on the left group the boxes "Overview" that allow navigation and placing/jumping to the landmarks, "Filter&Zoom" that offers filters by time and shade by distance of the filename from viewpoint and "Details on Set" that permit to explore and modify information about categories and documents. These three tasks obey to Shneiderman task by data type taxonomy[9], as well as fully reflecting his Visual Information Seeking Mantra "Overview first, zoom and filter, then details-on-demand". The window "VennFS Suggestions" offers a 3D visual of the desktop emphasizing the probable next three documents to be modified. The menus offer setting preferences ("VennFS") and exporting to a hierarchical file system ("HFS"). A complete description of the interface can be found in [3]. Here we describe the file suggestion windows that is the adaptive part interface.

File suggestions. The secondary window shows information about the next three documents that are probably going to be modified by the user. The proposed size of the window (i.e. 320×200 pixel) has been carefully chosen in order to allow effective visualization while, at the same time, allow to keep larger windows (with the main task) open. The VennFS 2 desktop's appears here in three dimensional and whereas a file is suggested, a "razor-shell" indicates it, with the height proportional to the probability of the transition in the Markov chain between the current file and the suggested one. This window using its focus has two "modus operandi": (*1*) when the window has not the focus then the camera offers a global view about the three suggestions rotating around them, (*2*) when the window has the focus then the user can, by pressing the TAB key, switch the view point on each suggestion, pressing the Enter key to open the file when a reasonable choice is made. We have chosen a 3d interface to visualize file suggestions because we want to provide the information in the focus+context flavor: VennFS 2 shows the 3 most probable files the user is interested to plus the context in which these files are flying around the razor shells. Since the first task is supposedly performed during

the main activity, it is realized when there is no focus on the window and, by using a slowly rotating point-of-view in such a way to provide a complete overview. Then, when the user decides to use (possibly) the suggestions, the files are shown with few clicks[1]. In fact, by using the Windows ALT-TAB shortcut, the most probable suggestion can be opened with 2 clicks (ALT-TAB and Enter) if the Suggestions window is the next task in the list of current tasks that is proposed by Windows and few more if it is not. Other proposals require 1/2 clicks more. Access by using the mouse is efficient as well. The motivation to our choice is that our adaptive interface must be a small window (possibly shown, as a consequence, with the window of the main task) that is only "few clicks" away from our main task and can be accessed with the smallest effort and intrusion in everyday activities. Some authors [11] have already noticed (in previous experiences with a smart editor) that among users, regardless of efficiency, only few of them used the provided extended capabilities with frequency, because of the cost of a slightly complex interface. The same attitude has been noticed in adaptable interfaces, such as UNIX customization [8], which suggests that inconspicuousness must be a paramount characteristics of any tool.

5. CONCLUSIONS

Our research proposes a tool that is meant to extends and facilitates management of files by offering an alternative classification of documents as well as a dynamic, adaptive support to interact with them. Being an interface to a hierarchical file system our tool allows the user to place in VennFS2 only the really meaningful documents, i.e. those that contain information and not only raw data, of no interest for the user. In fact, often several auxiliary files (like log files, backup copies, etc.) are automatically created with the "master" document and placed into the same folder: their explicit presence can only distract the user from his/her task since their role is determined/needed by the software and not by the user. The adaptivity mechanism is based on the files that are placed in VennFS2 and, therefore, only on the documents that were explicitly estimate as relevant by the user.

The work on VennFS2 is actually in progress. Several steps are to be taken about the visual presentation of suggestions. Among the characteristics currently under development, we are working to further make suggestions inconspicuous e.g., by increasing interaction with the status bar.

Another important characteristic is how we can extrapolate more abstract information about behavior: given that the user seems to be frequently opening a sequence of files `xxx.tex` $\longmapsto$ `xxx.dvi` develop a model that is able to infer that when the user opens `newpaper.tex` (never opened before) the tool presents as an option to open file `newpaper.dvi` right after. In this context, techniques as Relational Markov Models [1] can be extremely helpful and their application is under study.

6. REFERENCES

[1] C. Anderson, P. Domingos, and D. S. Weld. Relational markov models. In *Proceedings of KDD-02*, 2002.

[2] B. Davison and H. Hirsh. Toward an adaptive command line interface. In *Proceedings of the Seventh International Conference on HumanComputer Interaction*, San Francisco, CA, USA, August 1997. Elsevier Science Publishers.

[3] R. De Chiara, U. Erra, and V. Scarano. "VennFS: a venn diagram file manager". In *Proc. of the Seventh International Conference on Information Visualization, IV 2003, 16-18 July 2003, London, UK*. IEEE Computer Society, 2003.

[4] A. Dillon, J. Richardson, and C. McKnight. Navigation in hypertext: a critical review of the concept. In *Diaper, D., Gilmore, D., Cockton, G., and Shackel, B. (Eds) INTERACT '90*. North Holland, Amsterdam, 1990.

[5] P. Dourish, W. K. Edwards, A. LaMarca, and M. Salisbury. Presto: an experimental architecture for fluid interactive document spaces. *ACM Trans. Comput.-Hum. Interact.*, 6(2):133–161, 1999.

[6] S. Fertig, E. Freeman, and D. Gelernter. Finding and reminding reconsidered. *SIGCHI Bull.*, 28(1):66–69, 1996.

[7] C. Johns and E. Blake. Cognitive maps in virtual environments: Facilitation of learning through the use of innate spatial abilities. In *Proc. of 1st International Conf. on Computer Graphics, Virtual Reality and Visualisation in Africa, Cape Town, South Africa. 2001*, 2001.

[8] W. Mackay. Triggers and barriers to customizing software. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 153–160. ACM Press, 1991.

[9] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of 1996 IEEE Symp. on Visual Languages, pages 336-343, Boulder, CO, USA*, Sept 1996.

[10] E. C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55:189–208, 1948.

[11] D. S. Weld, C. Anderson, P. Domingos, O. Etzioni, K. Gajos, T. Lau, and S. Wolfman. Automatically personalizing user interfaces. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI03)*, Acapulco, Mexico, August 2003.

[1] By the term click we include also key pressed.