Ergonomics of Hybrid Automated Systems III
P. Brödner and W. Karwowski (Editors)
Elsevier Science Publishers B.V.

Designing and testing the usability of a relational data base system with end-user oriented benchmark tasks

M. Rauterberg

Work and Organizational Psychology Unit, Swiss Federal Institute of Technology (ETH), Nelkenstr. 11, CH-8092 Zürich, Switzerland

Abstract

The results of a survey among a group of users of a Data-Base-Management-System (DBMS) that runs on personal computers and the results of several experimental studies indicate that specific support is necessary for this group of users in the form of the interface concept of active joins. An active join, presented in a list on the interface, enables end-users to manipulate data records by direct manipulation. The definition of an active join, the generation of active joins by direct manipulation, and the usage of active joins are introduced herein. An experimental comparison between the desktop interface with and without active joins demonstrates the superiority of the concept of active joins.

1. INTRODUCTION

The customer of a data base system on a PC has several complex problems to solve: (a) he must define the data structure of the data base, (b) he must implement this definition, and (c) he must operate the data base system. These problems must be solved without specific knowledge about data base theory and without specific support through a human data base administrator. The end-user is responsible for both the entity integrity and the referential integrity her/himself.

There are three different views of the data structure of a data base: the *conceptual view* of all entities, the *internal view* of the physical organisation of the data records and the end-user's *external view* of parts of the data base [1, p.26]. A PC end-user must be supported with specific interface features in all of these three views to provide a user oriented [2] and a task oriented [3] software design. In most task contexts, end-users need a task specific combination of several attributes stored in different files. These combinations are commonly called joins. The possibility of direct manipulative operations on attributes combined in joins is an example of task oriented interface design [9,10,11].

The administrator's view of the data encloses the conceptual view and the internal view, and is dominated by technical constraints: a minimum of redundancy and a maximum of consistency [4]. These conditions can be realized in a relational DBMS by defining the appropriate relations. A simple example is a wholesale dealer who manages his orders with a relational DBMS. The exemplary data base is composed of three files: the article file, the supplier file and the supply file. These three files contain all relevant attributes to make an order, so the dealer needs a specific join.

The attributes that make up such a join can be handled in two different presentation modes, the list mode or the mask mode. In the mask mode, the user has full access to all attributes at once. For an overview of all data records in the virtual join file, users prefer the list mode. If an end-user is interested in a table, or list, defined by a (passive) join, he has to create a correct join definition. In the desktop version of ADIMENS GT the merge document with the retrieval commands to define the above mentioned join 'order' looks as follows:

```
#QUANTITY#      #SUPPLIER-NO->SUPPLIER# #SUPPLIER-NAME# #CITY# #<-#
                #ARTICLE-NO->ARTICLE# #ARTICLE-NAME# #TYPE# #<-#
```

The user has to merge this document with the file 'supply' which contains the both linking keys "supplier-no" and "article-no". In the specific task context of 'making an order' the wholesale dealer is not interested in knowing in which files the relevant attributes are stored. He needs all relevant attributes at once in one dialogue context. If the wholesale dealer wants to update one of the attributes, e.g. the attribute 'quantity', he then needs active access to this attribute in the context of all associated attributes.

2. USABILITY OF ADIMENS GT

In the desktop interface of ADIMENS GT the data definition language (DDL) and the data manipulation language (DML) are partially realized with a direct manipulative dialogue structure. The data records of a file (basis relation) can be listed in a window (list mode) or in a mask for each record (mask mode). Each row of the list in the list mode represents a single data record and can be handled with direct manipulative operations (e.g. copy, delete, modify, export, print, etc.). The user has two different possibilities to find a single data record or a set of data records: the temporary *selection* operating on only one key attribute, or, the definition of a *filter* with logical conditions operating on any attributes. The selection and the filter are only definable for one data base file at once. To define more complex queries, the end-user has to use the retrieval language AdiTalk, which is similar to dBase.

Since the development of the desktop interface for the relational data base management system ADIMENS 40,000 installations of this product have been sold. A survey was carried out by publishing a questionnaire in the newsletter of the ADIMENS user group. The results of 218 questionnaires completed by end-users show that the average data base structure (of 153 reported data bases) consists of six files (basis relations), with one or two link keys among these files. These empirical facts indicate, that most of the end-users, who reported on own data base structures, are able to solve (a) the definition problem as well as (b) the implementation problem. Nevertheless, 10% of the above mentioned data bases have no link keys among data base files. This indicates that this group of users needs special support. The complexity of the average data base structure suggests the need of joins by the end-users. Because there was no item in the questionnaire about the frequency of the use of joins, it is unclear how many joins were defined and in use.

Different empirical studies were conducted to analyse the usability and efficiency of several aspects of ADIMENS GT [3,5,6]. In an experimental investigation six qualified experts (beta tester, etc.) were asked to define a join using a simple retrieval language (the syntax of the retrieval language is given in the Appendix of [7] and [12]). One important interactive problem was observed: none of the investigated experts was able to define a join between two or more files without help from the investigator. This implies that in the context of a desktop interface, end-users refuse to switch over to command language mode. The generation of both tables, and lists, only with attributes of different basis relations (data base files) is done with a simple retrieval language. These tables, or lists, are only presented as output text in view, or browse, mode. The user has no possibility to operate on table entries in a direct manipulative fashion. The total number of keystrokes to solve ten tasks of all experts was 7311. The mask mode was used by the experts for only 21 keystrokes, indicating that they prefer the *list mode*. Since there was no possibility for a temporary selection of data records in the list mode, the user had to define a global filter.

Also, in ADIMENS GT all direct manipulative operations on data are possible only for data records of a single file, so that the basic idea of a relational DBMS could not be conceptualized adequately to support the end-user's mental model. To solve these interactive problems, a new version has been developed and implemented. The new ADIMENS GT+ can be characterized by the following features: the option of temporary selections in combination with the list mode and the option of definitions of virtual join files. The concept of virtual join files is based on equi-joins. In ADIMENS GT users have only the possibility to generate lists in a browse mode. This implies the necessity of *active joins*.

3. THE CONCEPT OF AN "ACTIVE" JOIN

The idea of the concept of "active" joins is based on the psychological theory of task orientation and control [3]. Two prerequisites for the development of task orientation are: "(a) the individual should have control over the materials and processes of the task; and (b) the structural characteristics of the task (should) be such as to induce forces on the individual toward aiding its completion or continuation" [8; p. 53]. Further on the two aspects of control are important: (1) "all those elements in which choice of how to do a job was left to the person doing it" and (2) "the extent to which an individual is free from intervention in the form of inspection and supervisory check-up" [8; p. 54]. The concept of the complete task is discussed in detail elsewhere [2,3,10].

Let us have a look towards our example. The simple task 'making an order' is characterised by the following feature: the dealer needs all information stored in three different files at once in one dialogue context to have task oriented control of his actions. He is really not interested in the informations stored in attributes like 'article-no' and 'supplier-no'. This kind of attributes is primary caused solely by technical reasons (the administrator's view!). An optimal task orientation would be given, if the DBMS could automatically manage attributes like 'article-no', etc. The end-user only needs to specify the different types of relations, which are relevant in the specific task context. Further on, a lot of task oriented research must be done to come to the right semantic of active joins, which can be implemented in DBMSs. A first step in this direction is described below.

3.1. The definition of an "active" join

In ADIMENS GT, the user could only take a look at the contents of attributes defined by joins using a simple retrieval language. The user had no possibility to change the attribute values in this passive browse mode. For this reason, this type of join is called a *passive join*. In contrast an *active join* is defined as follows: the user has full access to all attribute values displayed on the output screen. This condition can be fulfilled in the context of a desktop interface in two different ways: 1. the attribute values can be directly manipulated in the output tables, or lists, or 2. the attribute values can be manipulated after transfer to a dialogue box. The second way is implemented in ADIMENS GT+.

In the 'list mode' of ADIMENS GT+ version, the user has full access to all virtual data records of a join file. Each virtual data record is represented as a row in a table, or list. The complete list is mapped onto a desktop window. The list is sorted by a selected key attribute, shown in the window header. The following direct manipulative operations can be applied to all virtual data records represented in a window: delete, print, export, multi-level sort, etc. In order to change the attribute values, the user must first of all transfer the row, or virtual data record, to a dialogue box. This operation can be done by activating all rows of interest and then starting the function 'modify'.

Now we will consider the direct manipulative method to define and create a data base with active joins. In ADIMENS GT+ a data base can consist of three different file types and eight different data types. One of these three file types is the join-file type. To create an active join the user has to change to the definition mode of ADIMENS GT+. In the definition mode he copies all required objects (data file, join file, link file, and different types of attributes) from the part window (pw) into the working window and arranges them appropriately. There are two different types of working windows: the working window for the data, join and link files (wwf), and the working window for the different data types (wwd). An empty working window for the files (wwf) is automatically given.

After creating a file object in the wwf, the working window of the data types (wwd) can be activated by double click on the file icon in the wwf. Now, the user can create and arrange the different attributes of the active file by copying from the collection window into the wwd. The length of each data type is defined by a default value, which can be changed by the user immediately after creation. The user can create and arrange all objects with direct manipulation. After defining all necessary attributes of all files the user needs only to activate the pull down menu

item "DB generation", and the DBMS sets up the constructed data base automatically. Once a join file is created including all attributes of only one or of several dependent data files, specific joins can be produced by defining projections. A projection is a selection of those attributes which are of special interest. The definition of a projection can be done interactively by pointing with the mouse to the required attributes. This dialogue mode can be activated by clicking on the menu item 'attribute selection' in the pull down menu 'switch'.

3.2. The operations on "active" joins

At any time during operating a data base the end-user can define or modify active joins. This is no problem, because each definition of a join is based on attributes of data files. Only the data files (basis relations) define the structure of the data base. This feature of join files is the reason why we call join files 'virtual data files'. Unsolved problems are the semantics of the functions 'enter' and 'delete'. To protect end-users from unnecessary loss of data, a sophisticated protection mechanism was implemented. Each data file can have the following four protection modes: enter, modify, delete, view. Each attribute of a data file can be protected from unallowed reading, writing, and modifying, too. These different protection modes of attributes are necessary, to increase the safety of join operations. For example, it is dangerous to modify the value of the attribute 'article-no' or 'supplier-no' in the task 'making an order'.

The current implemented version of an active join is not powerful enough to allow end-users a semantic correct delete or enter operation. So, in all of these cases the end-user has to operate directly on the data files. Another restriction is the fact, that each active join is an equi-join. If an end-user is interested in other conditions of comparison, he must use the retrieval language AdiTalk. To test the usability of our concept of active joins a user oriented benchmark test was done.

4. THE USER ORIENTED BENCHMARK TEST OF THE "ACTIVE" JOIN CONCEPT

In an experimental investigation, the two different desktop interfaces (ADIMENS GT and ADIMENS GT+) were compared. Two samples of students in computer sciences ('group-1': n = 16, and 'group-2': n = 15) tested *each* interface version of the DBMS ADIMENS. The first group started with the old version (GT) and then tested the new one (GT+), the second group tested in reversed sequence (GT+ -> GT). The samples of users can be described as follows:

Group-1 (GT->GT+): average age 24 (± 2) years; 14 men, 2 women; 6. (± 2) semester computer sciences; 4138 (± 4022; range: 450-15.745) hrs. general experience in Electronic Data Processing (EDP), including 338 (± 515; range: 10-1.500) hrs. previous experience with DBM-systems [other than ADIMENS], and 3 (± 9; range: 0-35) hrs. with ADIMENS.

Group-2 (GT+->GT): average age 24 (± 3) years; 14 men, 1 woman; 5. (± 2) semester computer sciences; 3706 (± 4562; range: 705-15.535) hrs. general experience in EDP, including 67 (± 119; range: 0-350) hrs. previous experience with DBM-systems [other than ADIMENS], and 7 (± 21; range: 0-75) hrs. with ADIMENS.

There are no statistically relevant differences between group-1 and group-2, except the different amount of DBMS experiences (t-test, two tailed, df=30, $p \leq 0.05$). All users had to complete the following user benchmark task given in two parallel forms:

Task description for the interface without 'active joins' (ADIMENS GT):

"Create a list using the given retrieval language. This list should contain all values of the attribute 'quantity' of the file 'supply' and of the attribute 'article-name' of the file 'article'. Produce a adequate merge document with the text editor, merge the document with the file 'supply', and print the result."

Task description for the interface with 'active joins' (ADIMENS GT+):

"Create a list using the join file 'order'. This list should contain all values of the attribute 'quantity' and of the attribute 'article-name'. Define a adequate projection, and print the result."

The dependent variables were the time consumed ("test processing time" = TPT) and the number of keystrokes used ("number of keystrokes" = NOK) to complete the benchmark task. Each pressing of a key (or a mouse click) and a timestamp belonging to this keystroke was automatically recorded in a logfile. The investigator made sure that each test was completed.

Table 1:
The results of the dependent variable "test processing time" = TPT.

TPT	User Group-1	User Group-2
1. task	[GT]: 709 ±321 s	[GT+]: 282 ±186 s
2. task	[GT+]: 209 ±169 s	[GT]: 542 ±288 s

Table 2:
The results of the dependent variable "number of keystrokes" = NOK.

NOK	User Group-1	User Group-2
1. task	[GT]: 154 ±38 keys	[GT+]: 25 ±20 keys
2. task	[GT+]: 32 ±19 keys	[GT]: 145 ±54 keys

A variance analysis (ANOVA) with the two factors "interface [GT vs. GT+]" ($F_{1,30} = 49.2$, $p \leq 0.001$), and "sequence [group-1 vs. group-2]" ($F_{1,30} = 0.43$, $p \leq 0.516$) was computed. The main effect of the "interface" is significant (see Table 3, second column). A ANOVA including the number of keystrokes as a covariate with the same two factors "interface" ($F_{1,30} = .001$, $p \leq 0.961$), and "sequence" ($F_{1,30} = 0.43$, $p \leq 0.516$) was also computed (see Table 3, third column). The significant interaction between the factor "interface" and the factor "sequence" indicates, that the decrease of the test processing time can neither be explained exclusively by learning ($F_{1,30} = 4.87$, $p \leq 0.035$), nor by the amount of interactivity in combination with learning ($F_{1,30} = 3.88$, $p \leq 0.059$). May be, this significant interaction is a result of the different amount of DBMS experiences between both groups.

Table 3
Results of the two factorial variance analysis with the factor "interface" and the factor "sequence" for the dependent variable "test processing time" (variable TPT). The variable "number of keystrokes" (variable NOK) is used as a covariate in a second variance analysis.

dependent variable:	covariate:			covariate:		
Test Processing Time (TPT)	without			"number of keystrokes"		
source of variation	d f	F	p	d f	F	p
"interface" [GT vs GT+]	1	49.24	.001	1	.001	.961
"sequence" [group-1 vs group-2]	1	.43	.516	1	.43	.516
"interface"⊗"sequence" interaction	1	4.87	.035	1	3.88	.059

The test processing time working with the new interface (GT+; 244 s ± 178 s) was 39% of the test processing time working with the old interface (GT; 628 s ± 312 s). This decrease of test processing time can be completely explained by the different amount of "interactivity" measured by the variable "number of keystrokes". The users completed the benchmark task with the new GT+ interface much quicker than with the old GT interface (see Table 4).

Table 4:
The mean of the dependent variable "test processing time" = TPT for the interface GT (without active joins) and GT+ (with active joins).

TPT	[GT] interface	[GT+] interface
global mean	628 ± 312 s	244 ± 178 s

5. CONCLUSIONS

The dominance of technical aspects in conventional interface design led to insufficient usability of DBMS´s by end-users. Taking the user's task oriented view seriously leads to a useful interface concept: the active join [see also 12]. This interface concept was implemented in a relational DBMS, tested with a user oriented benchmark test against the old interface without the active join function. The empirical results show a significant superiority of the interface with the active join concept. Active joins can be implemented with other systems like Oracle-SQL/Forms, too. The superiority of the interface concept of active joins can be explained by the decrease of necessary number of keystrokes to solve the benchmark task. The more keystrokes a user needs to solve a task, the more possibilities the user has to make a mistake. The direct manipulative handling of active joins makes it possible for end-users to define their individual task oriented DBMS interface objects and their task oriented view of the data. There are still some unsolved problems: the task oriented semantic of the join operations 'enter' and 'delete'. Taking the user's view seriously will help solving these problems in the future.

6. REFERENCES

1 G. Schlageter & W. Stucky, Datenbanksysteme - Konzepte und Modelle. Teubner 1983.
2 E. Ulich, Some aspects of user-oriented dialogue design. North-Holland, (System Design for Human Development and Productivity: Participation and Beyond; Fuchs-Kittowski K., Docherty P., Kolm P. & Mathiassen L., eds.) 1987.
3 E. Ulich, M. Rauterberg, T. Moll, T. Greutmann, O. Strohm, Task Orientation and User-Oriented Dialog Design. International Journal of Human Computer Interaction, vol. 3, no. 2,.(1991) 117-144.
4 C.J. Date, An introduction to database systems, vol. 1&2. Addison-Wesley 1983.
5 M. Rauterberg, MAUS versus FUNKTIONSTASTE: ein empirischer Vergleich einer desktop- mit einer ascii-orientierten Benutzungsoberfläche. Teubner, (Proceedings of "Software-Ergonomie ´89"; Maass S. & Oberquelle H., eds) (1989) 313-323.
6 M. Rauterberg, Experimentelle Untersuchungen zur Gestaltung der Benutzungsoberfläche eines relationalen Datenbanksystems. Swiss Federal Institute of Technology - Work and Organizational Psychology Unit, (Reports of the research project "User oriented software development and interface design", no. 3; Spinas P., Rauterberg M., Strohm O., Waeber D. & Ulich E. , eds.) 1990.
7 M. Rauterberg, Benutzungsorientierte Benchmark-Tests: eine Methode zur Benutzerbeteiligung bei der Entwicklung von Standardsoftware. Swiss Federal Institute of Technology - Work and Organizational Psychology Unit, (Reports of the research project "User oriented software development and interface design", no. 6; Spinas P., Rauterberg M., Strohm O., Waeber D. & Ulich E. , eds.) 1991.
8 F. Emery, Characteristics of Socio-Technical Systems. Tavistock Institute of Human Relations Doc. no. 527 (Belsize Lane, London NW3, UK) 1959.
9 P. Spinas, D. Waeber & O. Strohm, Kriterien benutzerorientierter Dialoggestaltung und partizipative Softwareentwicklung - eine Literaturaufarbeitung. Swiss Federal Institute of Technology - Work and Organizational Psychology Unit, (Reports of the research project "User oriented software development and interface design", no. 1, Spinas P., Rauterberg M., Strohm O., Waeber D. & Ulich E. , eds.) 1990.
10 E. Ulich, Arbeitspsychologische Konzepte der Aufgabengestaltung. Teubner, (Proceedings of "Software-Ergonomie ´89"; Maass S. & Oberquelle H., eds.) (1989) 51-65.
11 DIN 66 234, Bildschirmarbeitsplätze - Grundsätze ergonomischer Dialoggestaltung, DIN 66 234, part 8, Beuth 1988.
12 M. Rauterberg, From passive to active joins: a task oriented desktop interface conception for a relational DataBaseManagementSystem, (Proceedings of "Intelligent Text and Information Handling" RIAO'91 held in Barcelona, Spain - April 2-5) (1991) 809-821.

Ergonomics of Hybrid Automated Systems III

Proceedings of the Third International Conference on Human Aspects of Advanced Manufacturing and Hybrid Automation, Gelsenkirchen, Germany, August 26–28, 1992

Edited by

Peter Brödner
Institut Arbeit und Technik
Wissenschaftszentrum Nordrhein-Westfalen
Gelsenkirchen, Germany

and

Waldemar Karwowski
Department of Industrial Engineering
University of Louisville
Lousville, KY, U.S.A.

ELSEVIER
AMSTERDAM • LONDON • NEW YORK • TOKYO **1992**

 [ISBN: 0 444 89520 5]