Sifting through Hierarchical Information

Doug Schaffer
Saul Greenberg

Department of Computer Science,
University of Calgary
Calgary, AB, T2N 1N4 Canada
schaffer, saul@cpsc.ucalgary.c
+1-403-220-6087

ABSTRACT

Modern computer users must often sift and manage vast amounts of hierarchically structured information. However, conventional interface tools have not kept pace with the information explosion, leaving users with inadequate means to manage their data. This paper promotes ideas of information filtering and fisheye views of hierarchies through the use of dynamic queries. In particular, we present FLEXVIEW, a graphical system for visualizing file systems.

KEYWORDS: Fisheye views, information filtering, dynamic queries, visualization.

Schaffer, D. and Greenberg, S. (1993). Sifting Through Hierarchical Information. Report 1993-499-04, Dept. of Computer Science, University of Calgary, January.

INTRODUCTION

As storage mediums decrease in cost, the information people keep in their personal space is increasing at an explosive rate. 120 megabyte disks are standard on many personal computers, and gigabyte disks are not uncommon. Of course Parkinson's law applies, and even large discs are filled to capacity. Unfortunately information visualization and management methods, originally designed for modest data spaces, have not kept pace and do not extend adequately.

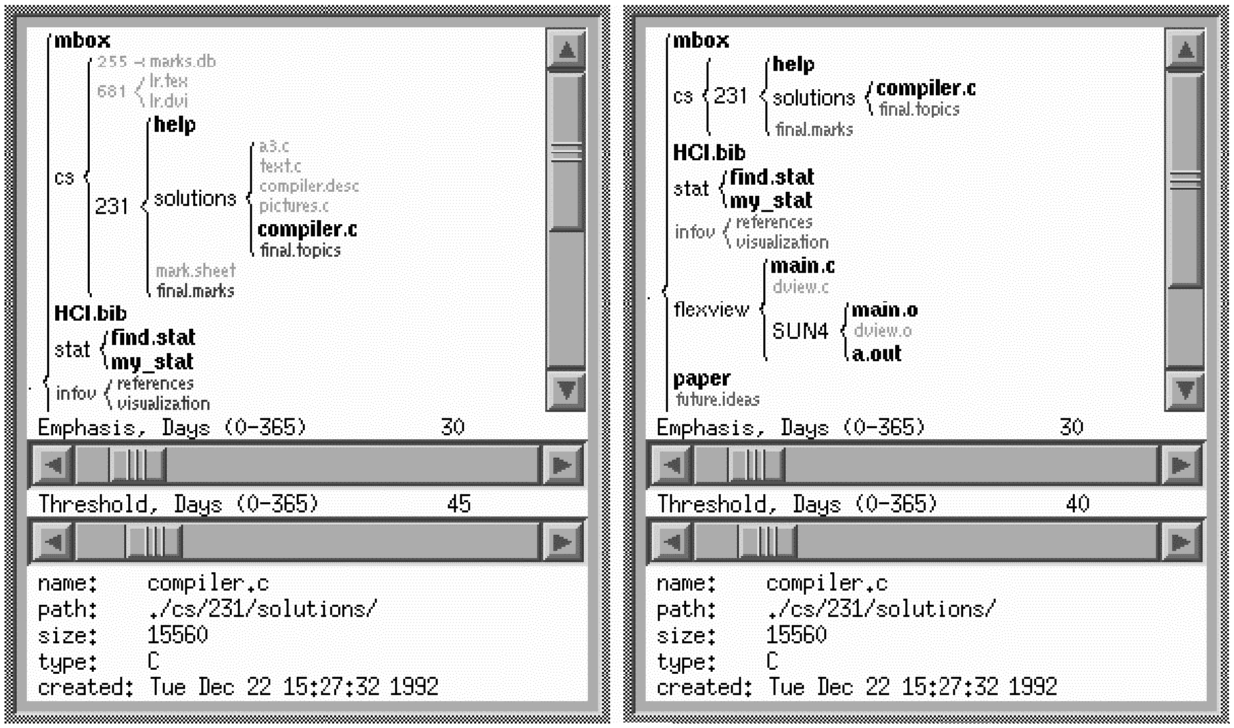
This paper is concerned with new techniques for viewing hierarchically structured databases that are quite common to computers, e.g. electronic documents and filing systems. It is not really a question of creating better keyword search or database query systems. Rather, people need a way to explore their information space. They require an overall feel for what is there and how things relate to each other. They should be able to give fairly fuzzy query specifications, and see the results as a range of candidates.

Four innovations help meet users' requirements mentioned above. With *filtering*, the hierarchy is screened by some criterion such that only elements passing the criterion are displayed. With *dynamic queries* [1], a user can interactively vary several filter attribute values; a restricted (and animated) view of the hierarchy is updated in real time. This technique is particularly valuable because the information updates emphasize the relations between attributes as well as the actual results. With *clustering* [5], a set of related points of information are grouped together and a representative symbol is displayed in their place. With *fisheye views* [2], focal points containing interesting information are emphasized on the screen at the expense of items of lesser importance. Recent systems based on some of these ideas include TREE-MAPS [3], CONE TREES [4] and VARIABLE ZOOM [6].

By combining these four innovations, users should be able to specify fuzzy searches, see the results and their relations in the context of the hierarchical space, and reduce the clutter of non-important information on the display.

DESCRIPTION OF FLEXVIEW

The ideas of fisheye views, dynamic queries, filtering, and clustering are combined and extended in our FLEXVIEW prototype. We use FLEXVIEW for viewing file hierarchies, where each file/directory contains a set of attribute values. File attributes could include: time of creation, modification, or last access; size; type; frequency and recency of access or modification; links to other files; and so on. However, the ideas behind FLEXVIEW can be generalized to any tree and attribute set.



```
Emphasis, Days (0-365)       30
Threshold, Days (0-365)      45
name:    compiler.c
path:    ./cs/231/solutions/
size:    15560
type:    C
created: Tue Dec 22 15:27:32 1992
```

```
Emphasis, Days (0-365)       30
Threshold, Days (0-365)      40
name:    compiler.c
path:    ./cs/231/solutions/
size:    15560
type:    C
created: Tue Dec 22 15:27:32 1992
```

Figure 1. FLEXVIEW file system display with Emphasis range of 0-30 days. The Threshold range is:

left view:	0-45 days,
right view:	0-40 days.

Clicking on an item causes information about that file to be displayed in the lower information window.

FLEXVIEW works as follows. As with other popular file-viewing systems, the hierarchy is displayed through a scrollable list, with each level indented in. As seen in the figure, each rightmost horizontal entry is a file, while directory names occur to the left of vertical brackets. Each bracket encloses its subdirectories and files. The difference is that users can filter their view by setting a *threshold range* for a file attribute, with items outside the threshold disappearing from view. A multi-foci fisheye view effect occurs through two methods. First, users can set an *emphasis range* (a sub-range of the threshold), where items within this range are made prominent on the screen through color, boldface, or font size. Second, files whose attribute value lies outside the emphasis range fade from black to gray on the display as they approach the threshold limit, disappearing when they exceed it. Dynamic queries occur when people use sliders to change the ranges; the view is immediately updated as the slider is moved.

At the time of writing, FLEXVIEW has been implemented for a single attribute, the file modification date. Through dynamic queries, users can quickly find all the files that they have recently used, with older work filtered off the display and newer work emphasized. If searching for something that occurred a longer time back, one simply increases the threshold until the right files come into view. For example, the figure shows two views of the same information system, with the one on the left including older files than on the right (although the emphasized files are held constant). Note the multiple focal points, as filtering and emphasis works on many parts of the hierarchy.

EXTENSIONS AND GENERALIZATIONS

We are extending FLEXVIEW and its ideas in several ways. First, it will allow users to query a database with multiple attributes. The decision on whether to filter or emphasize an item could be made on either the specific value of any of its single attributes, or as a weighted value of some or all of its attributes considered together. An issue is whether attribute values are continuous (such as date or size) or just values in a set (such as file type), for it is not clear how users pose dynamic queries on the later or how they can be factored into a weighting formula.

Second, while FLEXVIEW provides some coarse animation during dynamic queries, full animation would be a great improvement in seeing how items are inserted or deleted from the display (which happens when threshold values are passed) [4]. Current insertions and deletions are discrete, leading to a “jumpy” image.

Third, FLEXVIEW should be tied to the actual file system. The view should change as the user alters those files, and file manipulation facilities common to file browsers should be available (e.g. opening applications, moving things around, deleting items, and so on). Because FLEXVIEW is only a prototype, it just works on a static map of the file hierarchy, and does not allow files to be manipulated beyond some simple information retrieval.

Fourth, FLEXVIEW can use clustering to replace lists of files by a single directory name. For example, when a directory contains items at or near the threshold value, the directory name can be substituted for a lengthy file list. The user gains a cue that something interesting may be there without the expense of screen clutter.

Finally, the most important step yet to be taken is generalization. There is nothing in FLEXVIEW that restricts it to file systems, and all its ideas can be extended to a broad range of hierarchical structures. We are developing a generalizable widget that can attach dynamic queries, filtering, clustering, and fisheye views to any attribute set in any hierarchical data space. Developers can then attach callbacks to items to determine their behavior when a user selects an item.

CONCLUSION

You have to use FLEXVIEW on your own filing system to appreciate its capabilities. Even with this early prototype, people comment on the flexibility they now have in viewing their files. We believe it fairly straightforward to extend existing file (and hierarchy) browsers to incorporate the ideas in FLEXVIEW.

ACKNOWLEDGMENTS

This work was supported the National Science and Engineering Research Council of Canada.

REFERENCES

1. Ahlberg, C., Williamson, C. & Shneiderman, B. (1991) "Dynamic queries for information exploration: An implementation and evaluation." CAR-TR-284, Dept of Computer Sciences, U. of Maryland, USA.

2. Furnas, G. W. (1986) "Generalized fisheye views." *Proc ACM Human Factors in Computing Systems*, 16-23.

3. Johnson, B. & Shneiderman, B. (1991) "Tree-Maps: A space-filling approach to the visualization of hierarchical information structures." *Proc IEEE Visualization*, 284-291.

4. Robertson, G. G., Mackinlay, J. D. & Card, S. K. (1991) "Cone trees: Animated 3D visualizations of hierarchical information." *Proc ACM Human Factors in Computing Systems*, 189-194.

5. Sarkar, M. & Brown, M. H. (1992) "Graphical fisheye views of graphs." *Proc ACM Human Factors in Computing Systems*, 83-91.

6. Schaffer, D., Zuo., Z., Bartrun, L., Dill, J., Dubs, S., Greenberg, S. & Roseman, M. (1992) "Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks" Report 92/491/29, Dept of Computer Science, U. of Calgary, Canada.