

DocPlayer: why doesn't your desktop play this way?

Kevin McGee

Department of Computer and Information Science
Linköping University
58 183 Linköping
Sweden
kevmc@ida.liu.se

Jody Foo

Department of Computer and Information Science
Linköping University
58 183 Linköping
Sweden
jodfo042@student.liu.se

ABSTRACT

Document-management is becoming a central and troubling issue for typical users of computers; and the need for more intuitive, flexible, and effective management-systems is becoming more and more evident. This paper reports on work to develop DocPlayer, a solution inspired by the control-metaphor of media-players. First, other attempts to address the document-management problem are reviewed; then the DocPlayer system is described – along with a brief summary of reactions from users. The paper concludes with a brief analysis of the DocPlayer solution and some proposals for future research.

Author Keywords

Media-player Model, Non-hierarchical Document Organization, Document Management Systems, Design Analysis

ACM Classification Keywords

Interaction techniques, GUI, File Organization, Information Search and Retrieval

INTRODUCTION

Document-management is becoming a central and troubling issue for typical users of personal computers; and the need for more intuitive, flexible, and effective management-systems is becoming more and more evident.

It is becoming more and more common for the personal computer of a typical user to contain tens (or even hundreds) of thousands of files. Many of these are not of direct interest to the typical user, but many of them are important documents of different types, serving different functions, and coming from different sources at different times. Most users have very limited tools to help them with these management tasks -- mainly the file manager/browser included with their operating system. Even in the context of reasonably small document-collections, there are well-

known dilemmas associated with organizing (and re-organizing) a document collection – and using it to store and retrieve documents.

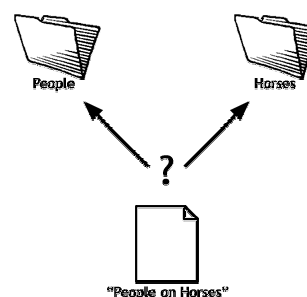


Figure 1. Multiple categorization dilemma

Consider a person who has one folder for documents about people and one for documents about horses. What does this person do with a new document about “People on horses”? The standard solutions on existing personal computers are limited and far from effective.

SURVEY

This section will review previous research into the technical proposals for addressing document-management issues – as well as work as it relates to different document-management activities.

Improving document management: three approaches

The different attempts to improve the document management situation for the user can be broadly classified in terms of three approaches: changing the user interface; adding automated and/or intelligent system components; or implementing more flexible/powerful representational structures to organize or store documents.

Improving the (Graphical) User Interface. This first approach focuses on strategies to make document management tasks easier without changing underlying mechanisms. The goal is to shift the cognitive load associated with navigating information structures to the human perceptual system. Some work has been done to extend the desktop metaphor by letting users organize documents spatially [24]; by providing the user with a better view of the existing hierarchical structure [15] (and see [16]

for an overview); and by using different techniques to avoid distortion [5,23]. An alternative is to explore alternatives to the desktop metaphor, such as “piles” [20] and “galaxies” [10].

Using automated or intelligent components. The use of automated components in document management systems tends to be limited to systems dealing with very large and mostly unfamiliar document collections, such as online libraries and other non-personal document collections. Automated approaches range from automatic clustering based on document-analysis [17], to assisted information retrieval (e.g. used by many search engines) – and systems have been implemented that use statistical algorithms [26]), as well neural networks [11]. While clustering uses document analysis, there are other less advanced methods that automatically group documents for users. Property or content based systems [4] allow users to access *fluid collections* or *virtual directories* that consist of documents that have been automatically gathered by the system as a result of a query or rule applied on the document collection. Such a rule might be “gather all documents by author Baz”. Finally, there are systems implemented in terms of intelligent agents [12].

Using more flexible information structures. Several attempts have been made to provide users with a non-hierarchical management systems that provides multiple categorization. Some of these systems [4,7,9,22] use a set-based approach instead of hierarchical trees to organize files. The set-based approach decouples document organization and document storage – thus, a document can belong to several organizational groupings (which can be hierarchical) without needing to be duplicated. The main differences between these systems is the level they target (file system or application level) – and what the sets represent (temporal dimensions, file properties, etc.).

Document management activities: Filing, Locating, & Managing

In addition to work that concentrates on different solutions, there is work that tends to be more task-centric, addressing such document-management activities as filing, managing, and locating documents [4].

Filing: ‘Where do I put this document?’ For various reasons, few systems attempt to change the underlying file-system. Thus, systems mentioned that try to improve the visualization of the hierarchical file-structure tend not to address problems inherent to the hierarchical storage model. Set-based approaches that decouple file properties/content from location let users store documents in multiple logical locations. The notion of *virtual directories* or *fluid collections* [4] helps the user by providing a mechanism to automatically and dynamically file or “gather” documents into specific, logical locations; implementations based on temporal metaphors [7] can completely remove the need to file documents; and automated system components can also help the user with the filing task by extracting document properties to be used by for instance virtual directories/fluid

collections [4,10], or by automated filing using rules similar to those used by e-mail software [20].

Management: ‘How do I (re-)organize these documents?’ Managing documents using a file system inherits file management issues such as file dependencies and storage space allocation. Approaches that exist independently or replace the hierarchical file-system can eliminate management issues (e.g. taking care of file dependencies) related to file system. Systems that decouple the document collection from the file system also have the potential to go beyond the file system by, for example, attaching more content-related properties (such as document author, and document title) to documents.

Finding: ‘Where did I put that document?’ Much of the work on document-management is devoted to solving problems of retrieval. Browsing directory structures is inherent to file systems and logical search using at least file names is implemented in almost all operating systems. Different approaches have different advantages and disadvantages, and depending on the scenario, they can be more or less effective; however, browsing seems to be the preferred method for most users, so approaches to improve document location include alternate hierarchy visualization [3,15,23]; automated document clustering or organization sometimes used in conjunction with optional visualization [1,11,17]; using a time centered metaphor [7]; taking advantage of spatial cues and spatial memory [3,5,24]; and implementing new interaction components [20].

RESEARCH PROBLEM

In file systems today, the filing and management of documents are integrated tasks; as Gemmel writes, “Computer file names mangle together the concept of ID, name, physical location and hierarchical organization” [9]. Many of the problems encountered in document management are related to the underlying hierarchical file system – and solving this is still an open research problem.

When document-management is tightly coupled to a hierarchical file system, users tend to be frustrated and confused when filing and retrieving documents that can reasonably reside in more than one location in the hierarchy. Not to mention the difficulty they encounter when creating or revising such hierarchical document-structures. As noted in the survey, there are a number of efforts that explore decoupling file storage from document management. This decoupling is starting to occur not only in specific applications (such as address books and the like), it is also being used to a certain extent in contemporary interfaces to file systems (in the form of top-level “virtual folders” and the like). Nonetheless, the degree to which it is convenient and flexible to make use of these mechanisms remains quite limited.

For document-management, one approach that does not seem widely-explored is to take the interface and representational model of media-players (see Figure 2 for an example media-player interface) and apply it to document

management. Not only do media players include the ability to play music (and other) files, they also provide a fairly intuitive interface and a set of control-mechanisms for grouping files into collections, for “tagging” these files with various keywords, and for quickly finding and sorting files that meet certain criteria. Additionally, some media players offer mechanisms for automatically generating different file properties and categories of use (“most recently played”, “most popular,” “recent imports” and the like).

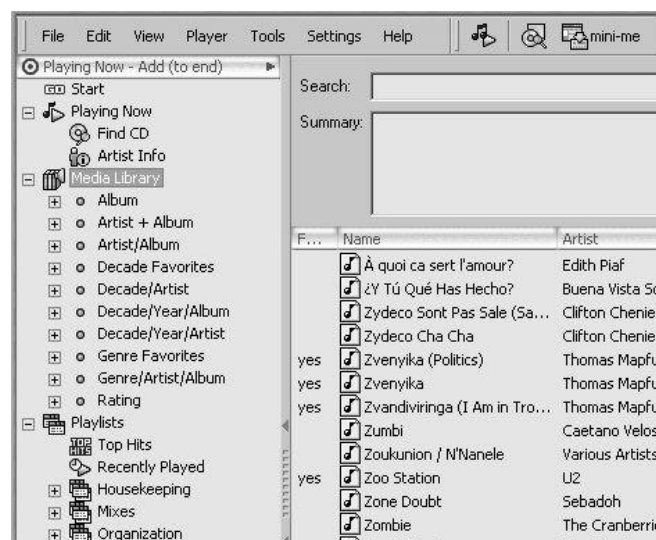


Figure 2. Media Jukebox media-player

Thus, it seems possible that the media-player model could empower users who perform document-management tasks in two significant ways. The very familiarity of the media-player model could facilitate ease of learning – and the particular control mechanisms could empower ease/flexibility of classification, storing, and finding. The research goal was, then, to test this idea: to see which aspects of document-management (if any) work well using the media-player model – and to see if such a model helps people begin and continue managing their documents.

METHOD

The research method can be summarized as follows. A document-management “player” – *DocPlayer* – was implemented in Java 1.4.2 using a MySQL 4.0 database to store and manage document-collection information. (For a more detailed description of DocPlayer, see [8].) During the implementation process a number of different document types were analyzed and potential document-management features were implemented. Finally, a number of informal sessions with end-users were conducted.

Two key features of media-players were incorporated into the design of DocPlayer: *play-lists* and *document properties* as database-fields. Thus, DocPlayer decouples the document collection from the file system by using a database to manage document information and provides users with a GUI to manipulate and explore different *views* of the document collection. Certain aspects of the media-player

model, however, do not translate well to the document domain. In a media player, a play-list of media files mean that the files are to be *played* – or, presented in a certain sequential order. Documents are not usually “played” in the same sense as media files, even though there are similarities between playing a music file and displaying a document page by page. Note that although displaying selected documents can be considered a generalization of *playing a file* in media players, this functionality was not implemented for the current version of DocPlayer.

System Overview

DocPlayer provides the means a) to import documents into the document collection, b) to manage the collection (both documents and Groups), and c) to open and view documents in the collection.

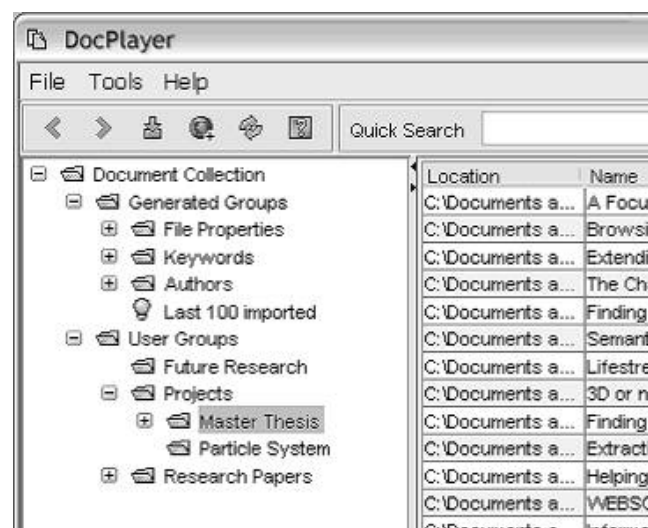


Figure 3. DocPlayer

The GUI consists of a) a view of system-generated Groups and user Groups to the right, b) a space for listing documents, c) the Quick Search box, and d) a toolbar.

Importing, Clearing, and Removing Documents.

Documents are imported into DocPlayer in a standard way. However, since users are largely manipulating *views*, “deleting” a document in a media-player is a little different from deleting a document from a file system. Users have two main options – a) clear a document from a Group, or b) remove the document from the collection altogether. Removing a document only removes it from DocPlayer; the current implementation does not support file-deletion from the file-system.

Managing Document-properties.

Currently, DocPlayer only implements some mechanisms for managing document-property fields – and does not allow users to add their own document properties. The property-set used in DocPlayer is {*name*, *location*, *import date*, *date of modification*, *keywords*, *author(s)*}. Property-value assignment is done by selecting a document or a batch of documents, choosing a property to edit in the right-click

menu, and then manually entering the information. When several documents have been selected, the entered value is applied to all documents. The user is also presented with the option to choose an existing value from the batch of selected documents and apply it to all selected documents.

Document Groups and Smart Groups. To reflect organizational features (rather than “a list of items to play”), the *play-list* concept is called *Groups* in DocPlayer. Groups can be named by the user and are essentially static sets of documents to which the user can manually add documents, similar to a play-list used in a media player. Any document from any view can be added to a Group – or to multiple Groups. In DocPlayer, Groups can be nested, creating a tree of Groups. Smart Groups are associated with a query and, in effect, form a virtual group that contains all documents that match that specific query. The content is dynamically updated according to the rule assigned; for instance, a Smart Group that uses a query for “PDF-documents that have a name that contains the string ‘visualization’”, will update itself when new PDF-documents having a name containing the word “visualization” are introduced to the document collection. Groups may contain Smart Groups – but Smart Groups cannot contain Groups or specific documents. In other words, no specific document can be added manually to a Smart Group. When selecting a Group, not only the documents contained by the selected Group are displayed; documents contained by sub-Groups and even sub-Smart Groups are displayed, and provide an overview of the total set of contained documents.

Automatically-Generated Groups. DocPlayer provides certain automatically generated Groups. Examples of these are ‘file type’ groups (containing all documents of a certain file type), and ‘keyword’ groups (containing all documents that have been assigned a certain keyword). For every file-type/keyword, a non-editable Smart Group is created which collects all documents of that specific file-type or keyword. Besides these file-type and keyword groups, automatically generated groups are also provided for each known author. Furthermore, the user also has access to a system group that lists the 100 most recently imported documents.

Group Management. Group management in DocPlayer is very straightforward using the standard operations copy, delete, and move. Both ordinary Groups and Smart Groups can be managed in this way. Due to restrictions in Java however, drag and drop was not implemented.

Quick Search. The Quick Search box implemented in DocPlayer enables users to perform database queries of different sorts.

DocPlayer Database. The database was designed to contain three types of items – documents, Groups and Smart Groups. In addition to properties associated with documents, such as name, location (in file system), keywords, last date of modification, and the like, the database also stores information about which documents and Groups are

contained by a certain Group, and what query is associated with which Smart Group.

User Response

Although database-based applications such as media-players and dynamically-generated Web sites are becoming more and more common, database-systems are still relatively difficult for typical users to understand. Therefore, the initial system and interface design of DocPlayer involved an iterated cycle of implementation, testing, and revision among the team members. Once the application was reasonably stable – and the basic functionality clearly represented in the interface – a number of informal meetings with non-specialists were arranged.

The main purpose of these meetings was to determine whether the DocPlayer metaphor was meaningful and helpful for people familiar with media-players – and to see which kinds of document-management tasks seemed natural and easy (and which did not). In general, the different participants had different combinations and degrees of experience, both with regard to media-player use and document-management. Participants were asked to answer a few questions about their experience with computers, with different kinds of document-management tasks, and with using media-players for different kinds of tasks (creating play-lists, for example). They were then given a short introduction to DocPlayer, asked to organize 25 documents, and then find certain ones (based on different criteria); in general, this activity was unassisted (except in the cases where there were technical difficulties).

Most of the participants found it fairly easy to start organizing files in DocPlayer. The one exception was a participant who had only used a media-player to play files (without creating play-lists and the like) – and who did very little in the way of file-management of computer-based files. Nonetheless, all participants enjoyed the activity – and all commented on the advantages of such document-management model. Although the more experienced participants made extensive comments about advantages of such a system over the usual file-system mechanisms for document-management, even the least experienced participant noted several explicit advantages. In particular, this participant like the ability to have documents organized into groups – and yet also quickly switch to a view in which all documents were visible at once.

Participants also indicated a number of limitations of the current implementation – and had several feature-requests. Several participants wanted a tighter integration between DocPlayer and the underlying file-system, some were interested in having additional document/group categories built-in, and there were some comments that indicated participants had a difficult time remembering all the keywords they themselves created. (Note that DocPlayer currently displays a list of keywords; the fact that this did not seem helpful raises issues of interface design that require further study.)

DISCUSSION

In general, the use of DocPlayer seems to aid in the tasks of filing and browsing a document collection, but new category-management issues are also introduced.

Importing, Clearing, and Removing Documents

There are many future improvements to DocPlayer that are obvious. In particular, there are many ways to make the importing of documents more efficient and to integrate the new documents into the system. Nonetheless, even though the user must currently manage much of the importing process manually, there is one significant benefit that should probably be considered even when automatic methods are possible. Namely, certain manual aspects of importing documents actually help the user know and trust that the DocPlayer collection is decoupled from the actual files in the file system

Filing Documents

The DocPlayer insights relevant to document-filing can be grouped into four categories: freedom of choice, reminders, filing-assistance, and multiple-workspaces.

Freedom of choice. In general, DocPlayer supports much more freedom than typical hierarchical file-management systems. This was obvious in the observations of, and discussions with, different test-participants. Everyone commented on the “sense of relief” at not having to worry about where something was filed – and how to find it later. Whether or not the media-player model turns out ultimately to be viable, it was fairly clear that a successful system should “relieve” users – not just practically, but psychologically.

Reminders. Surprisingly, DocPlayer turns out to be a useful system for *reminding oneself about future tasks*. For example, when doing research on one topic, it is common to discover interesting documents that are not directly relevant – or which could be relevant later. In DocPlayer it is simple to drag the same document into, say, “Current Paper” and “Interesting Articles about 3D Visualization.”

Filing-assistance. Smart Groups assist filing in many ways by automatically collecting documents that match a specified rule. For instance, the user might be collecting research papers and importing them into the document collection. A Smart Group can even be created that automatically collects papers – as they are imported -- that have properties related to particular keywords.

Multiple-workspaces. DocPlayer extends the notion of a “current workspace.” In conventional systems, a workspace takes on great significance, partly because there is such a cost to switch between different ones – and partly because the current mechanisms for “virtual workspaces” (to the extent they exist at all) typically require the manual creation and management of file-name aliases. In DocPlayer, it is simple to have many “current workspaces” – and not only switch between them, but to easily place common documents in each of them. Finally, “retiring” a DocPlayer

workspace becomes less of a requirement, and when it happens, it is much less of a burden – in particular, there is no dilemma about such things as “which of the documents in the current workspace should be retired with the workspace, and which should be moved to a new workspace.”

Filing limitations. Even though DocPlayer offers many ways to make filing an easier task for the user, the current model is not perfect. The approach used to organize documents into hierarchical groups in the media-player model is essentially *group centered*, which can be a limitation since some tasks require a *document centered* approach. This distinction – and the disadvantages – can be most clearly seen in the example of adding documents to Groups. When a document or collection of documents are considered to belong to more than one Group, users of DocPlayer have to add them to each Group separately; the more Groups, the more tedious the task becomes. We are currently considering a solution to this problem, but the point is that such issues tend not to arise in systems where multiple categorizations are not easy or possible.

It should be noted that since Groups of documents are hierarchically stored and organized, DocPlayer *does* have some of the limitations associated with hierarchical storage. For example, even though it is possible to add the same document to multiple Groups, it is not possible for the same Group (Group A) to be contained simultaneously by two Groups (Group B and Group C). Thus, duplicating Groups in DocPlayer presents some of the same versioning and management problems as duplicating documents in a hierarchical file system.

Locating documents

In DocPlayer it is possible to locate documents in ways not easy or possible using traditional hierarchical file systems. In addition to searching and querying, there are multiple browsing pathways and techniques. And although there are very complex combinations of mechanisms for finding documents, the initial feedback indicates that these seem fairly intuitive. In this brief space we mention one technique: *browsing by zooming*. In DocPlayer, selecting a Group not only displays the documents contained in the top Group, but recursively displays documents contained in sub-Groups (as well as those that match any rules specified by sub-Smart Groups). This behavior enables users to browse the document collection in a zooming fashion – by *reducing the document space* (rather than *presenting the user with a new document space*). Of course, in many ways this is like browsing through conventional directories and sub-directories; however, the addition of different queries and other constraints means that zooming-navigation in DocPlayer progressively narrows a *classification-space*, rather than a specific (file) *location-space*.

Browsing limitations. Some forms of browsing are not currently supported. Some of these will be incorporated into future versions, but in some cases the best way to incorporate these forms is not obvious. For example, it is

fairly straightforward to include the ability to move backward and forward through a history of browsed views. On the other hand, it would also be nice to include the equivalent of thumbnails for documents – that is, some kind of document previewing that lets users see “what the document is” at a glance; the resolution of this visualization challenge could greatly aid browsing.

Automating Property Extraction and Assignment

Similar to many media-players, DocPlayer supports batch property assignment; otherwise, the current implementation has only limited support for anything other than manual extraction and assignment of document properties.

During the early stages of development, we implemented one simple form of automatic property extraction. We were curious to see whether a user’s existing file-hierarchy organization might be used to automatically generate relevant keywords for imported documents. The user was given the option to use file-path information as keywords, i.e. if the document ‘author-title.pdf’ is located in d:\papers\visualization\chi03, the words ‘papers’, ‘visualization’ and ‘chi03’ would be used as keywords. Obviously, this mechanism only works as well as the organizational structure of the directories involved.

For certain media file-types, such as jpeg images or mp3 audio files, certain meta-data is accessible through a standard interface. However, for documents the situation is much less straight-forward; even though meta-data such as publication date, authors, number of pages, may be available in the textual content of the document, it can be difficult to extract such information – and in some cases, when this information exists as explicit meta-data, a proprietary file format may prevent access to such data.

Use of document properties

Having more document content information available allows for more sophisticated document management solutions. For example, if there is enough document meta-data, DocPlayer can help the user by automatically filing, as well as by supplying multiple paths to browse the document collection for a specific set of documents.

Properties vs. Groups

In some ways, *properties* and *Groups* have many things in common: they can and do serve as means for both the system and the user to classify documents. Groups, however, may allow more loose classification in the sense that documents can easily be put into any Group without presenting users with information that could constrain their filing strategies. The use of properties, on the other hand, presents users with the current categorization scheme, and can constrain them and force them to reflect on how the document has been previously classified.

CONCLUSION

The central focus of this research was to explore some of the consequences of extending the media-player model to

handle typical document-management tasks. The media-player model used in this context was found to have properties that can help users perform document management tasks in some ways that are more flexible than those found in a typical file-system. While DocPlayer has similarities to other document management systems, we believe that part of its appeal is the fact that the media-player control-metaphor is already familiar – and that this control-metaphor corresponds well with many document-management activities. Note that the very appeal of the model may also be partly a factor of limited ambitions: DocPlayer does not try to address all possible document-management tasks, and, indeed, the model may have very real limitations when the quantity of documents – and the complexity of the management tasks and features – is dramatically increased. In our work to date, DocPlayer has worked well with collections of documents in the hundreds; but of course, such a system probably needs to work at the scale of thousands (if not tens of thousands). More extensive implementation and testing is obviously required.

Directions for future research

In addition to involving typical end-users directly into future design and revision process, there are four main areas of future research.

Managing Document-properties. There are a number of mechanisms and features relevant to document-properties that we would like to incorporate into DocPlayer. In particular, it will be important to extend the system to handle different document formats (URLs, database files, and the like).

Inter-document Relationships. Typical media-players lack support for visualizing, browsing, or managing inter-document relationships; this is obviously an area of future interest. In particular, one of the major research issues involves the relationship between set-based and hierarchical representations. This tension is perhaps most clear if we consider the way play-lists are used for music – and the way the play-list model may be inappropriate for managing large document collections. For music, the relationship between songs in different play-lists rarely matters. On the other hand, document Groups often correspond to *projects* – and projects often stand in relationship to each other, whether in time, in priority, or in structure.

We can see this tension in the two diagrams below, where flexibility can also result in representational duplication in the interface. These issues are deeply problematic – and they are currently the focus of work in other domains, where, for example it is not clear how to reconcile the advantages of views (database-systems) with subclassing/super-classing and inheritance (object-oriented systems).

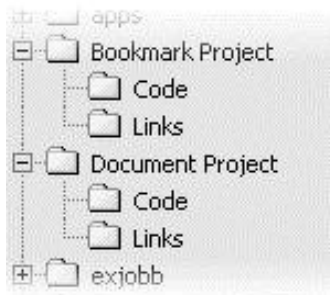


Figure 4. A traditional categorization hierarchy

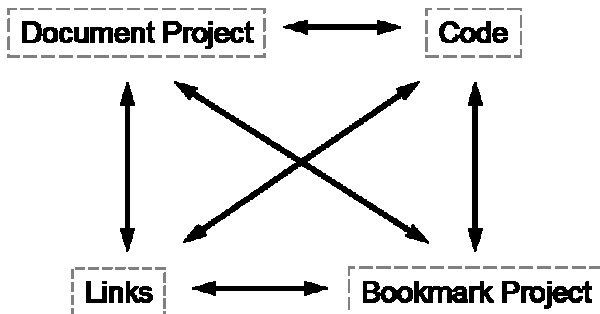


Figure 5. Hierarchy-less views

Integration of automatic meta-data collection. It will be useful to explore adding techniques such as clustering, self-organizing maps, and the automatic generation of meta-data (e.g., which words were used in the web-search that led the user to the document that was downloaded and imported into the document collection).

Document-reasoning. It may also increase the power and flexibility of DocPlayer if it included mechanisms for reasoning about semantic relationships between properties (e.g., an author is often associated with a specific institution). The system could include some predefined semantic ontology, or a user-built ontology, or a combination of the two.

Final thought: playing with document-management

We gave our system an initial “working name” of DocPlayer by analogy to its source of inspiration: media-players. Nonetheless, the experience of using the system to date suggests that the name may be appropriate for an unexpected reason. We have found that using the system seems to do more than empower users in the practical tasks of filing, managing, and retrieving documents; it also transforms the process into something more playful and enjoyable. There is much to be said for having “document-management” be more like “playing documents” than like “working to organize a desktop.”

REFERENCES

1. Ahlberg, C., Williamson, C., Shneiderman, B. 1992. Dynamic queries for information exploration: an

implementation and evaluation. *Proc. SIGCHI conference on Human factors in computing systems*, p.619-626

2. Barreau, D., Nardi, B. A. 1995. Finding and reminding: file organization from the desktop, *ACM SIGCHI Bulletin*, v. 27 n. 3, p. 38-43, July 1995.

3. Cockburn, A., McKenzie, B. 2001. 3D or not 3D? Evaluating the effect of the third dimension in a document management system. *Proc. SIGCHI conference on Human factors in computing systems*, p.434-441.

4. Dourish, P., Edwards W. K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry D. T., Thornton, J. 2000. Extending Document Management Systems with user-Specific Active Properties. *ACM Transactions on Information Systems*, Vol. 18, No. 2, April 2000, Pages 140-170.

5. Faichney, J., Gonzalez, R., 2001. Goldleaf hierarchical document browser, *Australian Computer Science Communications*, v.23 n.5, p.13-20, January-February 2001

6. Fertig, S., Freeman, E., Gelernter D. 1996. “Finding and Reminding” Reconsidered. *SIGCHI Bulletin Vol. 28, Numer 1, January 1996*.

7. Freeman, E. Gelernter, D. 1996. Lifestreams: A Storage model for Personal Data. *SIGMOD Record*, Vol. 25, No. 1, March 1996

8. Foo, J. 2004. DocPlayer: Design Insights from Applying the Non-Hierarchical Media-Player model to Document Management. MSc Thesis. Linköping University, Linköping, Sweden.

9. Gemmell, J., Bell G., Lueder, R., Drucker, S. Wong, C., 2002. MyLifeBits: fulfilling the Memex vision, *Proc. 10th ACM international conference on Multimedia*.

10. Hetzler, B., Miller, N. 1998. Four Critical Elements for Designing Information Exploration Systems. *Information Exploration workshop for ACM SIGCHI '98*

11. Honkela, T., Kaski, S., Lagus, K., Kohonen, T. 1997. WEBSOM – Self-Organizing Maps of Document Collections, *Proc. WSOM'97, Workshop on Self-Organizing Maps*.

12. Huynh, D., Karger, D. R., Quan, D. 2002. Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF. *Semantic Web Workshop, 2002*.

13. Jul, S., Furnas, G. W. 1997. Navigation in electronic worlds: *CHI 97 workshop. SIGCHI Bulletin 29, 4*.

14. Kobayashi, M., Takeda, K. 2000. Information retrieval on the web. *ACM Computing Surveys (CSUR)*, v.32 n.2, p.144-173, June 2000

15. Lamping, J., Rao, R., Pirolli, P., 1995. A focus+context technique based on hyperbolic geometry for visualizing

- large hierarchies, *Proc. SIGCHI conference on Human factors in computing systems*, p.401-408
16. Leung, Y. K., Apperley, M. D. 1994. A review and taxonomy of distortion-oriented presentation techniques, *ACM Transactions on Computer-Human Interaction (TOCHI)*, v.1 n.2, p.126-160, June 1994
 17. Leuski, A. 2001. Evaluating document clustering for interactive information retrieval. *Proc. 10th international conference on Information and knowledge management*.
 18. Mackinlay, J. D., Zellweger, P. T. 1995. Browsing vs. search: can we find a synergy? (panel session), *Conference companion on Human factors in computing systems*, p.179-180.
 19. Malone, T. W. 1983. How do People Organize Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Information Systems (TOIS)*, v. 1 n. 1, p 99-112, Jan 1983.
 20. Mander, R., Salmon, G., Wong, Y.Y. 1992. A 'Pile' Metaphor for Supporting Casual Organization of Information. *Proc. SIGCHI conference on Human Factors in computing systems*, p. 627-634.
 21. Noy, N. F., Hafner, C. D. 1997. The State of the Art in Ontology Design: A Survey and Comparative Review. *AI Magazine, Fall 1997*, p. 53-74
 22. O'Toole, J. W., Gifford, D. K.. 1992. Names should mean what, not where, *Proc. 5th workshop on ACM SIGOPS European workshop: Models and paradigms for distributed systems structuring*.
 23. Robertson, G., Mackinlay, J. D., Card, S. K. 1991. Cone Trees: animated 3D visualizations of hierarchical information, *Proc. SIGCHI conference on Human factors in computing systems: Reaching through technology*, p.189-194.
 24. Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D., van Dantzich, M. 1998. Data Mountain: Using Spatial Memory for Document Management. *Proc. UIST '98*.
 25. Soules, C. A. N., Ganger, G. R. 2003. Why can't I find my files? - New methods for automating attribute assignment. *Proc. HotOS IX: The 9th Workshop on Hot Topics in Operating Systems*.
 26. Wong, S. K. M., Raghavan, V. V. 1984. Vector space model of information retrieval: a reevaluation. *Proc. 7th annual international ACM SIGIR conference on Research and development in information retrieval*, p.167-185.