

Self-Organizing File Cabinet

Dawn Lawrie

Department of Computer Science

Dartmouth College

Technical Report: PCS-TR97-313

May 29, 1997

Senior Honors Thesis

Advisor: Daniela Rus

Abstract: This thesis presents a self-organized file cabinet. This file cabinet uses electronic information to augment the physical world. By using a scanner to transform documents into electronic files, the self-organized file cabinet can index the documents on visual and textual information. The self-organized file cabinet helps the user find the documents at a later date. The focus of this thesis is on the design and evaluation of the self-organized file cabinet. User studies show that this tool is natural to use.

Table of Contents

	Page
1 Introduction	1
2 Related Works	7
2.1 The Intelligent Room	7
2.2 InteractiveDesk	8
2.3 The Self-Organizing Desk	8
2.4 Scatter/Gather Browsing	9
2.5 The Online Star Algorithm	9
3 System Description	11
3.1 The File Cabinet Structure	12
3.2 The Searching Algorithms	17
3.3 Graphical User Interface	18
3.3.1 The Algorithm for Adding a Paper	20
3.3.2 The Algorithm for Removing a Paper	21
3.3.3 The Algorithm for Retrieving a Paper	21
3.3.4 The Algorithm for Moving a Paper	21
3.3.5 The Algorithm for Summarizing a Drawer	21
3.3.6 The Algorithm for Removing a Drawer	22
3.3.7 The Algorithm for Clustering a File Cabinet	22
3.3.8 The Algorithm for Finding Similar Papers	22
4 Experiments	24
4.1 Correctness and Completeness	24
4.2 Performance	24
4.3 Naturalness and Usefulness	25
4.3.1 Survey	26
4.3.2 Poster Symposium	27
4.3.3 User Test Studies	29
5 Conclusion	35
6 Acknowledgments	36
7 References	37

1 Introduction

Self-organization refers to the ability of the system to expose some structure of itself. Initially, self-organization was thought to be impossible. St. Thomas Aquinas who is arguably the most influential Christian thinker constructed logical proofs of the existence of God. In one proof he referred to God as the ultimate organizer or designer. Since everything had to be organized, there must be an organizer. In turn this organizer had to be organized, and so on, until one ran into the original organizer and this was God [HJT97a]. The idea of self-organization was born out of a study of life in greater depth. Life itself appeared to be self-organized. The chemicals needed to produce proteins had to organize themselves so that life could begin. There are other places where self-organization can be observed occurring rather naturally. For example in a classroom, the teacher asks the class to divide itself into groups of five and a few minutes later that is exactly what has happened. The class has self-organized.

Self-organization was first formally defined by Farley and Clark of Lincoln Laboratory in 1954 in their paper on the Transactions of the Institute of Radio Engineers, Professional Group on Information Theory. They defined it as "a system which changes its basic structure as a function of its experience and environment" [HJT97b]. This definition has been slightly modified because

an organism does not organize itself independent of its environment. Von Foerster persuasively argued that only organisms and their environments taken together organize themselves. Ashby redefined a self-organizing system to be not an organism that changes its structure as a function of its experience and environment but rather the system consisting of the organism and environment taken together [HJT97b].

Self-organization has been studied in many fields including biology (insect societies, human brains), chemistry (thermodynamics), computer science (decision algorithms, robotics systems), geology (tectonic movements), sociology (communication and migration) and economy (socio-spatial systems).

This thesis describes our experiences with self-organized information systems. For the self-organization of information, one expects the data items to arrogate according to unspecified qualities. Self-organization is the ability to compute the underlying structure that is revealed by the execution of an algorithm. Depending on the change patterns of the electronic data, different methods of self-organization can be applied to both static (unchanging) information systems and dynamic (changing) information systems. Dynamic databases are more challenging because the system has to be able to deal with adding and removing information from the database without recomputing the entire organization. Such a method that works on dynamic databases has been developed called the Online Star Algorithm[APR]. The Online Star Algorithm uses an enhanced version of the Smart Information Retrieval System [Sal91] to compute a document to document similarity graph. The vertices of the graph correspond to documents and the weighted edges show the similarity between the corresponding documents. The similarity graph uses a threshold so that the resulting graph can be covered by dense star-shaped subgraphs. Related documents are then displayed within clusters so that a user can easily examine similar documents. Documents may be part of more than one cluster because this algorithm examines subtopics as well as main topics. Accurate clustering of information is crucial to the development of self-organization of information within the mundane world.

A natural question that arises is whether self-organization can be applied to physical objects. We believe it can because sensors can be attached to physical objects to extract and monitor electronic information about the physical object. Self-organization remains an operation at the electronic level which is mapped onto the physical level. The example that we study is a smart filing cabinet augmented with a sensor (scanner) and a processor (database), but one can imagine this methodology for other mundane objects. For example, one can imagine an office that has several well-placed cameras. These cameras could track the movement of everything within the office. One might be placed to monitor the bookshelf, and another watching the desk. A computer would then analyze the input from the camera and track locations of objects. The implementation of a self-organized desk has already been attempted [RD97]. It turns out that three-dimensional objects are extremely hard to identify. The self-organized desk has restricted its objects to papers because they are easier to identify. However, it is also difficult to identify the exact location of a paper.

In the development of a self-organized file cabinet, the hard problems encountered in organizing a desktop are removed. The file cabinet only contains flat pieces of paper so the design of the object is trivial. It is the same for every object. Also, there is more control over the follow of information. Initially one adds a document. Sometime later one might want to take it out and read it. Finally the information is no longer important, so one throws the document away.

Self-organization of information is useful especially now when the proliferation of information has generated vast repositories of data both electronic and physical. How can one locate all the useful sources of information? By designing systems that augment physical reality with

electronic data, humans can improve their ability to keep track of large amounts of information. The creation of a smart physical world provides a mechanism for that world to keep track of its own contents, to index, and to organize its objects within electronic views.

The self-organized file cabinet is designed to be such a tool. Consider a work-related file cabinet which contains papers, letters, and other miscellaneous bits of information that are desirable to save. Personal filing systems are inadequate because people have trouble developing a consistent way of filing documents since it is hard to predict future demands for the information. Instead, people leave papers in piles on top of their desk which clutters their work space. The self-organized file cabinet takes on the responsibility of keeping track of files once they are put away. It will then be able to answer a wide range of queries so that a specific paper can be found rather than requiring users to remember the one exact piece of information that they used to file the document some months ago. The file cabinet can answer such queries as "where are my papers about women in science?", "what are the main topics in my file cabinet?", "which papers have I not touched in five years?", *etc.* The file cabinet might answer with a list of papers and graphically approximate where the papers are within the file cabinet. Or it might provide a diagram of clusters to represent the topics. Each cluster would be labeled with a description of its topic.

This paper describes a system that implements the self-organization metaphor on the documents in a file cabinet. The users create two electronic representations of their documents by scanning the document and saving it in a visual format as well as textual one. A paper is added to the filing cabinet by referencing these files and specifying the drawer in which the document will be placed. This enables the document to be indexed on its attributes

(words, color, location, time, *etc.*). The system interacts with users through a graphical user interface (GUI) to help them locate documents that have been filed. The GUI displays the paper within the file cabinet as well as enables the user to view the document electronically and provide textual information about its specific placement within the drawer and the time it was filed or last moved. The file cabinet supports such operations as changing dimensions of the file cabinet; adding, removing, retrieving, and moving a document; and summarizing and removing a drawer. The GUI allows the user to submit queries based on colors or text strings to locate documents. One is also able to find out how documents are related to each other as well as find all documents that are similar to a given one.

A normal file cabinet requires one to fine by one topic even if an article is about three different topics or by one author when five people co-wrote it. Months later one must remember the same topic or author that was important months ago. The self-organized file cabinet does not care what you remember, author, title, topic, or even a color that appeared on the page as long as you remember something.

Initially, you set up your file cabinet so that it looks like the one in your office. If you have two drawers under your desk and a column of five drawers in the corner, you make the file cabinet on the screen have two drawers in one columns and five in another. To use the file cabinet, you first scan in the page and create a tiff file of it to capture the picture of the page and an OCR version to have an ASCII file. The document is added to the file cabinet, and the file cabinet recommends a drawer based on the how related the document is to documents already in the file cabinet. You can decide to accept this recommendation or specify another drawer. Then you are expected to put the document in the front of that drawer.

When you want to find a document, the file cabinet answers queries such as "where is the paper with the red table in the lower right hand corner?" or "which papers are about intelligent agents and have red pictures on the left side?". Let us say that you found the intelligent agents paper you were looking for, and now you want to find all similar papers to reference in your paper. This is possible by clustering on a paper. Now that you have located all the papers you want, retrieve them from the file cabinet. It is also important to tell the computer the papers are no longer in the file cabinet, but you don't want to remove them because the same papers might also contain information about another topic. Instead, move the papers to a "desktop" drawer. This way the computer can at least tell you that the paper is not in the filing cabinet even if it cannot give you anymore specific information about the paper's whereabouts. Once you are done reading the paper, you ask computer to suggest a drawer or you can place it in the front of any drawer.

The self-organized file cabinet that we implemented has all the features described above except that it lacks the "desktop" drawer. We realized that such a drawer was imperative as we sat down with six professors, graduate students, and undergrads during user studies. They had many more suggestions for improvement, mostly revolving around more operations done using the mouse. Through a survey of about 80 Dartmouth students, we learned that 86 percent were interested in this tool. At the Women in Science Program poster symposium, we presented the idea to many professors outside the department of computer science. All left with the hope that someday they could use this filing cabinet.

In the rest of the paper, we will discuss related works, present the system that implements the file cabinet system, detail the user studies that we performed, and conclude with future extensions.

2 Related Works

Other groups are interested in the development of tools that augment physical reality with electronic data. MIT is developing an Intelligent Room. The Hitachi Research Laboratory is working on the InteractiveDesk. Here at Dartmouth a Self-organizing Desk is being developed.

Organization is another area of research that is involved in the self-organizing file cabinet. Xerox is working on the scatter/gather methodology and trying to design user interfaces that help people search through large amounts of information faster than the current paradigms allow. The Online Star Algorithm for Information Retrieval addresses the problem of computing accurate topic clusters for a collection of documents.

2.1 The Intelligent Room

The Intelligent Room has been design to help groups of people collaborate in a shared meeting space. The Intelligent Room offers many features which include: the computer's ability to track the room's occupants and understand human gestures; intelligent agents that "autonomously navigate National Information Infrastructure to retrieve information or to provide computational service to the room" [Tor95]; and agents that keep a repository of data and plans in a persistent database.

The room uses cameras to tract the occupants of the room. The computer selects two of the most interesting views of the image to track the proceedings in the room. The room watches for humans to point at objects in the room and if the human is pointing at a place where the computer registers a command, the command will be carried out. They are also implementing speech recognition commands to enhance a speaker's presentation.

2.2 **InteractiveDesk**

The designers of the desk believe that the desk is the center for intellectual human activities and that the object will retain its importance even with the increasing capabilities of computers[AM+94]. For this reason, they have developed a desk which allows the user to interactive with the computer as if it were a more traditional desk. The components of the desk include a computer with keyboard and mouse. It is important to have large displays to make it easy to view the working environment. They use a 26" CRT display as well as an A1-sized transparent tablet as the desktop display with a pen input facility.

The InteractiveDesk allows the user to add personal notes in writing to files. It also allows one to link real world objects to files on the computer. This plays on the principal that a person would rather retrieve a scrapbook instead of digging through the hierarchy of directories to find the documents of interest. The desk uses a camera to recognize these objects. The camera is also used to anticipate which work space the user to using. This determines which monitor is used to display the items.

2.3 **The Self-Organizing Desk**

The self-organizing desk [RD97] shares many features with the self-organizing file cabinet since the desk was the predecessor to the file cabinet. The desk uses a camera to monitor the traffic on a desktop. It takes pictures of papers and processes the contents of a paper so that a user can submit queries to the computer when trying to find a paper rather than sifting through papers on the desktop. It then keeps track of the location of the paper as it moves around on the desktop. The system relies on the fact that papers tend to be stacked on the desk so it allows one to move a stack of pages at a time without loosing track of a document.

One is then able to search for documents using keyword, color, layout, and location queries. The system answers these queries with the objects location telling what part of the desk it is on and if it is hidden from view, what is hiding it.

2.4 Scatter/Gather Browsing

Scatter/Gather provides a fast way to browse a large set of information [PS+96]. It has been designed to use with web searches. The general principal is that people would prefer to get an idea of what type of information is available before investing a lot of time reading specific documents. The application first displays clusters that are labeled with the topics discussed in the documents. A user selects (gathers) the clusters that seem to be most relevant and then reclusters (scatters) them to form more specific clusters. This process is repeated until the search has been narrowed so that the documents that remain are a reasonable number to examine individually. This helps prevent relevant documents from being overlooked or running into deadends without finding anything of interest.

2.5 The Online Star Algorithm

The Online Star Algorithm provides a more accurate way to cluster documents [APR]. It clusters either all the documents within the current database or a sub-set of the documents. The Online Star Algorithm uses an augmented Smart Information Retrieval System [Sal91] to index the documents. When clustering the documents, it creates a similarity graph where the documents are vertices and the weighted edges correspond to similarity. The algorithm uses a threshold to cover the graph with star-shaped subgraphs and create the clusters. Users can then examine each cluster individually. The clusters displayed on the screen vary in size

depending on the number of documents contained within. The distance between clusters represents how closely the clusters are related.

3 System Description

The self-organized file cabinet was designed so that it would be easy to use and worth the trade-off of initial time versus saved time later. Instead of figuring out where to store a document, one must be patient while the computer scans the document. But later, it becomes much easier to find documents that are in the self-organized file cabinet. Figure 1 (below) shows the relationship between the electronic database and the file cabinet in the physical world.

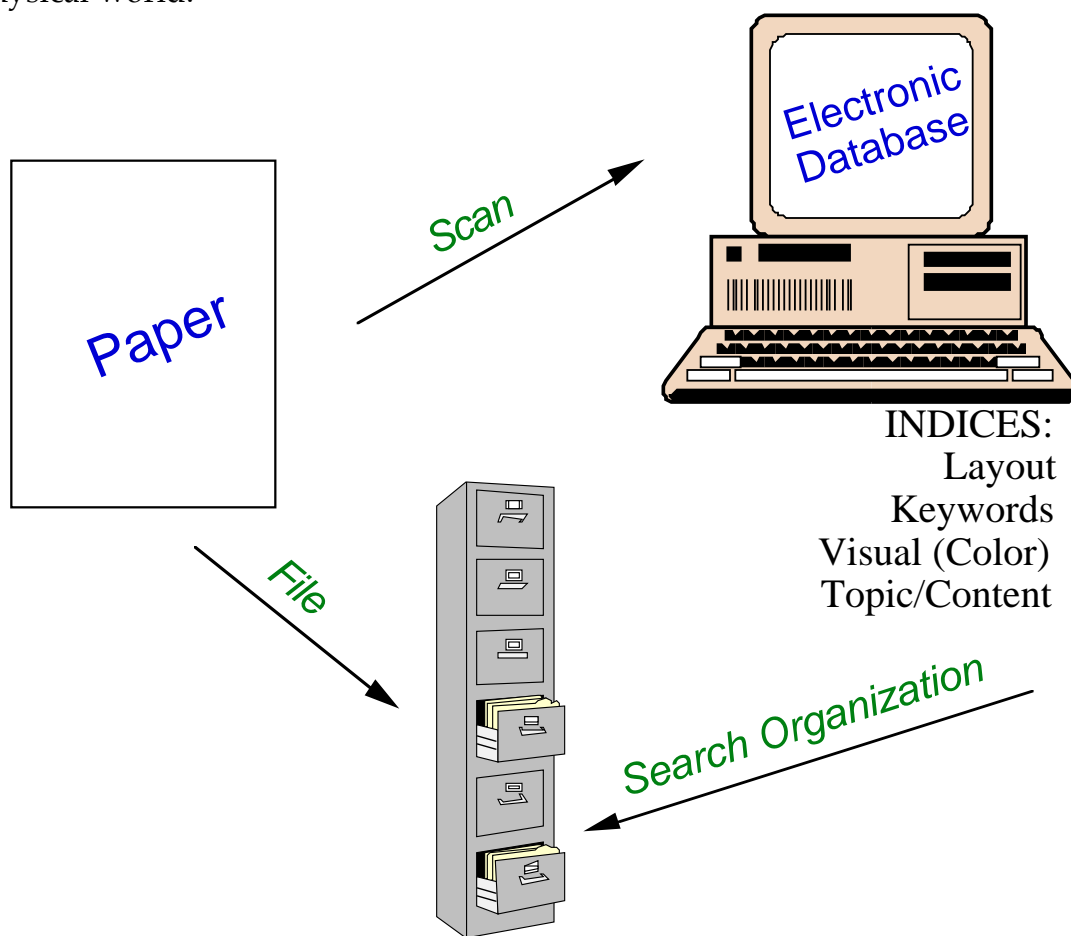


Figure 1: This image represents the relationship between the physical object and the electronic database. A paper is entered into the electronic database by capturing its contents using a scanner. The paper is then placed in a drawer in the filing cabinet. The electronic database indexes the paper and gives pointers to the location of the paper in the physical cabinet. Search algorithms can be used to find the paper in the electronic file cabinet.

There were three main technical challenges encountered when designing the self-organized file cabinet. One was in designing a database that would support multilevel indices of information (physical indices, text indices, and visual indices). Two was in designing efficient search and visualization methods. Three was in designing a GUI that was natural to use and correctly captured the metaphor of the self-organized file cabinet. Thus, the self-organized file cabinet consists of three main modules: the file cabinet structure, the searching methods, and the graphical user interface. These modules are described in detail below.

3.1 The File Cabinet Structure

The file cabinet database is structured so that it mirrors a file cabinet in the physical world from the design of the structure to the way in which manipulations are performed. For example, in order to retrieve a paper, one first thinks of retrieving the paper from the cabinet. The first question that needs to be answered is which drawer. Once a drawer is known, the task can now be thought of as retrieving a paper from the drawer, and the files are thumbed through to find the correct one.

We designed classes in C++ to make normal operations on a filing cabinet as natural and intuitive as possible. The parent structure is the filing cabinet itself. A filing cabinet consists of drawers. The implementation structure of the drawers is a dynamically allocated array. The textual and visual index modules are also included. The textual index is an augmented Smart object developed from the Smart Information Retrieval System[Sal91]. Smart examines the statistics of word occurrence in a document relative to a document collection. It uses the statistics to create vector space model for the document collection that supports queries using an algorithm for computing

vector proximity in the vector space. The indexing allows the system to quickly find documents that match a given query and rank these documents by how well they match the query. In addition to word queries, we also support color queries. The color index is a Color Database object which uses colors to index all documents. Because of the nature of this file cabinet, a few other details are important. One is a translation between the identification number that the textual and visual databases use and the paper's location within the file cabinet. The file cabinet also names the drawers so that a drawer once created can always be referred to by the same name even if the number and placement of drawers change over time. For a complete list of data and functions see Figure 2 (below).

More specifically the data fields of a drawer are as follows:

- Provides a storage place to keep track of all papers and their order by using an ordered list of papers (Figure 2). Papers are organized by date filed. The newest paper appears at the front of the list.
- The integer that was assigned to it by the filing cabinet is another data field. This number is unique and never used for another drawer again (Figure 2).
- The integer that will be assigned to the next paper added to the drawer is also crucial. The integer is unique and never assigned to more than one paper even if the paper is removed (Figure 2).

Drawers consist of documents. Each document representation contains the following information (also enumerated in Figure 2):

- The name the drawer gave the paper through the paper ID.
- The name the indices gave the paper, the Smart ID.
- The date that the paper was filed.

- The Filter Data which consists of the names of two files that contain the electronic information about the paper.
- Its location in the file cabinet i.e.. the drawer number.

The file cabinet, apart from remembering information about its contents, also controls all manipulations of everything within the file cabinet.

Functions of the File (see Figure 2):

- Adding a paper (int AddNewPaper(char *, char *, Location))
- Removing a paper (int RemovePaper(Location))
- Finding a paper (int FindPaper(Location, int, Paper *))
- Moving a paper (int MovePaper (Location, int, Location))
- Adding drawers (int AddDrawers(int, int *))
- Removing a drawer (int RemoveDrawer(Location))
- Summarizing a drawer (void PrintDrawer (Location, ofstream &))
- Counting the number of papers within a drawer (int NumPapersInDrawer (Location))
- Summarizing the file cabinet (void Print ())
- Counting the number of papers within a file cabinet (int NumPapersInFile ()).
- In order for the file cabinet to be persistent, it is also necessary to have a mechanism to load the file cabinet (friend int load_file (ifstream &, File *, char *))
- Save the file cabinet (friend int save_file (ofstream &, File *, char *))
- Telling if a drawer or file cabinet contain any papers (int IsDrawerEmpty(Location) or int IsEmpty())
- Counting the number of papers before a specific paper within in a drawer (int PaperCount (Location, int))

- Returning the location of a paper within the cabinet given its name in the textual and visual indices (int translate (int, FileID))

Each Drawer data structure provides the following methods for interfacing with the data (also see Figure 2):

- Count the number of papers in a drawer (int NumPapers())
- Find out how many papers proceed a given one (int PaperCount (int))
- Allows one to have access to the drawer name and the smart identification number (void AssignDrawerNum(int), int ReturnDrawerNum(), and int addSmartID(int, int),)
- Add a paper (int AddNewPaper { (FilterData*), (FilterData*, int), (Paper)})
- Remove a paper (int RemovePaper(int))
- Retrieve a paper (int FindPaper. (int, Paper *))
- Summary of the drawer (void Print {(),(ofstream &)})
- Load a drawer (friend int load_drawer (ifstream &, Drawer *))
- Save a drawer (friend int save_drawer (ofstream &, Drawer *))

Each Paper data structure supports the following interface (also see Figure 2):

- One is able to change the date filed (void MakeDateCurrent())
- Change the identification number that the drawer assigns it (void ChangeID(int) or int ReturnID())
- Access to the smart identification number (void addSmartID(int) or int ReturnSmartID())
- Summarize the data about a page to varying degrees (void Print {(),(ofstream &)} or void PrintAll ())
- Load a paper (friend int load_paper (ifstream &, Paper &))
- Save a paper (friend int save_paper (ofstream &, Paper &))

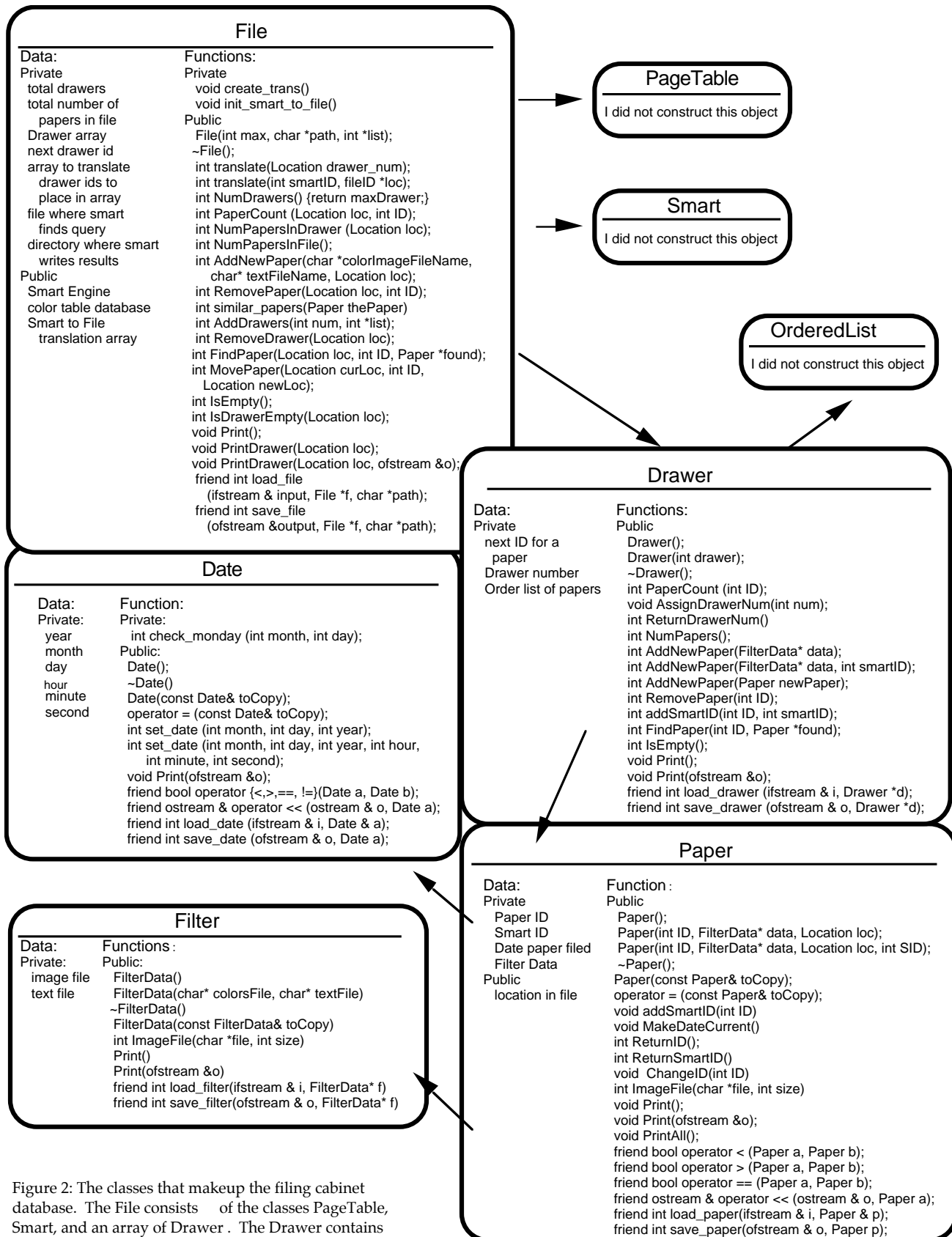


Figure 2: The classes that makeup the filing cabinet database. The File consists of the classes PageTable, Smart, and an array of Drawer . The Drawer contains an OrderedList of Papers. Finally a Paper consists of a Filter and a Date.

3.2 The Search Algorithms

There are three different types of searching methods available. The file cabinet can be queried on keywords, with color and placement of color, and documents related by text.

The keyword queries use an augmented version of the Smart System [Sal91], which is a sophisticated text-retrieval system. Smart uses a vector space model for document retrieval. The vector space is defined by the important words occurring in the corpus. This version of Smart has been augmented to incorporate the layout of a paper to give more information about topics and subtopics discussed in the paper [RA96]. Similar documents are connected by hyperlinks which allow for the comparison between blocks of texts of similar types of information. Hyperlinks are actually edges of a weighted graph determined by the Hausdroff shape comparison metric [HR93].

There are three components used within Smart. First the segmenter automatically divides the paper into units. Then the classifier automatically labels the units produced by the segmenter with document types. Types have been determined by classifying attributes of a research paper: sections, subsections, theorems, proofs, definitions, figures, figure captions, paragraphs, itemized lists, and tables. Finally, the retriever finds the responses to user queries.

Smart was augmented to also support color, layout, and table indices. The color index divides a page into subsections corresponding to a 3 by 3 grid. It then creates a histogram by counting the number of times a pixel appears of a given color. There are twenty-four recognizable colors which each

represent a range of RGB color values. The algorithm decides the three most prevalent colors within that section of the page and indexes them. Queries can be submitted for the color. One can narrow the search by specifying a region of the document.

To support the ability to find related papers, the Online Star Algorithm was incorporated into the file cabinet. This incorporates the information provided by smart to produce a similarity matrix. This is a post-processor algorithm that gives a "hierarchical organization of a collection into clusters. Each level in the hierarchy is determined by a threshold for a minimum similarity between the pairs of documents within a cluster at that particular level in the hierarchy" [APR]. The star algorithm produces dense clusters that approximate cliques with probable guarantees on the pairwise similarity between cluster documents. Through user studies it has been determined that this algorithm has a good performance. The clusters provided by the algorithm matched fairly closely to clusters made by humans organizing the documents.

3.3 Graphical User Interface

The GUI was spatially designed with four quadrants as shown in Figure 3 (see appendix). In the upper left hand corner a graphical representation of the file cabinet appears. In the upper right hand corner, a picture of a file cabinet is shown. In the lower left hand corner, the search manipulations appears. The output from the search is printed in the scrolling text area. In the lower right hand corner there are a series of buttons that allow for the manipulations of documents and the file cabinet as well as buttons pertaining to the clustering by topic features.

The display of the file cabinet draws a picture of the file cabinet. The file cabinet is blue with drawers outlined in yellow on a black background.

Each drawer is actually a side view of the open drawer rather than a front view. This enables the interface to display the approximate position of a paper within the drawer for the find results. When a drawer becomes full, many papers could be represented by one line. These papers appear red. The results of most finds yields more than one paper. In order to figure out exactly which paper is the one the user is interested in, the user can highlight a specific paper in green. Only one paper is allowed to be green at any given time.

A picture of a file cabinet appears in the upper right hand corner. This was added for aesthetic appeal. For example, it makes the interface interesting. One user complained about it wasting space. In the future this space could be converted into something more useful. It could be used to make the drawing space larger or to add functionality to the file cabinet.

The lower left part of the screen is devoted to the searching mechanisms. The top part of this section is used to set colors for which to search. One constructs colors by mixing red, green, and blue to form twenty-four different colors. The color that is the product of this mixture is displayed to the right of the sliders. Above this color display are nine radial buttons. These are used to specify the part of the paper where one believes the color appears. For example, users would use this when they thought that a red table was in the lower right corner of the page. They would construct red and select that part of the page and then press the "Color Search" button to display the results of the search. Next to the "Color Search" button is the "Combo Search" button. A combination search takes the results of a text query and color search and displays only those papers that are found by both searches. Below the color search part of the display is a line to enter text queries. The text query commences by pressing enter in that box. The results of both the

color search and the text query are displayed in the scrollable text area at the bottom of the screen. The color search displays pages by page identification number, drawer row, and drawer column. The text query also includes information about the similarity of the document to the query. The document that is most similar to the query is the first one in the output. A sample search is shown in Figure 4 (see appendix).

The lower right part of the screen consists of all the buttons used to manipulate the file cabinet as well as the ones related to clustering documents by topic. These buttons included changing the size of the file cabinet; adding, removing, retrieving, and moving a paper; summarizing and removing a drawer; and clustering the whole file cabinet or finding the clusters that include one particular paper to discover similar papers.

The first button deals with changing the size of the file cabinet as shown in Figure 5 (see appendix). One enters the number of columns and rows to be in the new file cabinet. If this makes the file cabinet bigger than as long as it is not too big (more than twenty-four rows or columns) the new file cabinet will be redrawn in the space. If the file cabinet contained any papers, they remain in the same drawer specified by row and column. Drawer one, one is always drawer one, one. If the change in size would remove any of the drawers, each drawer to checked to make sure it is empty before the operation is allowed. This is because it is unlikely that users would want to throw away all the papers contained in a drawer. The file cabinet is then redrawn in the display area.

3.3.1 The Algorithm for Adding a Paper

To add a paper, one needs to specify a drawer by row and column number as well as the two files that contain the graphical and textual information about the document as shown in Figure 6 (see appendix). Before

a paper is added, both files must exist as well as the drawer specified. Even if the particular document does not contain any ASCII text, a blank file must exist. The document is then placed in the front of the drawer and indexed in the color and textual databases.

3.3.2 The Algorithm for Removing a Paper

When removing a paper, the paper is identified by its page identification number as well as the drawer which contains it. When a paper is removed it is deleted from the file cabinet database as well as removing all references to in within the color and textual indices.

3.3.3 The Algorithm for Retrieving a Paper

To retrieve a paper, one must again use the page identification number and drawer to locate it as shown in Figure 7 (see appendix). This function displays its placement within the file cabinet and well as providing more information through a pull down menu. The pull down menu has three attributes. "Get Info" tells how many documents are in front of the paper, the name given to the text file, and the date it was placed in the drawer. "Display Page" displays the tiff file so that one can verify that it is in fact the paper of interest. "Highlight in Drawer" turns the display of the paper green.

3.3.4 The Algorithm for Moving a Paper

"Move Paper" asks the user to identify the paper by page indentation number and drawer. It also needs to know the new drawer which is also identified by row and column number.

3.3.5 The Algorithm for Summarizing a Drawer

Summarizing a drawer provides a window that lists every document in the drawer along with the information provide by "Get Info". One can learn the number of papers in front of the document, the name of the text file, and the date the document became a resident of that drawer.

3.3.6 The Algorithm for Removing a Drawer

"Remove Drawer" is provided for fine tuning the display of a file cabinet. Although most file cabinets are in the shape of a grid, some peoples file cabinets are spread throughout the office and have different numbers of drawers in varying columns. By removing drawers, one can come up with the same configuration that exists in the real world. Again, the program will not allow the removal of a drawer with papers in it.

3.3.7 The Algorithm for Clustering the File Cabinet

To cluster the file cabinet, a threshold needs to be provided between zero and one. This threshold specifies how closely related the documents are in each cluster. The closer to one the threshold is, the more related the documents will be in any given cluster. The clusters are displayed in a window as shown in Figure 8 (see appendix). If a cluster is selected, the papers within the cluster will appear in a "Find Results" window. This window allows all the features of a find: "Get Info", "Display Paper", and "Highlight in Drawer". It also displays all the papers within the cluster on the file cabinet display.

3.3.8 The Algorithm for Finding Similar Papers

"Cluster on Paper" also requires a threshold to specify how closely one would like the documents to be related. In addition, a paper must be identified by page identification number and drawer. All the documents found are listed in a "Find Results" window and displayed in the file cabinet.

In the Menu Bar there are two pull down menus. One is "File" which allows one to load and save a file cabinet and to quit the application. For load and save one is asked to specify a full path directory. The second pulldown menu is labeled "Preferences". As of this time, the only option is to "Set

Image Directory" which specifies the directory where all scanned documents in the file cabinet are located.

4 Experiments

Since the self-organized file cabinet is a novel concept, there is no standard methodology for evaluating it. The following plan was implemented: (1) evaluate each software module for correctness and completeness by designing an extensive set of benchmark operations, (2) evaluate each software module for performance by measuring the response time to each query, and (3) evaluate the naturalness and usefulness of the system by designing a user study.

4.1 Correctness and Completeness

The software was first tested in the three separate parts described in section 3 for correctness before they were integrated and the system tested for completeness. The file cabinet was first tested using a text based interface. Limits were checked such as making sure that one could not declare a negative size file cabinet and that pages were added correctly. The retrieve function was tested to make sure that it only found a document when one existed. Remove was tested to make sure that a document was actually removed from all the databases. The designers of the search algorithms, Katya Pelekhov and Hans Kieserman, tested the search algorithms separately for correctness and completeness. The user interface was then tested to make sure that it prevented errors from occurring such as making sure that the files actually exist that are specified during an add and disallowing the user to perform operations on non-existent drawers.

4.2 Performance

The system was tested to make sure that all operations occurred at a reasonable speed so that users would not become impatient with the system. The results of this test are shown in Figure 9 (below). All time consuming

tasks only need to be performed once while tasks that are performed frequently, such as searches, are much faster and, usually instantaneous.

Function	Time
Scan	2 minutes
Load	3 seconds
Save	1 second
Change Size of File Cabinet	~0
Add	4 seconds
Remove	6 seconds
Retrieve	~0
Get Info	~0
Display Page	6 seconds
Highlight in Drawer	~0
Move	~0
Summarize	~0
Remove Drawer	~0
Cluster File Cabinet	3 seconds
Cluster on Paper	3 seconds
Color Search	~0
Text Search	2 seconds
Combo Search	2 seconds

Figure 9: This is an approximation of the amount of time each function takes to run. Scanning is the most time consuming, but most functions seem instantaneous or do not take long at all.

4.3 Naturalness and Usefulness

Most of the testing emphasized the user interface and user studies. The goal of the user study was to determine if the program would be usable. In

order for a tool such as the self-organizing file cabinet to be worthwhile, people have to believe that it is worth changing their old habits to incorporate the new technology into their lives.

We used three different methods to find out if such a system would be used by the general public. First we sent out a survey to about seventy Dartmouth students. We then discussed the file cabinet with 12 people during the Women in Science Program poster symposium. Finally, we spent an hour with six different professors, graduate students, and undergrads. They used the system we designed and gave us critical feed back. In each of the three different types of tests, we had an overwhelming positive response.

4.3.1 Survey

We conducted the survey by BlitzMail. All of the members of the sorority Delta Gamma and the fraternity Sigma Nu received a description of our file cabinet and were ask to give a yes or no response as to whether they would use it or not. The blitz was written as follows:

Hi,

I'm working on my honors thesis in Computer Science. I've design a Self-Organizing File Cabinet and now I'm polling people to see if they would be interested in using such a tool.

The way the file cabinet works is that you scan the paper you are entering and tell the computer which drawer to want to put it in and then stick in the front of that drawer. When you go to look for the paper, you use the computer to search for it based on keywords or color. So instead of trying to remember how you filed something, the computer remembers the location and tells you where it is. If someone wanted to give you this product, would you use it? Yes or no is fine.

thanks for reading and responding,

Dawn

Seventy-two people received this blitz, thirty-three women and thirty-nine men. Seventy-six percent of the women responded while only twenty-eight

percent of the men did, so the total response rate of the survey was fifty percent. The percentage of people that said they would like to use the self-organizing file cabinet did not vary significantly between men and women as shown in Figure 10 (below). Eighty-six percent of the group who responded said they would use this product. The difference in response did come from people who were unsure whether they would use it. Women expressed an uncertainty about being able to trust the system while men were concerned about the time it would take to scan a document. There was one woman in the former category and one man representing the later opinion. Overall people seemed to be very excited about the idea, although the group might have been slightly biased since they all knew who I was.

	Surveyed	Respond	% Respond	Yes	% Yes	No	Technophobe	Inconvenient
Women	33	25	76 %	22	88 %	2	1	0
Men	39	11	28 %	9	82 %	1	0	1
Total	72	36	50 %	31	86 %	3	1	0

Figure 10: Reports the results of the email survey. This reveals two types of information: one that such a product is desirable and two that men and women are equally interested in it.

4.3.2 Poster Symposium

At the poster symposium we talked with twelve students, faculty, administrators, and members of the Hanover community. The students were interested in the project but were not able to relate it any real world experience so did not have any critical questions pertaining to the file cabinet. The faculty, administrators, and members of the Hanover community on the other hand were able to relate this project to their own experiences in keeping a file cabinet. All left our poster with the desire to have a self-organized file cabinet in their office.

We also were able to find out about extensions that different people would want their self-organized file cabinet to be able to do. One person wanted the file cabinet to be able to learn so that as time past, it would be more likely to find the document that one was looking for in the front of the list of found documents. This type of technology is becoming available and could easily be added to the file cabinet since the user is always the same. Another person wanted to add electronic documents to the file cabinet that wouldn't have any place in the physical world. This could be implemented by adding a "dummy" drawer to store all electronic documents and also keeping a black and white image that could be entered as the tiff file for all electronic documents.

While explaining the file cabinet, it became imperative to have a "desktop" drawer, so that while papers were being read, they would still remain within the file cabinet system, but if the same paper showed up in a search, the computer would not direct one to a drawer in the file cabinet when the paper was not actually in the file cabinet at the present time. This also made it easier to replace documents because there would be no need to put it back in the same drawer, never mind the same part of the drawer. This would be easy to implement. Drawer 0 could automatically be assigned as the "desktop" drawer and desktop could be used as a keyword to refer to this drawer for the remove, retrieve, and move functions. The desktop could be summarized like any other drawer. When results are displayed, 0,0 could be the coordinates used so that the graphical interface knows to write "Desktop" as the location rather than "Row: x, Column: y."

4.3.3 User Test Studies

The user studies were much more revealing of attributes of the file cabinet that were difficult or uncomfortable to use. The design in general, however, was easy to understand. To instruct people in using the file cabinet, we showed them the graphical interface. We read each button and then showed them the results of the "Cluster File Cabinet". We then pointed out the pull down menu on the "Find Results" and showed them what each of the selections did. Finally, we demonstrated a color, text, and combination search before quitting the application and allowing them to use it. We gave them a piece of paper with sixteen tasks as follows:

load: /usr/plum/lawrie/save
set preferences: /usr/plum/lawrie/images

Find out how the documents in the file are related.
Examine a paper in the file cabinet in more depth.

Make the file cabinet bigger.
Add page 102 to a new drawer
Print the drawer.
Move the paper to another drawer.
Print that drawer.
Find the paper in its new location.
Remove a drawer.
Remove the paper that you added.
Make the file cabinet smaller.

Find the articles written by Doug Riecken
Look at page 63 in the magazine: what might you remembered about this page a month later and then try to find it based on those details.
Search for the article on pages 72-76.

The main complaint of the users as shown in Figure 11 (below) was that they didn't like trying to construct a color, and they wanted to do more things with the mouse and less with the keyboard. Another feature they wanted was to have the cabinet recommend a drawer. It also became obvious that logical operators would make search strings more specific. In general

they wanted the system to have more memory. These users also would like to be able to assign titles to pages rather than see the identification number assigned by the file cabinet. Finally, many expressed an interest in adding information to what smart was already indexing in case the first page of the document did not include their interest in the paper.

	Number of People (out of 6)
Dislike the Color Set-up	5
Requested more Mouse Interaction	5
Requested more Memory in System	4
Logical Operators for Searches	3
Assign Titles to Papers	3
Recommend a Drawer	2
Add Keywords to Papers	2

Figure 11: This table lists some of the reactions by the users and how many of those users had the same reaction.

Constructing a color was very problematic. One thing is that people don't really know which mix of colors makes the one they are looking for so it is not very fast to have to build each color. The second problem was that while someone might call a color one name, they might select another if they were trying to match colors. This was visible in the a picture on page 63 of the Communications magazine we used as a document of the file cabinet. The color appeared green, but was actually yellow-green. None of the subjects instinctively recognized the color as yellow-green. Had they been asked to select a color from among the twenty-four available, it is much more likely that they would have used the yellow-green as the color because it was "closer" to yellow-green than green. To solve this problem, using buttons would be a much better solution. The human brain is better at matching a

colors than constructing them. Hans Kieserman has already added this new feature to the Self-Organizing Desk and the new buttons can be designed for the file cabinet from that. I believe it would be best to have four rows of six colors in the space where the sliders are now located.

When it comes to making the file cabinet more mouse friendly, we are slightly hesitant. We do not want it to be too easy to move papers around because this file cabinet is suppose to duplicate the one in the real world, but it also cannot be too cumbersome because people will not want to use the system. We believe that one should be able to summarize a drawer by clicking a the left mouse button over the drawer on the file cabinet display. When removing, retrieving, or moving a paper, the last paper examined should appear in the window. If one clicks the right mouse button over a drawer while the move window is active than that drawer becomes the drawer the paper will move to.

Moving a large number of papers to the desktop would be very convenient. A system could be created so that buttons in the "Find Results" could be marked to go to the desktop by providing that option on the pulldown menu and then a button could be added to every "Find Results" window so that all those marked documents would move to the desktop drawer. Marked buttons would be another color so that the user could easily tell which papers he or she intends to move. When returning a paper to a drawer, an option on the paper could be to recommend a drawer. A window would appear with the recommendation and an accept button. If the user clicks on any drawer with the right mouse button, the drawer would change to that one and then the accept button would be pressed. The program would then show the paper in the front of that drawer.

To have the file cabinet recommend a drawer initially, only the scanned files would be needed as initial input. The page could be added to the desktop. As a second part to the add, the same code could be used that recommends a drawer to a paper on the desktop by clustering the file cabinet on the paper at a predetermined threshold like 30 percent. The computer then finds the drawer that contains the most number of similar papers and recommends that drawer. The user again has the ability to accept or select another drawer and accept.

People wanted to be able to use logical operators within the text queries. This would require multiple searches with smart. A function would need to be written that parses the string where logical operators appear and then collects the information from smart before another search is done. The results would then be put together. The problem comes when figuring out what to do with the similarity. The advantage of smart is that the first few files it returns are most likely to be the ones of interest. A constant way of ordering the results would have to be found to keep this advantage while allowing the logical operators. The operators "and", "or", and "not" would all be easy to implement with this design.

Users found the "Find Results" window intimidating because all the buttons basically looked the same. They were not easily distinguished so people forgot which button they had clicked. The last button clicked should change to some other color besides the default color, so that the user can easily tell which paper he or she is looking at.

Memory of the system also deals with the actions that were last performed. If the scroll-text area also displayed the function calls for each button pressed, it would be easy to allow input into this area using the function calls. People would then have the option of using the keyboard

rather than the mouse. This would also enable the user to implement scripting.

Another request was to allow the user to title a document when adding it to the file cabinet. Too things are an issue here. One is that the system needs to be able to determine if the name is unique, and it will also have to search the entire file cabinet to find the paper with this title. Although such a search would be easy to implement it would be slow. Another solution would be to place the title in a hash table along with its location within the file cabinet. This would be much faster when trying to find a page or determine if a name is unique. One would just have to make sure that on a move or remove the hash table was also updated.

Adding keywords to the document was a feature that many desired. If only the first page of a paper is scanned and added to the file cabinet, it is possible that the reason the person is filing the paper is not discussed on the first page. It would be very easy to append a copy of the OCR'd file with the added keywords and have smart index the new file. The keywords could be kept with the filter data and if in the future, the user changed the key words, the paper could be reindexed by Smart. One user also desired that his keywords be given more weight than what appeared on the page. This might also be able to be implemented using Smart.

People had many other concerns besides the ones enumerated above depending on their special expectations of the file cabinet. One user had his file cabinet spread throughout his office. He did not want a block file cabinet. Instead he wanted the change file cabinet button to give him a window. If he asked for a specific file cabinet, one would be drawn, but he would be able to place it on the screen. He also thought that removing a drawer should be within the same window. When he pressed the button to remove a drawer,

he would be given the destroy icon used by UNIX. By placing it over a drawer, the drawer would be removed.

Another user asked that the cursor change when the system was busy so he would know when he could continue. This will reassure users that the program did in fact register the input.

When a paper is added or moved, the users want to see it in its new location. This would be easy to implement by running "Retrieve Paper" so that all the information about a paper is automatically available. This would involve returning the page identification number and then calling `find_page`.

I feel that all these changes described above are feasible to implement and will make the self-organized file cabinet a more user friendly and usable system. Overall the response to the file cabinet has been extremely positive. There is definitely a market for such a tool.

5 Conclusion and Future Work

The self-organization metaphor has been used to implement a self-organizing file cabinet. As this thesis shows, not only is it possible to design and implement a physical reality augmented by electronic data, it is highly desirable. The tool that has been designed is easy to use and friendly enough for people to use a model that is only slightly modified from its present state.

Of course, there are some things that need to be done before consumers have the opportunity to purchase the self-organized file cabinet. Changing the interface with the visual index and well as providing more mouse operations are imperative.

There are also other avenues for which the file cabinet could become responsible. These include notifying the user through email that something is due. The file cabinet could also help out with spring cleaning by letting the user know about documents that have not been read in years. Another feature that allowed the user to scan documents from the file cabinet application would also be desirable.

The self-organized file cabinet is by no means a perfect tool at this time, but the additions needed are not extremely complicated. This product will be very useful in the future.

6 Acknowledgments

I would like to thank my advisor, Daniela Rus, for all her support and enthusiasm during my thesis. I would like to thank both Hans Kieserman and Katya Pelekhov for working with me during the integration of the file cabinet and the search indexes. I would also like to thank those that participated in my user study, Javed Aslam, Tom Corman, John Danskin, Ashan Kabir, Katya Pelekhov, and Cliff Stein. I would also like to extend my appreciation to the members of Delta Gamma sorority and Sigma Nu fraternity for their participation in my survey.

7 References

- [AM+94] T. Arai, K. Machii, S. Kuzunuki, and H. Shojima, InteractiveDesk: A computer augmented desk which responds to operations on real objects, in *Proceedings of Computer Human Interactions*, 1994.
- [APR] J. Aslam, K. Pelekhov, and D. Rus. Generating, Visualizing, and Evaluating High-Quality Clusters for Information Organization, forthcoming Technical Report.
- [HJT97a] F. Heylighen, C. Joslyn, and V. Turchin. "SELF-ORGANIZING", *Principia Cybernetica Web*, <http://pespmc1.vub.ac.be/ASC/SELF-ORGANG.html>, (21 May, 1997).
- [HJT97b] F. Heylighen, C. Joslyn, and V. Turchin. "SELF-ORGANIZING SYSTEM", *Principia Cybernetica Web*, http://pespmc1.vub.ac.be/ASC/SELF-O_SYSTE.html, (21 May, 1997).
- [HR93] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdroff distance, in *IEEE Transactions on Pattern Matching and Machine Intelligence*, 1993.
- [PS+96] P. Pirolli, P. Schank, M. Hearst, and C. Diehl. Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection. In *Human Factors in Computing Systems CHI '96*, pages 213-220.
- [RA96] D. Rus and J. Allan. Does Navigation Require More than One Compass? In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, Coirns, Australia, 1996.
- [RD97] D. Rus and P. de Santis. The Self-Organized Desk. In *Proceedings of the 1997 International Joint Conference on Artificial Intelligence*, Nagoyo, Japan, 1997.
- [Sal91] G. Salton. The Smart document retrieval project. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 356-358.
- [Tor95] Mark C. Torrance. Advances in Human-Computer-Interaction: The Intelligent Room, in *Working Notes of the CHI 95 Research Symposium*, 1995.

[Uns93] Cem Ünsal. Self-organization in large populations of mobile robots. Thesis (M.S.)--Virginia Polytechnic Institute and State University, 1993. <http://armyant.ee.vt.edu/unsalWWW/cemsthesis.html#cont> (21 May 1997).