

Providing task-oriented document management in pre-existing applications through the use of metadata and historical information in file chooser dialogs

Felipe Erias Morandeira

Background

Interactive systems have used different strategies to tackle the problem of document management. The most traditional approach has been to use the office desktop metaphor, structuring documents in a file hierarchy. This forces the users to deal with an imposed complex structure, the file system, that might not be directly related to the task that they are actually trying to achieve with the system.

In order to reduce this problem, there are systems that try to integrate metadata and historical information in the desktop and the file manager, allowing the user to search for the document that he is looking for. However, this has the problem that the domain that the user can search is by default the whole system, instead of being restricted to the specific task being performed.

In other cases, applications (e.g. music players, photo managers) might decide to implement their own user interfaces to browse documents in a way that is tailored to the task at hand. A special case of this are systems where applications are full-screen and one at a time, effectively turning the device into a temporary information appliance. This approach usually requires the application to develop its own user interface for finding and opening documents, and even its own metadata storage. In some cases (e.g. iPhone OS), the applications can not even share documents. There has been little effort towards having a common solution that could be used across applications so that they share the same metadata backend and a consistent interface for browsing and searching.

Proposed solution

This project would describe and design a way of accessing documents that is adapted to the activity that the user is performing, allowing for different search strategies depending on the concrete kind of documents that the application wants to manipulate. This solution would use a common metadata storage and would be consistent between applications.

In order to achieve this, we would take advantage of the fact that most applications already use a file chooser dialog for opening documents. As this dialog is provided by a shared library, this means a single point of change to alter how the application manages its files from the user's point of view. A new document chooser dialog would substitute the existing one, but instead of just offering a way to browse the filesystem hierarchy, the new dialog would use metadata and historical information to provide search strategies that would be tailored to the specific activity that the user is performing.

Possible scenario #1: the user opens his text processor and gets a list of the last documents that he edited with it; as the desired one is not on the list, he performs a text search to find the document that he wants to open.

Possible scenario #2: using an image manipulation application, the user locates a foto by searching on its EXIF data for the date when it was taken.

Possible scenario #3: from a music player, the user performs a search on music from a certain group and plays a song. Willing to use one it for his own mix, he opens a sound mixer application and searches for the song that he wanted to use: the open dialog is consistent with the one used by the music player, so the same strategy to find the song can be used again.

Project goals

This project would produce a tentative design for an document chooser dialog that hides away the complexities of the file hierarchy and instead offers an interface that is adapted to the specific kinds of files that the application is intended to manipulate. Ideally, the application should need to provide as little information as possible (maybe just the file types) and the dialog would contain the appropriate metadata fields. Those search fields would be taken from the system-wide ontologies of the metadata storage system, so they would be consistent among applications manipulating the same kinds of documents.

Although a different problem, the process of creating a file could be similarly simplified, maybe requiring the user to just give a name and (optionally) some tags.

The ideal outcome for this project would be an experimental confirmation that this is indeed a good approach, along with design ideas that could be used in a realistic solution. This real library should be able to handle the selection of search fields in an automatic way depending on the kind of files that the application wants to use. As that would obviously be too ambitious for a student project, the prototypes would be tailored for specific applications.

Prototyping

We would begin with low fidelity prototypes for some of the most common cases (e.g. images, text documents or music). The search strategy for each of these cases could be based on the studies described in the literature.

The experimental argument is that this solution would allow people to remain focused on the task at hand while being more effective at finding and using information than existing solutions. In order to test this, we would develop one or more high-fidelity prototypes that could be used in experimental testing.

This implementation would be based in GNOME technologies, which already provide metadata and historical data storages. As the standard file chooser dialog is already a part of the GTK+ library, it should be easy to alter an existing application to use a different one.

A possible experimental evaluation could involve implementing a document chooser dialog for a text editor, and then comparing the performance of both the standard and the new solutions when participants have to carry out tasks that involve reading and browsing through many documents.

Another possible evaluation could be to implement a document chooser for an image viewing application, and then have participants find images with a common characteristic from a big set.

The evaluation could measure the difference in performance between the standard dialog and the new one, and it could also use questionnaires or other methods to capture the participant's subjective reaction.

References

- Tracker - GNOME's metadata storage and search engine
 - <http://projects.gnome.org/tracker>
- Zeitgeist - GNOME's backend for user activity tracking
 - <http://seilo.geekyogre.com>
 - <http://live.gnome.org/Zeitgeist>
- The ACM database contains multiple accounts of different search strategies focused on different kinds of information, including text, images and music.