Freefeed-Instead of Forms

Youssef Ali, Lars Hallnäs, Mats Jontell*, Nader Nazari & Olof Torgersson
Department of Computing Science
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
+46 31 772 54 06
{youssef, lars, nazari, oloft}@cs.chalmers.se
*Clinic of Oral Medicine, Faculty of Odontology, Göteborg University

1. INTRODUCTION

The most common way to design an application where data needs to be entered is to use forms. The forms are typically built from objects such as text-fields, pull-down lists, and checkboxes (Frank 1988). This paper describes Freefeed; a technique for entering data where the forms are replaced by a specialized text-editor coupled with hypertext links for navigation and easily scrollable text lists containing possible values.

Freefeed was originally developed as a solution for entering detailed patient, medical history, and status information during clinical examinations.

The design goal behind Freefeed was to create an unobtrusive, easy-to-use, space efficient, and scalable method for entering data, where the forms used could be created by users without requiring any programming knowledge. We describe the interaction technique and experiences from using it regularly for about two years.

2. DESIGN CONSTRAINTS

Freefeed was conceived as a solution for entering data based on an analysis of the constraints given. This analysis describes a conceptual model of the act of entering data and external requirements describing the environment in which data is to be entered.

2.1 Entering Data

The conceptual database model for which Freefeed was developed is that of a collection of definitions, where each definition describes one record. Each such definition can be pictured as a collection of equations:

```
Occup = Dentist.
Born = Sweden.
```

In this setting, entering data is the act of creating a definition. Our goal was to support the act of defining medical examinations, while keeping a non-technical interface to the user.

2.2 Other Requirements

Some of the more important external requirements were:

- Data is entered by the clinician him/herself while a patient is being examined.
- Each record in the database can have a large number of different attributes and each attribute a very large number of possible values.
- Values for attributes are most often taken from formalized lists of valid values.

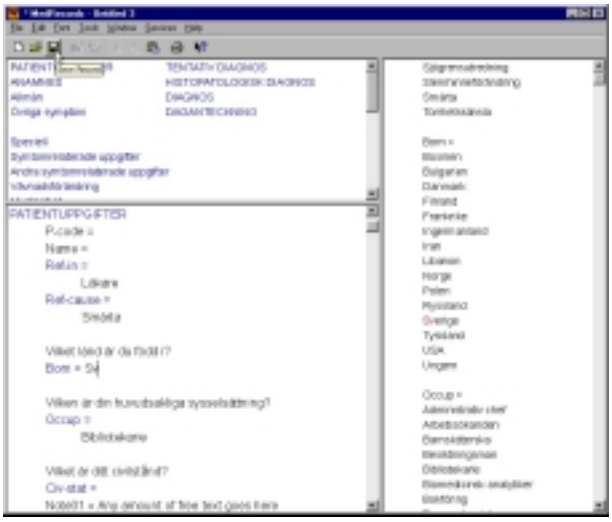

Figure 1: Freefeed main window.

3. DESIGN

Freefeed is designed to display partial definitions and to provide efficient techniques for completing them.

The user interface consists of one window divided into three views as shown in Fig. 1. At the top left is a navigation area, below it is the input view where data is entered, and to the right a list of commonly used values. All work is performed within this single window.

The contents of each view is taken from template files in Rich Text Format. Thus, different forms can be used without modification to the program. Furthermore, the layout of each view can be designed using all common formatting attributes wrt font, colors, tabbing, etc.

The interaction paradigm is based on a small number of common operations found in many applications. The input view works as a specialized text-editor. It displays an incomplete definition, which is edited when data is entered. This "form" contains arbitrary lead texts and a number of database attributes, each followed by an equals sign. The equals sign marks the beginning of an implicit input textfield where values are entered. Only these implicit input fields may be edited by users. All other parts of the text are fixed.

The user navigates within the input view by tabbing between the different attributes, scrolling, using standard navigation keys, or by following the links in the navigation view. The navigation view typically displays links into all the main subsections of the input view. Clicking a link moves focus to the corresponding area of the input view. Values may be entered in several ways. First by typing the value. As a value is being typed, the first matching value in the list to the right is highlighted. Pressing the completion key or clicking the value inserts it into the form. Second, following a link from an attribute to its value list to the right and clicking the desired value inserts it into the input view. External documents (e.g. images) are included by dropping them on the input view. Thus, Freefeed is based on a simple flow of actions from navigation view, to input view, to value list view and back.

4. DISCUSSION

Freefeed has evolved through a continuous interaction between users and developers. The version described here is the third iteration and is used daily at several clinics.

Data from more than 1200 examinations have been entered using Freefeed. All this data has been entered by the clinician while examining a patient. The interaction paradigm works very well. Current forms consist of some 100 attributes and a large number of values, e.g., lists of different drugs and diseases. The navigation tools are sufficient although some fine-tuning of the systems scrolling behavior is called for.

Compared to traditional form-based interfaces we believe that Freefeed scales very well. Having several hundred different readily available attributes in one screen poses no problem. Displaying forms for the same amount of attributes would require navigating between many different screens, typically in some fixed order.

5. CONCLUSIONS AND FUTURE WORK

We have presented an alternative interaction technique for entering data in an efficient manner. The technique uses well-known components such as keyboard, mouse, hypertext links, drag & drop, and ordinary text editing and combines them in a manner that has been tested and proven useful in a real-world situation.

So far, we have not tested enough other domains to correctly judge where Freefeed is best applied. Testing the system on a large number of different kinds of forms is an area for future work. Another is incorporating other interaction methods. Naturally, there are situations where a graphical interface works better. Adding a Plug-in architecture to account for this should be trivial.

6. REFERENCES

Frank, M.R. and Szekely, P. (1988) Adaptive Forms: An Interaction Paradigm for Entering Structured Data, in *Proc of the 1998 International Conference on Intelligent user Interfaces*, San Francisco, CA USA. ACM, 1998.

