Problems with Save

Sari A. Laakso
Interacta Design Oy
Vuorimiehenkatu 23 B
FI-00140 Helsinki,
FINLAND
+358 9 6813 8520
sari@interacta.fi

Karri-Pekka Laakso
Dept. of Computer Science
University of Helsinki
P.O.Box 26 (Teollisuusk. 23)
FI-00014 University of
Helsinki, FINLAND
+358 9 191 44268
kpalaaks@cs.helsinki.fi

Panu Vartiainen, Asko Saura
Interacta Design Oy
Vuorimiehenkatu 23 B
FI-00140 Helsinki,
FINLAND
+358 9 6813 8520
{panu,asko}@interacta.fi

ABSTRACT
Saving documents, i.e. moving data manually between main memory and disk storage, is a difficult concept for novice users and causes unnecessary work and data loss both for novices and experienced users. Use scenarios show that the problem cannot be solved simply by re-designing the save feature or adding an autosave, because the save problem is entangled in a broader complex of document management problems.

Based on the analysis of use scenarios, we have designed and implemented a prototype that solves a set of the most essential problems relating to the save problem in the context of word processing.

Keywords
GUI design, goal-based design, save problem

INTRODUCTION
Even experienced users often lose data because they have not saved their documents often enough. Programs crash, data connections break down, power outages occur, or users simply select the wrong option when exiting the program [1]. To avoid this, experienced users learn to save continuously, and although saving gradually becomes nearly an automatic cognitive process for them, it remains an error-prone unnecessary burden. Novice users have trouble even with the concept of saving. They cannot grasp the idea of having one copy in disk storage, another in the main memory, and the need to synchronize these two. The user interface based on the implementation model causes a lot of trouble [2].

The problems related to saving have been partially solved in programs that offer support for autosave, undo, and versioning, for example, but interaction design of the features is usually so weak that users cannot find and use them. In addition, some useful features are missing, e.g. support for renaming documents. Currently, users are forced to rename documents with the Save As function which creates a new copy of the document as a side effect.

We have taken the first step towards a more complete solution by trying to solve the problems related to the concept of saving, opening, and closing documents in the context of a word processor. We have based our solution on frequently occurring users' goals that have been extracted from use scenarios.

USE SCENARIOS AND THE DESIGN PROCESS
To find out users' goals, we traced the evolution of documentation in four projects in our company during six weeks. From the scenarios, we extracted frequently occurring patterns related to the save problem. We found out that the most recurring patterns included using a prior document from a very similar project as the basis of a new one (locate the document, make a copy), copying material from earlier documents (locate a document, copy some text, paste it into the new one), and creating a new version of a document (locate the current version, edit it, save a new version without touching the previous one).

In the last case, the user's natural work flow does not begin with creating a new version of the document (Save As or New Version) – she wants to make changes instantly. The user might have received a phone call from a customer who wants to make minor changes to the contract, and the user is fully focused on the contents of the contract. If she hits the save key during editing (an automatic process), it will be very difficult to restore the previous version which might be the last official version of the document, for example. If she does not save the document during editing, she will probably some day lose her changes.

Interaction design was based on the use scenarios mentioned above, and some ideas originally presented by Cooper [1]. We designed the user interface and created paper mock-ups which were evaluated by walking through use scenarios in three iterative design phases. In parallel with design and evaluation iterations, we implemented a prototype with Microsoft Visual C++ in order to confirm that the implementation was possible on top of the existing Windows 98 (MS-DOS) file system.

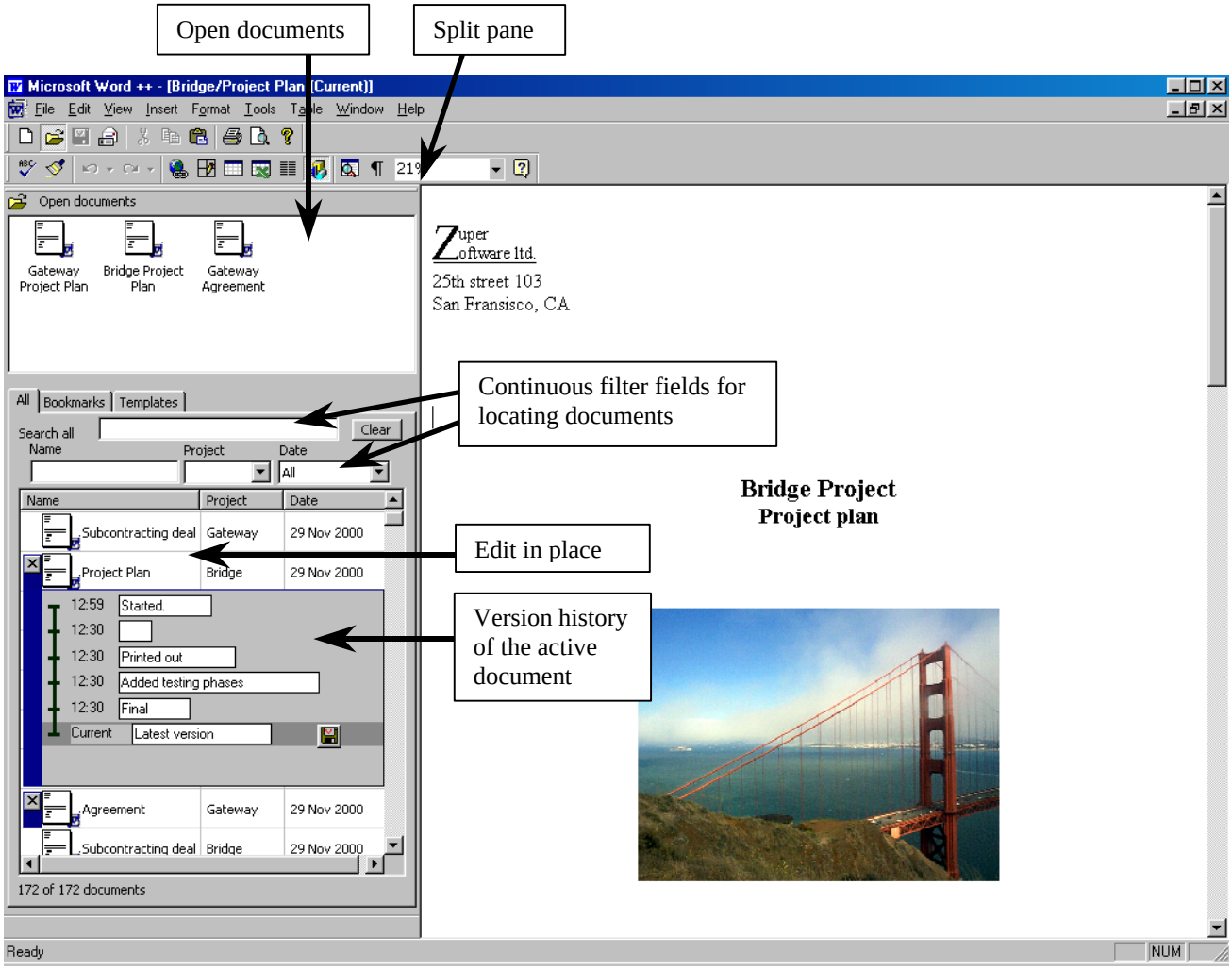

Figure 1. A prototype of our save problem solution for Microsoft Word.

PROTOTYPE

Our current prototype (Fig. 1) writes the user's changes continuously to the disk, i.e. the data between the main memory and the disk is synchronized. Users do not have to face the implementation model, and when closing a document, they never get the error-prone "Do you want to save changes?" dialog box that breaks their work flow.

Because most of the scenarios begin with locating earlier documents, the standard File Open dialog box has been replaced by a more usable design that offers a continuous filter for the documents that can be edited by this program (see dynamic queries in [3]). When the user starts typing the name of the document or the project, the system dynamically filters the document list.

When the user has found the document he is looking for, he opens it by selecting the row. The system opens the document on the right side of the window, and an icon of the document is shown in the open documents area (top-left). Open documents are always visible, and the user can easily switch between them while cutting and pasting material, for example. To rename a document, the user simply clicks and edits the name on the document list.

In the prototype, the version history of Microsoft Word has been replaced by interaction based on direct manipulation. The currently visible document is expanded under the corresponding row in the document list on the left side. When the user wants to return to a previous version, he simply clicks on the row of the version. If he edits the previous version, a new vertical branch of the version tree will be created (not shown in Fig. 1).

FUTURE WORK

Although the design decisions for the most recurring parts of the use scenarios can be demonstrated with the current prototype, the prototype does not yet fulfill all the requirements of user interface specification, and its appearance is not finished. These shortcomings will be fixed before the next walkthrough sessions.

REFERENCES

1. Cooper, A. About Face. The Essentials of User Interface Design. IDG Books Worldwide, USA, 1995.
2. Norman, D.A. The Psychology of Everyday Things. Basic Books, New York, 1988.
3. Shneiderman, B. Dynamic queries for visual information seeking. IEEE Software, Vol. 11, 70-77.