Task Orientation and User-Oriented Dialog Design

Eberhard Ulich
Matthias Rauterberg
Thomas Moll
Thomas Greutmann
Oliver Strohm

Swiss Federal Institute of Technology (ETH)
Nelkenstrasse 11
CH-8092 Zurich
Switzerland

Work psychologists have introduced a concept, in which dialog design is a part of task design. A set of criteria of user-oriented dialog design is presented here. These criteria are consistently integrated into a control concept. Empirical investigations on some of these criteria are then described. In an experiment to prove the criterion *transparency,* a desktop interface (high transparency) and a conventional menu selection interface (low transparency) were compared. The main result is the clear superiority of the user interface with direct manipulation over the conventional user interface with menu selection. *Support* is another of the criteria of user-oriented dialog design that was empirically investigated. The advantages of goal- and task-oriented help messages are also described here. To make the criterion *flexibility* and the criterion *user-definability* practicable, the implementation of a dialog handler for user-tailorable systems is introduced. The criterion *participation* was investigated in field studies. It will be shown that in projects with active participation the costs were exceeded to a lesser degree than in projects with passive participation and to a much lesser degree than in projects without participation.

INTRODUCTION

Concepts of user-oriented dialog design are often limited to criteria as learning time, error rate or "usability," which are more or less precisely defined. In this paper, user-oriented dialog design is embedded in the wider context of job and task design. Concepts of task orientation and control are seen as the basis for a better understanding of what the user really needs. The relevance of some of the derived criteria is supported by current findings from our empirical research.

CONTROL, TASK ORIENTATION, AND DIALOG DESIGN

In his paper "Characteristics of Socio-Technical Systems," Emery (1959; 1978, p. 78) specified two prerequisites for the development of task orientation: "(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."

A further distinction is made between two functions or aspects of control: (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" (Emery, 1959; 1978, p. 79).

It becomes clear here that control in the psychological sense must be understood as the possibility to choose, as well as an opportunity to exercise influence. Emery (1959; 1978, p. 79) continues: "Thus, the knowledge that a skilled man brings to a job enables him to make choices between alternative modes and rates of operation that are not obvious to an unskilled man." Reference to the "materials" as well as the "alternative modes of operation" must relate to the use of the respective work means and tools.

If we try to apply this consideration to questions of dialog design as a part of task design, the first question arises regarding the selection of dialog technique. From the point of view of control, it is essential—and yet somehow trivial—to distinguish between three dialog forms. In computer-controlled dialogs, control rests with the computer; there is no possibility for the user to select or in any way influence the dialog. In user-controlled dialogs, the user determines the procedure and sequence of events, may choose between various alternatives, and, according to Blumenfeld's description of energy disposition (1932), is able to regulate his own tempo and degree of effort. The third form can perhaps be labeled a hybrid dialog, where control over certain segments of the task performance lies with the user, and for others, with the computer. In the area of dialog design, the application potential of hybrid techniques of this kind has been conceptionally, yet scarcely elaborated upon. Perhaps Kamoun, Debernard, and Millot (1988) addressed a similar issue in their article "On Human Decision Making and Manual Control."

Yet how should dialogs be designed? That is, which criteria must be met in order to satisfy the demand for control? If we turn to the German Industrial Standard "Principles of Dialog Design" (DIN, 1988, p. 6), we find, to begin with, the extremely important hint "that the type of dialog cannot be designed independently of the work task and the work organization." This means that user interface design is not simply a sub-problem that can be isolated and solved independently of the user's entire and overall task. An important contribution to a conceptual framework we find, for example, in the draft for a regulation on "Software Ergonomics in Office Communication" of the German Association of Engineers (VDI), in which the level of furthering competence and flexibility of action are named as the main criteria for software ergonomic job design. As for the concept of furthering competence,

> Ergonomic software design of man-computer interaction should contribute to making it possible for the user to have competent dealings with the system and thereby promote his competence of action. Competence of action means that the user has gained knowledge of the system and its organizational embedding, and that he is able to relate this knowledge to the tasks to be completed....Maintenance of and further improvement in competence of action presuppose the possibility of successfully applying the acquired knowledge, depending upon the task and situation. It must be possible to carry acquired knowledge over to changed situations. (VDI, 1988, p. 15)

Which requirements, however, should a user-controled dialog fulfill? At the 1986 IFIP Conference on System Design for Human Development and Productivity, a set of criteria that were unrelated to each other was presented (Ulich, 1987). In the meantime, however, it can be shown that our criteria can be consistently integrated into a control concept. The result is outlined in Table 1.

Table 1. Concept of User-Oriented Dialog Design (from Ulich 1989).

User-Oriented Dialog

Task Orientation	Calculability as a Prerequisite for Control	Control
• Completeness	• Transparency	• Flexibility
• Variety of Demands	• Consistency	• Individual Selection Possibilities
• Interaction Possibilities	• Compatibility	• User Definability
• Learning Potential	• Support	• Participation
• Autonomy	• Feedback	

The control aspects of transparency and predictability referred to by Troy (1980), Hacker (1986), and other authors are necessary, but in no way sufficient prerequisites for the possibility of realizing the specified control functions. In fact, situational transparency and the possibility of predicting a situation's outcome (without being able to influence the situation and its outcome) can of course mean even a total loss of control.

As in the diagram, and in concordance with Spinas (1987), the criteria of transparency, consistency, compatibility, support, and feedback are assigned to the orientation function here. Thus, with the fulfillment of each of these criteria, a contribution is made to the calculability of the system's behavior. This, however, as previously mentioned, is not control in itself, but rather an indispensible prerequisite for control. Control itself consists in the possibility to choose and in the possibility to exercise influence.

Thus, we have a meaningful classification of criteria for dialog design. We should not, however, content ourselves with this, as the development of software for dialog programs finally determines the division of functions between man and computer. In this context, we find, in many articles on questions of user-friendliness, a more or less elaborated concept of task appropriacy. In many cases, what is meant by this is not at all defined. The German Industrial Standard "Principles of Dialog Design" (DIN, 1988, p. 2) reads as follows: "A dialog is task appropriate if it supports performance of the work task without unnecessarily burdening the user with the characteristics of the dialog system." Thinking further reveals that what a definition of this kind really means is that a dialog can also be task-appropriate if the task itself is neither humane nor corresponds with criteria of economic efficiency. And that is why we should ask ourselves the question whether task appropriacy should be understood in a broader sense, namely as adequate support of tasks that correspond with our criteria for task design. One attempt has been made to classify these criteria—which essentially again lead back to Emery's work—in order to come closer to an overall concept of user-friendliness (see Table 1).

Following, several of the criteria will be discussed in more detail. First the concept of the complete task embedded in the tradition of the theory of action regulation will be introduced. Then, several investigations pertaining to the dimension *calculability*, specifically the criteria *transparency* and *support*, and to the dimension *control*, specifically the criteria *flexibility*, *user-definability*, and *participation*, will be presented.

Completeness—the first criterion named in the concept of user-oriented dialog design—is in fact the most important. It is, to a large extent, identical to the concept of the complete task.

THE CONCEPT OF THE COMPLETE TASK

For Hacker (1986, p. 61), the work order, its interpretation or its acceptance as a work task, is "the central category of psychological consideration of activity..., as decisive specifications for the regulation and organization of the activities occur with the 'objective logic' of its contents." For Volpert (1987, p. 14), "The character of a 'point of intersection' between the organization and the individual renders the work task the psychologically most relevant part of the given work conditions." Both quotations make it clear that for work psychologists oriented in the activity or action theory, the work task becomes the most important point of departure in job design. In this context, great importance is attached to the concept of the complete task.

More than 50 years ago, the German psychologist W. Hellpach (1922, p. 27) described the task concept as follows:

> A task consists of one's own planning and design—if not design of the task, design of its solution, including choice between various possibilities, decision for one of these, taking responsibility for the decision, the ways of performance, while completing the task the constant, infinitesimal evaluation of success related to the mentally represented goal, and finally, the belief that it was well done.

Hellpach's description already included the main elements of the concept of the complete task as defined by Tomaszewski (1981, p. 23):

> A complete task contains three basic elements: diagnosis of the starting situation, outline (project) of a new situation and a program of realizing activities. During the course of the task performance, these three elements are supplemented by a continuous diagnosis of the changing situation.

These are the basic constituents of Hacker's concept of complete activity (Hacker, 1986), as well as Volpert's concept of complete action (Volpert, 1987).

Characteristics of complete tasks, which must be taken into consideration when designing tasks, are, according to the concept presented here:

1. independent setting of goals that are embedded in the superimposed goals;
2. independent action preparation in the sense of taking on planning functions;
3. selection of the means including the necessary interaction for goal attainment;
4. performance functions with feedback on performance pertaining to possible corrections of actions;
5. control with feedback on results and the possibility of checking the results of one's own actions against the set goals.

Incomplete activities—or, according to Volpert (1974), partialized actions—"lack, to a large extent, possibilities for independent goal-setting and decision-making, for developing individual work styles or sufficiently precise feedback" (Hacker, 1987, p. 35). Complete activities (or tasks) offer the possibility of setting goals and sub-goals, as well as of offering decision-making possibilities during the various phases of task completion, and therefore provide latitude of activity or action. Complete activities are therefore becoming fundamental for realizing the concept of user-definability (Ulich, 1987).

DIRECT MANIPULATION: A SPECIAL CASE OF TRANSPARENCY

State of the Art

The advantages of direct manipulation seem to be so obvious that there are hardly any experimental studies investigating the superiority of direct manipulation over other styles of interaction (Shneiderman 1987; Smith & Mosier 1986); therefore Hutchins, Hollan, and Norman (1986, p. 123) demand a more detailed empirical evaluation of the interface with direct manipulation to bridge the "gulf of execution" and the "gulf of evaluation." The importance of *compatibility* and *transparency* to build this bridge was emphasized earlier (Ulich, 1987a) (see also Table 1).

Altmann (1987) compared one user interface with direct manipulation (MacWrite) and another with command language (WordStar) with regard to how difficult it was for novices to learn to handle them. Novices with MacWrite did considerably better than novices with WordStar. Whiteside, Jones, Levy, and Wixon (1985) have come to apparently opposite results. Margono and Shneiderman (1987) were able to show the superiority of user interfaces with direct manipulation over user interfaces with command languages.

Questions

As only insufficient and contradictory results are available so far, the following two interaction styles must be compared within the framework of this study (Rauterberg, 1989a, b). The first user interface is characterized by direct manipulation (more precisely: a desktop interface), and the second user interface is typified by conventional menu selection.

User interface with menu selection ("ascii"). The conventional, ascii-oriented user interface—the dialog is carried out by means of function keys (including cursor keys) and selection menus.

User interface with direct manipulation ("mouse"). The bit-mapped, graphic-oriented user interface—the dialog is carried out with a mouse by clicking into the mouse-sensitive areas of a desktop interface; most of the menu options of this desktop interface could be chosen with function keys, as well.

The following two questions must be answered:

Question 1. Is there an ergonomically relevant difference between these two types of user interfaces with regard to task solving time?

Question 2. Is there a significant interaction between the type of task and the type of user interface?

Method

A relational database system with two selected user interfaces was available for ten benchmark tasks, and exactly the same database machine served as the application manager.

Each time a key was pressed (or a mouse clicked) the keystroke and a timestamp belonging to this keystroke was automatically recorded in a logfile.

A two-factorial variance analysis with repeated measurements for one factor was used; the *first factor* was the "type of user interface" ("ASCII" on IBM-compatible PCs under MS-DOS vs. "MOUSE" on IBM-compatible PCs under GEM), the *second factor* was the ten "benchmark tasks." These two factors are two independent variables. As a result, there are two independent samples (2*6 users = 12 *N*).

The *dependent variables* ascertained were: pure task solving time per task according to the logfile record, excluding system response time (Usertime); the medium reaction time per keystroke (Timelevel, sec/key) as the average value of all time-stamps per task; the *control variables* (covariates) were: the self-reported number of hours of general previous experience with Electronic Data Processing (EDP) and the self-reported number of hours of specific previous experience with the respective user interface.

Description of the experimental subjects.

The 12 subjects who had been working for several years with the respective user interface in their daily work were referred to as expert users; these experts received no payment.

Sample 1 (expert, ascii). Average age of 38; 6 men; 8,750 hours of general previous experience with EDP (± 5,100 hours; range: 4,080 hours - 17,580 hours); 1,740 hours of experience with specific user interface: "menu selection" (± 1,600 hours; range: 314 hours - 4,573 hours).

Sample 2 (expert, mouse). Average age of 38; 6 men; 3,690 hours of general previous experience with EDP (± 3,440 hours; range: 876 hours - 9,920 hours); 1,500 hours of experience with specific user interface: "desktop" (± 1,200 hours; range: 413 hours - 3,840 hours).

In contrast to the inexperienced subjects of the most published investigations (e.g., Altmann, 1987), this previous experience with EDP and the specific interface type is a valuable feature of the experimental subjects in this investigation.

Course of the investigation.

After the subjects had filled in a questionnaire about their previous experience (four scales: programs, languages, computer systems, operating systems), they began work on the tasks. At the end, every subject completed an evaluation questionnaire. The investigation lasted between 195 and 270 minutes per user (individual sessions).

All subjects (except for one) were able to finish the ten set tasks. The order of tasks was the same for everybody. The subjects were only allowed to continue with the next task (controlled by the investigator) when they had finished the previous one.

Description of the ten benchmark tasks.

The ten tasks were selected according to whether they allowed the subjects to use exactly the same functionality of the application manager under both types of interfaces and to take those working steps which are most common in daily database work. The test database consisted of three files, which contained the necessary attributes to manage a camping place.

Tasks 9 and 10 were selected in order to test whether the design of the user interface was appropriate for these types of tasks. The users of the interface with direct manipulation were requested to make up a text document for Tasks 9 and 10 with an external text editor, according to the syntax of a given simple retrieval language, whereas the users of the interface with menu selection were able to define and make up the relations interactively in a "list" module.

The descriptions of the ten benchmark tasks (enclosed in "...") and the solution for each task follows:

Task 1. "Find out, how many data records are in the file Address, in the file Place, and in the file Group."

Activating a certain menu option and reading the file size of each file: Address = 280 records, Place = 17 records, Group = 27 records.

Task 2. "Delete only the last data record of the file Address, the file Place, and the file Group (sorted by the attribute 'name')."

Opening (sorted according to a given key), selecting, and deleting the data record of each file: Address, Place, Group.

Task 3. "Search and select the data record with the key 'D.8000C O M' in the file Address, and show the content of all attributes of this data record on the screen. Correct this data record for the following attributes:
State: D; Place offer: 07; Remarks: The ATARI system dealer can give a demonstration of ADIMENS ST."

Selecting a certain data record (file: Place), correcting the data record with regard to four attributes.

Task 4. "Select all data records with a valid content of the attribute 'place no.' in the file Address. Correct this attribute so that all selected data records afterwards contain the place number '07.'"

Selecting a number of data records (file: Address), correcting every data record with regard to one attribute.

Task 5. "Define a filter for the file Place with the following condition: all holidaymakers arrived on 02/07/87. Apply this filter to the file Place, and show the content of all selected data records in the mask browsing mode on the screen."

Defining a filter for one attribute (file: Place), applying the filter to the data records; displaying the content on the screen of the data records found.

Task 6. "Load the calculation program named '.\DB-CAMP\END' (menu interface) or 'PLAENDPR.CAL' (desktop interface) respectively for the file Place and apply this calculation to all data records on the file Place. Show the results on the screen. Store the results permanently only for the data records of departed holidaymakers."

Loading a calculation program (file: Place), applying the calculation to all data records, displaying the content of each data record on the screen and storing the results only for a selection of data records.

Task 7. "Make a list with the attributes 'name', 'birth date', and 'national status' of all members of the group of 'D.Trainer, Siegfried' of the file Group, and show the results on the screen. Then subsequently print this list."

Selecting a number of data records (file: Group), making up and printing out a list for the data records found that consist of three attributes.

Task 8. "Prove the existence of the data record with the content 'D.Verlängerungswunsch, Sonja' of the attribute namekey in the file Address. Then make a bill for the data record with the content 'D.Verlängerungswunsch, Sonja' for the attribute namekey in the file Place; load the form document 'PLARECHN.MIX' after activating the icon 'mixing'. Then subsequently print this bill."

Seeking a (nonexistent) data record (file: Address); selecting a data record (file: Place), loading the calculation program, mixing it with the data record, printing the calculation made.

Task 9. "Make a list including the attributes 'place no.', 'last name', 'birthdate', 'citizenship', and if need be the attribute 'passport no.' of the place tenant of all present holidaymakers only. Then subsequently print this list."

Selecting a number of data records (file: Group), making up and printing out a list consisting of five attributes from file Place and Group (two-file relation).

Task 10. "Make a list including the attributes 'place no.', 'first name', and 'last name' of all holidaymakers only, who are now staying at place '07' or have been at any time at place '07'. Subsequently print this list. Bear in mind, that correct data records can exist in all three files."

Selecting a number of data records (file: Place and Address), making up and printing out a list consisting of three attributes from file Place, Address, and Group (three-file relation).

Results

First, the results were analyzed from the point of view of a two-factorial design across all ten tasks (factor "task" and factor "interface") with repeated measurements for the factor "task." The mean and standard deviation of the two dependent variables are given in the Tables 2 and 3. In the last columns of Tables 2

Table 2. The mean and the standard deviation of the task-solving time and the ratios of "mouse"-solving time to "ascii"-solving time (variable USERTIME) for the ten benchmark tasks

Task No.	"mouse" Interface (*N*=6) (s)	"ascii" Interface (*N*=6) (s)	Ratios "mouse"/"ascii" (%)
1	53± 23	197± 66	27
2	125± 52	320±200	39
3	155± 47	346±182	45
4	154± 61	445±206	35
5	114± 44	200±162	57
6	416±156	698±256	60
7	180± 62	409± 98	44
8	412± 93	693±189	59
9	1591±564	1261±415	126
10	929±165	1568±477	59

Table 3. The mean and the standard deviation of the medium reaction time per keystroke and the ratios of "mouse" reaction time to "ascii" reaction time (variable TIMELEVEL) for the ten benchmark tasks

Task No.	"mouse" Interface (*N*=6) (s/key)	"ascii" Interface (*N*=6) (s/key)	Ratios "mouse"/"ascii" (%)
1	4.0±1.2	9.1±1.0	44
2	3.2±1.0	5.3±0.7	60
3	1.3±0.4	2.2±0.8	59
4	4.5±1.8	3.0±0.6	150
5	3.3±1.3	3.3±1.2	100
6	4.6±1.2	4.6±1.4	100
7	3.3±0.5	3.0±1.0	110
8	3.5±1.5	2.8±0.5	125
9	4.9±1.6	5.0±1.1	98
10	6.7±1.9	5.9±1.3	114

and 3, the ratios of task solving time of the desktop interface and of the conventional interface are given.

On the average, users of the interface with direct manipulation ("mouse") needed only 55% of the task solving time (USERTIME; see Table 2) required for the users of the interface with menu selection ("ascii"). Only in Task 9 was this ratio reversed in favor of the interface with menu selection (126%, see Table 2; see also Figure 1).

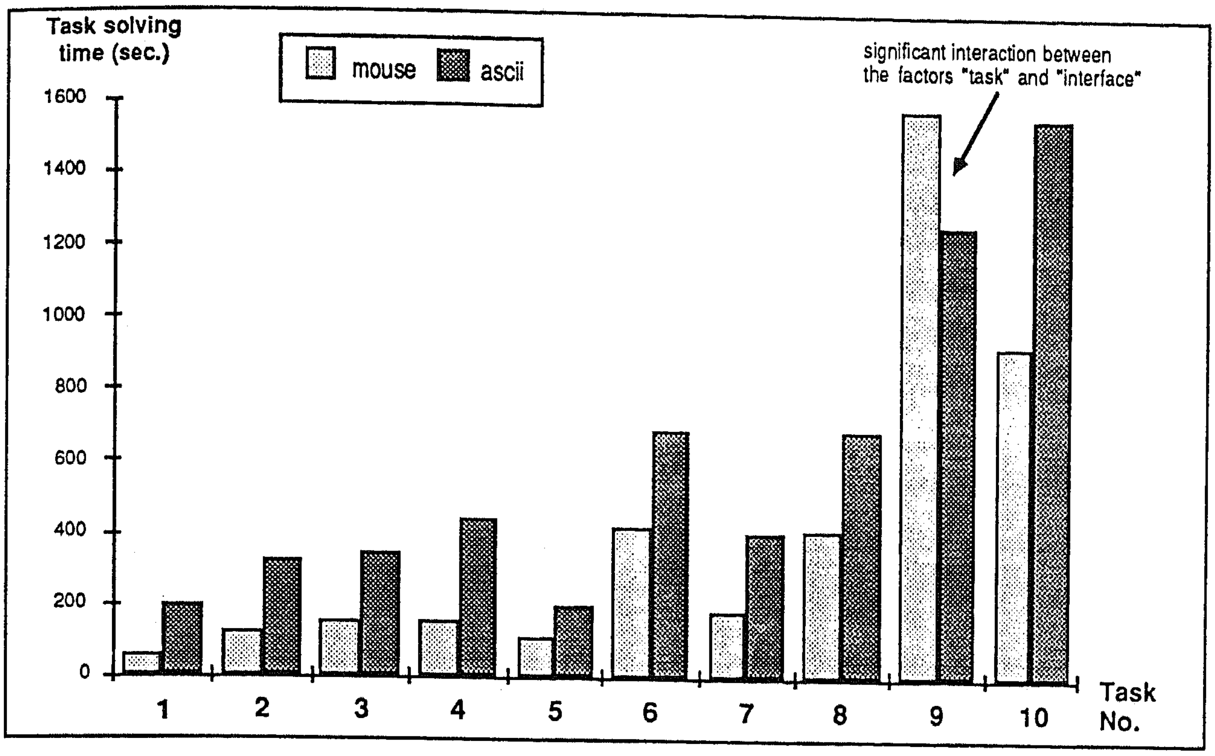


Figure 1. Task solving time (variable USERTIME; excluding the system response time) of the sample with the desktop interface ("mouse") and the sample with the menu selection interface ("ascii") divided in the ten benchmark tasks ("Task No.").

The subjects with the direct manipulative interface needed only 374 seconds ± 481 seconds (on the average of all ten tasks) compared with the subjects with menu selection: 581 seconds ± 467 seconds. This difference in task-solving time between the two groups is significant (main effect of factor "interface": p = 0.001, or 0.001; see Table 4).

On the whole, there is no difference (main effect of factor "interface") in the reaction time per keystroke (TIMELEVEL: p=0.708, resp. 0.312; see Table 5). Particu-

Table 4. Results of the two-factorial variance analysis of the task-solving time (variable USERTIME in sec.) for the ten benchmark tasks

	Covariate: "general experience with EDP"			Covariate: "specific interface experience"		
Two-factorial VA	*df*	*F*	*p*	*df*	*F*	*p*
factor "interface"	1	16.78	0.001	1	24.3	.001
factor "task"	9	46.43	0.001	9	46.3	.001
factor "interface"⊗"task"	9	2.98	0.004	9	3.0	.003

Dependent variable: USERTIME.

Table 5. Results of the two-factorial variance analysis of the medium reaction time per keystroke (variable TIMELEVEL in sec./key) for the ten benchmark tasks

	Covariate: "general experience with EDP"			Covariate: "specific interface experience."		
Two-factorial VA	*df*	*F*	*p*	*df*	*F*	*p*
factor "interface"	1	0.14	.708	1	1.04	.312
factor "task"	9	4.00	.001	9	3.75	.001
factor "interface"⊗"task"	9	1.75	.088	9	1.60	.126

Dependent variable: TIMELEVEL.

larly remarkable, however, is the relatively long reaction time by the users of the interface with menu selection in Task 1 on one hand ("mouse:ascii"=44%; see Table 3) and on the other, by the users of the interface with direct manipulation in Task 4 ("mouse:ascii"=150%; see Table 3).

For all tasks except Task 9, the experts with the direct manipulative interface needed less time to solve the tasks than the experts with the menu selection interface (see Figure 1). This effect can also be shown in the significant interaction of factor "interface" ⊗ "task" ($p = 0.004$, resp. 0.003; see Table 4). Remarkable is that all users virtually never used the online help function.

Discussion

Although the subjects of the interfaces with menu selection had more than twice as much experience with EDP and more specific experience with their interface (see above: "Method"), a clear superiority of the user interface with direct manipulation over the conventional user interface with menu selection can be shown. Therefore, the answer to *Question 1* "Is there an ergonomically relevant difference between these two types of user interfaces with regard to task solving time?" is *Yes.* The user interface with direct manipulation makes it possible to reduce the problem-solving time by a factor of 2. This clear difference in task-solving time cannot be explained by the different reaction time per keystroke, because obviously the users of the interface with direct manipulation on the whole found it necessary to press fewer keys to finish each task (the continuous "mouse" movements of the users of the desktop interface are not included in this result). This result is an effect of the high transparency of the direct manipulative interface and not only an effect of the different dialog structures of the two interfaces. The users of the menu selection interface type needed several unnecessary keystrokes to make sure that they were in an appropriate solution state.

It is not enough, however, simply switching over from menu selection to direct manipulation (Task 9!): Careful, appropriate dialog design must be made for each type of task (*Question* 2: "Is there a significant interaction between the type of task and the type of user interface?" Answer: *Yes*). Once the experts in direct manipulation had found a solution scheme for Task 9 (definition of a two-file relation), they were able to apply it profitably to the more complex Task 10 as well. This suggests that the user interface with direct manipulation promotes learning.

Contrary to the often-voiced opinion that direct manipulation of the user interface is good for beginners only, it is the experts of direct manipulation who are the real "winners" of this comparative study. This can be explained by the fact that the relevant dialog context of the user interface with menu selection gives too little or insufficient direct feedback about the state of the dialog and application manager.

On the whole, it can be emphasized that the user interface with direct manipulation (using the "mouse") is clearly superior as a general element of interaction to the conventional user interface with menu selection (using "function keys"). This is particularly true for users with long previous experience in EDP work.

SUPPORT

Support is another of the criteria of user-oriented dialog design, which our research group is empirically investigating. In spite of the fact that most guidelines for user-oriented software design include recommendations concerning user support, online assistance, or online help (Smith & Mosier, 1986, Ulich, 1987a), there is little empirical knowledge about the effectiveness of different forms of online help, and there are discrepancies in the sparse results reported from empirical research on the utility of help systems. Because more investigation needs to be carried out in order to determine what forms of online assistance will support users, we have been analyzing, evaluating, and developing online support since 1985.[1] First, we evaluated a context-specific online help system written by software designers. Proceeding from our empirical data of this study (Moll & Sauter, 1987), we then developed and evaluated an online tutorial that should support users in real problem situations by handling complete working tasks (Moll & Fischbacher, 1989).

Different Forms of Support

According to our concept of user-oriented dialog design (see Table 1) based on the concepts of task orientation and control, support should facilitate the user's orientation. The aim of developing and designing support systems is to provide users with the information they need to render software more transparent (information

[1] These investigations were carried out and financed in cooperation with the software company MECASOFT S.A., Switzerland.

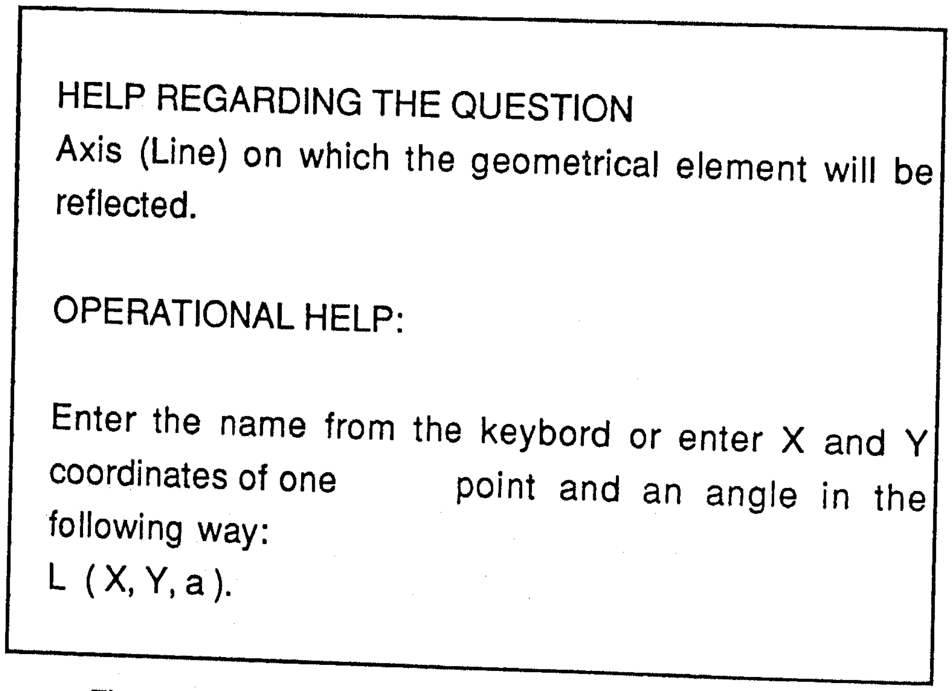
on important relations, explanations of events, the consequences of actions, etc.), in addition to providing them with information that would enable them to anticipate events (forming hypotheses and making predictions) as well as allowing them to exercise influence in keeping with their own goals.

Nievergelt (1982, p. 203) pointed out that most difficulties encountered by novices are connected with questions that we refer to as problems of orientation. Providing information regarding answers to questions as *Where am I?, What can I do here?, How did I get here?* and *Where else can I go and how to I get there?* is a first and important step toward user support.

Support of this type can be found in context-specific help systems. In the help system we evaluated (implemented in the interactive numerical control programming system MECANIC), a context-specific help message was assigned to each menu element and systems prompt. At every dialog point users receive information regarding the questions: "What can I do now?" and "What are the input rules?" (see Figure 2).

From the work psychology point of view, concepts of *task orientation* and *control* are the basis for an understanding of the function of support. User support must go beyond providing information that relates to the dialog context and software. We attach great importance in this context to the concept of the *complete task*. Online support should support users in successful handling of complete job tasks. This is illustrated in the following example:

> After defining geometric elements, writing a machining program, and inputting the necessary technical commands, a user wants to generate a program with which he can cut a die at wire cut electro-discharge machining. In order to reach his task related goal, he must select *Postprocessor* from a specific menu and then select *PP Run*



HELP REGARDING THE QUESTION
Axis (Line) on which the geometrical element will be reflected.

OPERATIONAL HELP:

Enter the name from the keybord or enter X and Y coordinates of one point and an angle in the following way:
L (X, Y, a).

Figure 2. Example of a context-specific help message

from the subsequent menu. Assuming that he does not select Postprocessor but a different menu item and consequently encounters another selection on the ensuing menu, he finds himself in an area of the dialog in which he can no longer reach his task-related goal.

In such situations (which, according to our investigations, are typical of novices), system-related information of the type "What can I do here?" is of no further help to users. Rather, users need task-related messages from which they can deduce what they must do in order to carry out their job task.

Task-oriented support must take into consideration Troy's (1986) distinction (on the basis of Rice, 1958) between primary tasks and secondary tasks. Whereas closer dealings with software (e.g., particular menu selections) are construed as secondary tasks, the primary task is understood as the actual job task itself. Technical problems originate from the primary task (from producing a matrix at a wire cut electro-discharge machining, for example). These problems would also arise if the task were carried out without a computer. Interaction problems originate from the secondary task (i.e., from interaction with the software). This can, for example, arise from incorrect understanding of the syntax. Evaluations of thinking errors in real work situations have established that these constitute mostly a mixture of problems related to both tasks (Zapf & Frese, 1989, 211). From this, we can deduce the dictate that support must always relate to primary and secondary tasks.

Meca-Teach: Task-oriented Support in a Learning Environment

During the last two years, we have been developing a learning environment for a Numeric Control (NC) programming system, which meets the aforementioned support requirements. This learning environment is more fully described in this section.

Once users have started MECANIC (the NC Programming System), they can access a learning environment (Meca-Teach) by simply pressing a function key. By doing so, a tutorial window will appear on the righthand side of the screen. Function keys will appear in the lower section of this window, tutorial messages in the upper sections. Users can then begin to work on an exercise.

The exercise (a standard task for which the tutorial messages were written) is an example of a typical and complete task as these will later come up at the programming work place. Users receive a drawing of a tool and must define the geometric elements as well as input a graphics application program including the necessary technical commands. After completing this sequence of subtasks, they can generate an NC program that could be used for cutting dies with wire cut electro-discharge machining. The tutorial window messages that they receive while programming in the normal system refer not only to the secondary task (in addition to stating in the dialog why the user should perform certain operations), but also refer to their primary task and specify what would happen at the machine if they actually ran their program (see Figure 3).

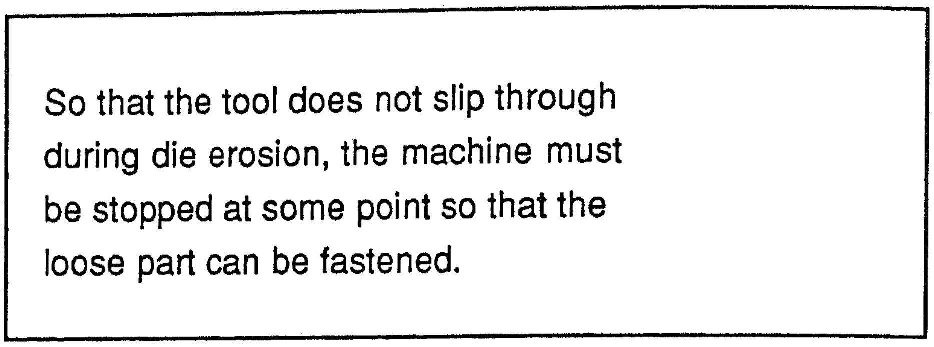

So that the tool does not slip through during die erosion, the machine must be stopped at some point so that the loose part can be fastened.

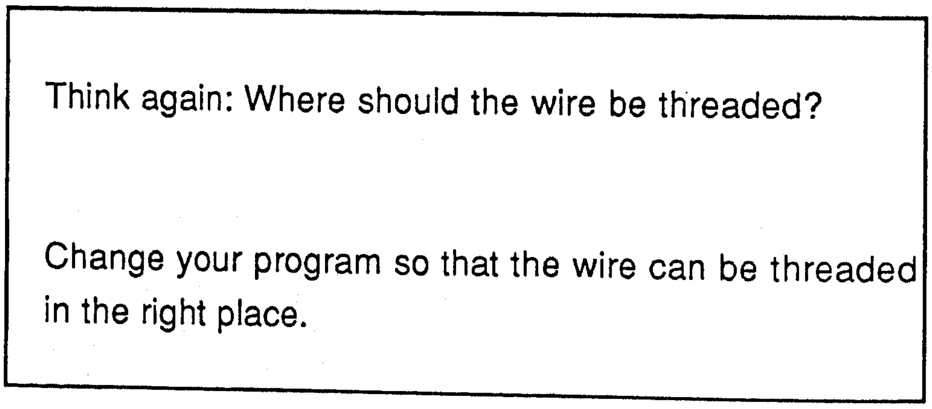
Figure 3. Online tutorial message relating to the user's primary task

If users come up against a problem situation while working on their exercise with the online tutor, the tutor will first send *a general message to stimulate their reasoning processes* (see Figure 4).

If, after reconsideration, users still do not know how and where to input their information, they can request *operational help* that relates to their particular problem situation. If users do not know, for example, which NC machine command indicates where the wire should be threaded, they receive information as in Figure 5.

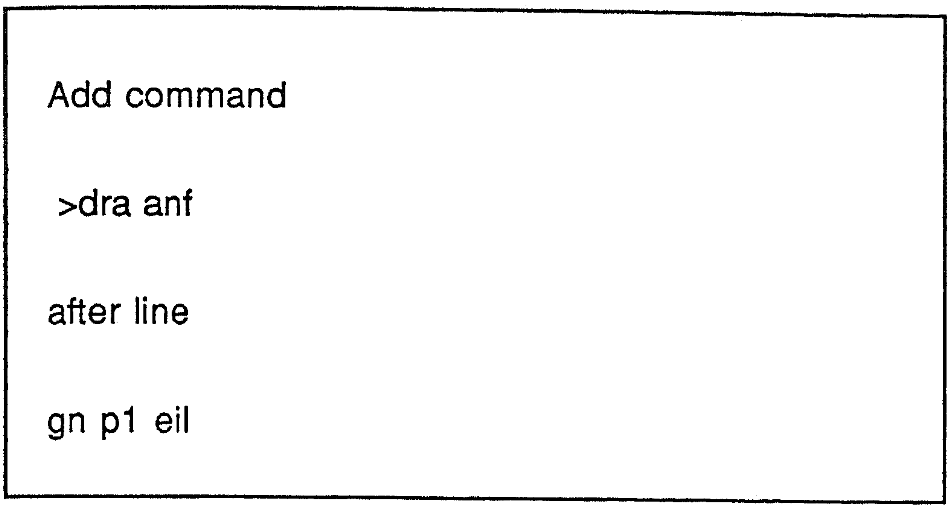
Because conveying background information (i.e., knowledge about why certain operations are carried out) can lead to improved learner programming performance, users of the MECANIC learning environment have the option of asking *why they should carry out particular operations* (see Figure 6).

In addition, users can at any time actively request tutorial support. If users press the tutorial function key *What next?*, the tutor will suggest subgoals on how to proceed (taking into account what users have already done). If users wish to reach a particular goal and see no solution of how to attain it, they find themselves again in a problem situation and can request support for the questions *How can I reach this?* and *Why must I do that?* It was deducted from the principle of differential and dynamic work design (Ulich, 1987b) that users should not be forced into the "one best way" approach when working on exercises. When users work on exercises in our learning environment, they have access to all of the available functions


Think again: Where should the wire be threaded?

Change your program so that the wire can be threaded in the right place.

Figure 4. Online tutorial message stimulating the user's reasoning processes



```
Add command

 >dra anf

after line

gn p1 eil
```

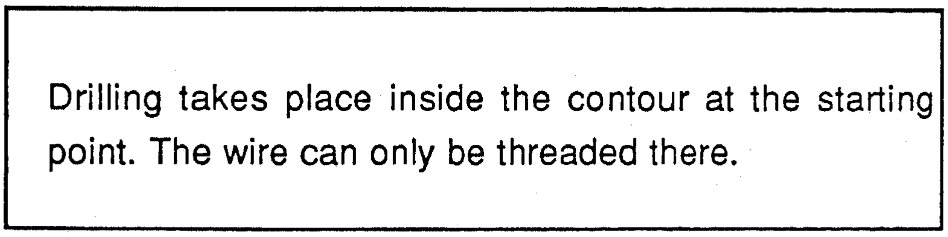
Figure 5. Online tutorial message response to the question of how a particular goal can be reached.

in the program. Users work on an exercise *completely* (i.e., they independently plan how to go about solving the task, decide on a possible solution utilizing the software's total flexibility)—(for which no single alternative is prescribed), write their program, and then check whether it corresponds with the set task.

Users can benefit from the application program's available flexibility and are able to solve the task in several different ways. If they are still uncertain, they may enter into a dialog and have their questions answered; or, they may choose to bypass the dialog completely and simply type in the necessary data on one line in abbreviated form. The online tutor will only intervene when users have carried out an operation that no longer relates to the task.

Evaluation of Different Forms of Online Support

In two field studies, we evaluated the effectiveness of different forms of online support. The real behavior of end users of the NC programming system was studied by means of a combination of methods (standard task, logfile recording, thinking aloud, video self-confrontation; Moll, 1987). Users had 30 minutes to work on the standard task. We analyzed situations in which users called the context-specific



Drilling takes place inside the contour at the starting point. The wire can only be threaded there.

Figure 6. Online tutorial response to the question of why a particulal operation must be carried out at a particular place.

help or interacted with the online tutor. We observed and analyzed the behavior of 24 end users (tool and die makers) interacting with the context-specific help system, and investigated the interaction between the online tutor and 9 subjects. What happened after a help call or interaction with the online tutor was listed in the following categories:

1. Reaches goal: The user has a problem, gets a help message, types in the next input, and reaches his goal immediately.
2. Reaches goal in a roundabout fashion: The user has a problem, gets a help message, reads it, first carries out wrong or unnecessary operations, but finally types in the right input and reaches his goal.
3. Changes goal: The user has a problem, gets a help message, reads it, types in the wrong input, fails to reach his goal, and then types in input which brings him to a goal different from his first intended goal.
4. No solution: The user has a problem, gets a help message, reads it, and tries again and again to reach the goal, but cannot find a solution to complete the subtask.

Table 6 shows that users of the context-specific help system were able to solve their problems after reading a context-specific help message in only 37% of the cases. The users had problems exploring the right goals and subgoals for solving the task, did not have the necessary dialog- and application-oriented background knowledge to understand the system's messages, lost orientation in the system, and carried out operations that had no connection with their goals.

In contrast to this result, our analysis indicates that computer novices profit from goal- and task-oriented messages. They were able to solve their problems after reading *task-oriented messages* in the learning environment in 82% of the cases. After reading the task-oriented help messages, the users were able to reach solutions to their problems more frequently than after reading the system-oriented help messages. To conclude, online support must be goal- and task-oriented, and should relate to the user's primary, as well as secondary, task.

Table 6. Comparison of Different Forms of Support

	Reaches Goal	Reaches Goal in a Roundabout Fashion	Changes Goal	No Solution
Context-specific Helpsystem (System-oriented)	30%	7%	17%	46%
Online Tutorial (Task-oriented)	73%	9%	14%	4%

$\chi^2 = 65.80$; $df=1$; $p \leq 0.0001$.

IMPLEMENTATION OF FLEXIBLE AND USER-TAILORABLE DIALOG SYSTEMS

As stated in Table 1, dialog systems should offer the user alternative procedures for accomplishing his or her goals (flexibility) and allow the user to tailor the system according to his or her needs (user-definability). These criteria take into account both inter-individual and intra-individual differences of the users. In this chapter we will show how these global objectives can be translated into specifications for a user-tailorable dialog handler.

The Dialog Handler for User-tailorable Systems

We are currently developing a concept for a portable dialog handler that should support the designer in several aspects:

1. Skeletons of new applications should be easy to develop to allow rapid prototyping.
2. Possibilities for user-tailoring should be included into the dialog handler as far as possible to facilitate the development of tailorable interfaces.
3. The portation of application systems between very different environments should be facilitated. This requires a special architecture for the dialog handler.

In this article we will restrict ourselves to the user-tailoring facilities of the dialog handler.

Possible Ways of Individual System Tailoring

User tailorizations are possible on different levels and with different properties and consequences, ranging from simple window rearrangements on a desktop interface, to complex end-user programming with 4th-generation languages. There is also a variety of methods for accomplishing user tailorizations, including single commands embedded in the application, as well as textual user profiles where complex modifications and macros are formulated by means of a special syntax.

For the discussion of the properties, we have classified the different possibilities using a matrix scheme. The modified system components are listed in the columns; the rows of the matrix consist of the different tailorization methods.

The modified system components are classified into six categories (C1–C6):

C1. *Input device:* The input device can be selectable (e.g., mouse or keyboard).
C2. *Physical screen layout:* Size, positions, colors, etc. of windows, menus, texts, etc. can be modified.

C3. *Data display mode:* The display information can be changed qualitatively. Information can be shown or hidden, alternative display modes (text/graphic) are selectable.
C4. *Scope of commands:* Commands can be included or excluded from a predefined repertoire, but the definition of new commands is not possible.
C5. *Command structure:* New commands can be developed (e.g., by means of a macro facility).
C6. *Individual applications:* New applications with dialog structures can be developed.

The methods for accomplishing the individual modifications can be classified into four categories (M1–M4):

M1. *Selectable alternatives:* Different procedures for accomplishing a goal are implemented into the system in a parallel fashion and can be selected (but not modified) by the user.
M2. *Online configuration commands:* User tailorizations can be accomplished by means of special commands included in the applications.
M3. *Configuration program:* The modifications can be accomplished interactively by means of a special program or module outside the application.
M4. *Configuration file:* The modifications can be accomplished by editing a special configuration textfile with a text editor.

Basically any of the six different system components can be modified using any of the four methods. Thus, 24 different categories of user modifications are listed in the classification matrix (although some are not meaningful). It is now interesting to analyze which of the 24 categories are suitable under which circumstances. This question will be addressed below.

Necessary Knowledge for Individual System Modifications

It was stated before that in order to influence (modify) a dialog system, the system must be — as necessary preconditions — transparent and predictable for the user (control concept). For individual user-tailoring, this means that the user must have acquired a relatively deep understanding of the system before he is able to modify the system properly. Thus, individualizations that require a modification of the system by the user (methods M2, M3, and M4) should be provided only for experienced users, typically regular users. For users with less experience (casual users), individualizations should be provided by predefined alternatives that can be selected by the user (method M1). This way, the user does not have to learn how to modify the system, he must simply be aware of the possible selections.

The necessary knowledge for user-tailoring a dialog system can be analyzed in more detail using the matrix classification scheme. In Table 7 the necessary system knowledge is listed for each of the 24 individualization categories.

Table 7. Necessary System Knowledge for Each of the 24 Individualization Categories

	C1 Input Device	C2 Screen Layout	C3 Data Display Mode	C4 Scope of Commands	C5 Command Structure	C6 Indiv. Applications
M1 Selectable alternatives	ALTER	ALTER	ALTER	ALTER	***	***
M2 On-line commands	ALTER MCMD	ALTER MCMD	ALTER MCMD	ALTER MCMD	ALTER MCMD	***
M3 Configur. program	ALTER CONP	ALTER CONP	ALTER CONP	ALTER CONP	ALTER CONP	ALTER CONP PLSYN
M4 Configur. file	ALTER PLSYN TXTED	ALTER PLSYN TXTED	ALTER PLSYN TXTED	ALTER PLSYN TXTED	ALTER PLSYN TXTED	ALTER PLSYN TXTED

Abbreviations:

ALTER:	The existing alternatives must be known.
MCMD:	The modification command must be known.
CONP:	The use of the configuration program must be known.
PLSYN:	The syntax of a programming language or a description language must be known.
TXTED:	The use of a text editor must be known.

*** This category is not meaningful.

Table 7 shows that method M1 (selectable alternatives) requires no special knowledge about individualization procedures and is therefore most suitable for casual users, as has already been claimed. More interesting is the distinction between methods M3 and M4 (configuration program/configuration file), since these methods offer the same possibilities (a high degree of freedom), but require different knowledge. Table 7 shows that a special syntax in addition to the use of a text editor must be known for the configuration file method M4.

Method M3 requires only knowledge about a special configuration program and will therefore be easier to understand. Although we explicitly do not promote "ease of learning" as a design criterion, a system should be easy to understand ("transparent") when it is only occasionally used. This assumption is supported by an experimental investigation (Greutmann, Ackermann, & Krebs, 1988). Since user-tailoring is typically not the main part of the user's job, configuration programs or configuration files will only be occasionally used. In this case, a configuration program is superior to a configuration file that rather prevents than encourages the user from tailoring his system.

However, this superiority will not hold if the user creates his or her own application programs (modification of component C6). Here user-tailoring becomes a dominant part of the user's job and will be carried out regularly. In this case (see Table 7) the benefits of a configuration program are not obvious and the method of a configuration textfile has to be considered as a suitable solution that could even be more flexible than a configuration program. However, end-user programming is a very special case of user-tailoring and should be analyzed separately.

Constraints of User-tailoring

Table 1 shows that user-definability is an aspect of control. If a designer wants to provide this aspect, he must make sure that other aspects (e.g., transparency, consistency) that are prerequisites for control are not negatively affected. By means of a systematic analysis procedure (Greutmann & Ackermann, 1989), we tried to exclude such negative backward effects.

Such effects can arise when the same dialog system is used by several users. If modifications of one user affect the work of other users, the system behavior is no longer transparent to those users (e.g., Microsoft Word on PC). It must therefore be guaranteed that the modifications of one user do not affect other users. This can be accomplished by means of "login" facilities or some other suitable identification system.

The system behavior will also neither be transparent to a user if his system modifications have been carried out erroneously or wrongly. The user must therefore be provided with an overview facility showing all his modifications ("central switchboard") and a "reset" facility that sets the system in a well-defined state.

Summary of Specifications for User-tailoring

The specifications for dialog systems with user-tailoring facilities can be summarized as follows:

- For casual users, user-tailorable systems are not useful. Flexible systems that offer different possibilities to choose from are more suitable.
- For regular users of a dialog system, user-tailoring should be provided by means of commands embedded in the system or by a special tailoring module that can be called from the dialog system. Textual configuration files are, in most cases, not favorable.
- Textual configuration files can be useful for users who write their own application programs (end-user programming).
- System modifications made by the users should not negatively affect the transparency and predictability of the system behavior. This would be counter-productive.

PARTICIPATION

Participation is the criterion of user-oriented dialog design, which refers to the users' participation in the development and installation of their dialog system. In accordance with the concept of control, there should be the possibility for the user to choose and to influence. This can be managed by involving the user as an expert in his work very early in the process (Gould & Lewis, 1984). Hence, in another project of our research group, models and methods for participative software development are to be developed and evaluated empirically.

As a basis for this, software projects were analyzed. This was not done with regard to technical aspects of software development, but with regard to work organization, use of methods, user participation, and the problems connected with different procedures.

Sample, Methods, and Approach

The field studies were carried out in Swiss and German firms that develop information systems for offices and administration. The sample consisted of software firms (75%) as well as banks, insurance companies, industrial plants, etc. with internal software development departments.

The following research methods were used:

1. In one step, different documents of several firms (n=12) in the form of organigrams, handbooks for project management, etc. were analyzed.
2. The general procedure of software development was analyzed in semi-structured interviews (n=22).
3. In a questionnaire, the subjects described a concrete, representative software project, which had been finished (n=540 were asked; n=80 firms answered; return rate=15%).

One or more of the following persons of the firms were involved: director, head of software development, project leader, software ergonomics or product manager.

Results

The following passages refer to the main results of the study that concern the question of user participation (Strohm, 1990).

Intensity of User Participation. The data of the study show that willingness on the part of software engineers to involve users in software development has become greater. Statements from the subjects such as "We must involve the user" or "Without the user it doesn't work" verifies this.

Figure 7 contains the frequency of active, passive, and no participation in the analyzed projects. The classification in the active and passive form of participation was done according to Degree, Content, and Time of Participation.

Projects with active participation are characterized by the following components:

Degree. The involved users participated in determining functionality, data structures, and interface.

Content. The users realized ideas and had decision possibilities.

Time. The users were already involved in the first step of the project (problem analysis).

In projects with a passive form of participation, the users brought information and/or evaluated the ideas of the software engineers (mainly interface design). They were mainly involved in the test phases.

The Attitude Toward User Participation among Software Engineers. The following advantages and disadvantages of user participation were most frequently named by the subjects:

Advantages
1. Adequate consideration of the users' needs
2. Receiving relevant information
3. Higher acceptance
4. Lower expenditure for training

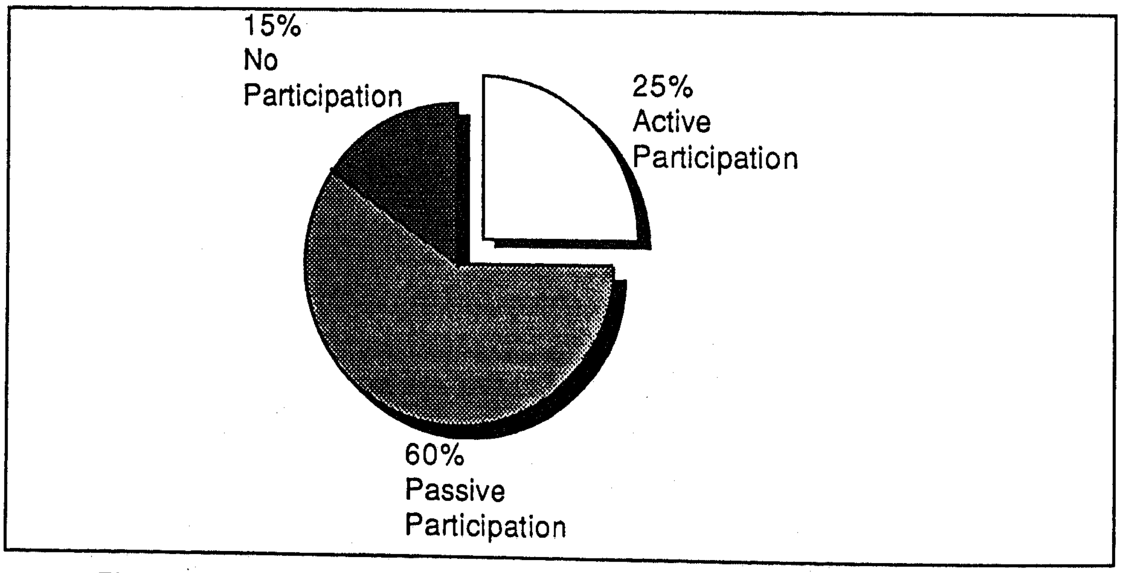


Figure 7. Frequency of active, passive, and no participation (*n*=79).

Disadvantages

1. Time consuming
2. Conflicting opinions among the users
3. Lack of the user's computery
4. Communication difficulties
5. Lack of motivation among the users

However, a quantitative evaluation of user participation resulted in some differences in dependence of intensity of participation. In projects with active participation, communication** and cooperation* with the users as well as motivation** of the users was rated better by the subjects than in projects with passive participation (**$p<0.05$, *$p<0.1$). This implies that projects with active participation go more smoothly (i.e., function more optimally).

Participation and Economic Data. The economic data of the study show that software projects are often confronted with economic risks. In the analyzed cases the costs were exceeded on an average of 45% (minimum: 0%, maximum: 400%). Figure 8 shows the exceeded costs dependent on intensity of participation.

From Figure 8 it becomes apparent that in projects with active participation the costs were exceeded to a lesser degree than in projects with passive participation and to a much lesser degree than in projects without participation. Thus, participation can also avoid economic risks in software development.

Discussion

The results of our study show that there are stronger attempts towards user participation in software development practice. Nevertheless, from a work

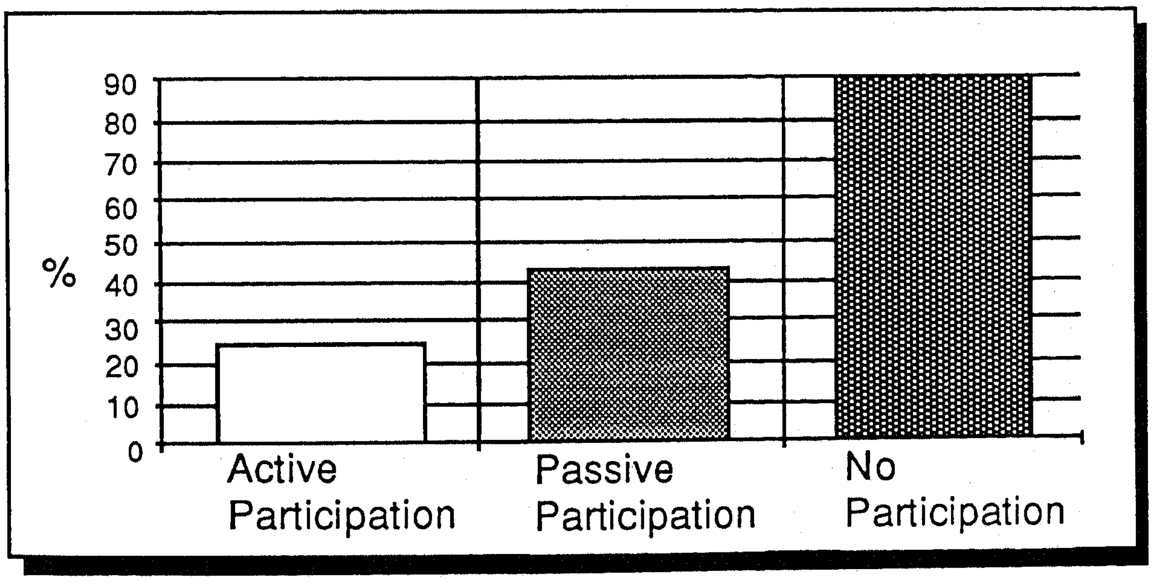


Figure 8. Exceeded costs in dependence of active, passive and no participation. (*n*=70; *p*<0.05; one-factorial analysis of variance).

psychological point of view, inadequate forms (no or passive participation) are still being practiced.

The data of the study verify the fact that active and early participation can be more successful. Projects with active participation were realized with fewer problems and economic risks.

GENERAL CONCLUSIONS

There exist serious gaps in the area of basic research on user-oriented dialog design. Future research should follow a sequential strategy, based on knowledge from general and work psychology. Task orientation and control belong to the relevant concepts, as the demands set on people are not determined exclusively or predominantly by technology, but to a high extent, by organization and work structuring. As a consequence, the predominant direction of the design process must change: Instead of following tool design, task design must come first.

REFERENCES

Altmann, A. (1987). Direkte Manipulation: empirische Befunde zum Einfluss der Benutzeroberfläche auf die Erlernbarkeit von Textsystemen. *Zeitschrift für Arbeits- und Organisationspsychologie, 31* (N.F.5) 3, 108–114.

Blumenfeld, W. (1932). Über die Fraktionierung der Arbeit und ihre Beziehung zur Theorie der Handlung. In *Bericht über den XII. Kongress der Deutschen Gesellschaft für Psychologie (pp. 291–294). Jena.*

Deutsches Institut für Normung DIN. (1988). *Grundsätze ergonomischer Dialog-gestaltung.* Berlin: Beuth.

Emery, F. (1959). *Characteristics of socio-technical systems.* Tavistock Institute of Human Relations, Document 527.

Emery, F. (1978). *Characteristics of socio-technical systems* (An abridged version of TIHR Doc. 527). In F. Emery (Ed.), *The emergence of a new paradigm of work.* Canberra: Centre for Continuing Education, Australian National University.

Greutmann, T., & Ackermann, D. (1989). Zielkonflikte bei Software-Festaltungs-kriterion. In S. Maass & H. Oberquelle (Eds.), *Software-Ergonomie '89. Aufgabenorientierte Systemgestaltung und Funktionalität* (pp. 144–152). Stuttgart: Teubner.

Greutmann, T., Ackermann, D., & Krebs, M. (1988). Easy to learn or easy to use? Consequences for the development of competence. To appear in *Proceedings of the 7th European Annual Conference on Human Decision Making and Manual Control*, Paris, October 18–20, 1988.

Gould, J.D., & Lewis, C. (1984). Designing for usability — Key principles and what designers think. In A. Janada (Ed.), *Human factors in computing systems. Proceedings of the CHI'83* (pp. 50–53). Amsterdam: Elsevier.

Hacker, W. (1986). *Arbeitspsychologie*, Bern: Huber.

Hacker, W. (1987). Software-Gestaltung als Arbeitsgestaltung. In K.-P. Fähnrich (Ed.), *Software-Ergonomic. State of the Art 5* (pp. 29–42). München: Oldenbourg.

Hellpach, W. (1922). Sozialpsychologische Analyse des betriebstechnischen Tatbestandes 'Gruppenfabrikation'. In R. Land & W. Hellpach (Eds.), *Gruppenfabrikation* (pp. 5–86). Berlin: Springer-Verlag.

Hutchins, E.L., Hollan, J.D., & Norman, D.A. (1986). Direct manipulation interfaces. In D.A. Norman & S.W. Draper (Eds.), *User centered system design* (pp. 87–124). Hillsdale, NJ: Erlbaum.

Kamoun, A., Debernard, S., & Millot, P. (1988). Implicit dynamic allocation of tasks between man and computer based on the optimal command. In *Proceedings of the 7th European Conference "On Human Decision Making and Manual Control"* (pp. 55–62).

Margono, S., & Shneiderman, B. (1987). A study of file manipulation by novices using commands vs. direct manipulation. In *Proceedings of the 26th Annual Technical Symposium of the Washington D.C. Chapter of the ACM*. Gaithersburg, MD.

Moll, T. (1987). On methods of analysis of mental models and the evaluation of interactive computer systems. In M. Frese, E. Ulich, & W. Dzida (Eds.), *Psychological issues of human-computer interaction in the work place* (pp. 403–417). Amsterdam: North Holland.

Moll, T., & Fischbacher, U. (1989). Über die Verbesserung der Benutzerunterstützung durch ein Online-Tutorial. In S. Maass & H. Oberquelle (Eds.), *Software-Ergonomics '89. Aufgabenorientierte Systemgestaltung und Funktionalität* (pp. 223–323). Stuttgart: Teubner.

Moll, T., & Sauter, R. (1987). Über den Gebrauch eines kontext-spezifischen Help-systems. In W. Schönpflug & M. Wittstock (Eds.), *Software-Ergonomie '87: Nützen Informationssysteme dem Benutzer?* (pp. 408–416). Stuttgart: Teubner.

Nievergelt, J. (1982). Errors in dialog design and how to avoid them. In *Proceedings of the 1982 International Zurich Seminar on Digital Communications* (IEEE Catalog No. 82 CH1735-0).

Rauterberg, M. (1989a). MAUS versus FUNKTIONSTASTE: Ein empirischer Vergleich einer desktop- mit einer ascii-orientierten Benutzungsoberfläche. In S. Maass & H. Oberquelle (Eds.), *Software-Ergonomie '89. Aufgabenorientierte Systemgestaltung und Funktionalität* (pp. 313–323). Stuttgart: Teubner.

Rauterberg, M. (1989b). Ein empirischer Vergleich einer desktop- mit einer menüorientierten Benutzungsoberfläche für ein relationales DBMS. In M. Paul (Ed.), *GI-19. Jahrestagung Band I. Computergestützter Arbeitsplatz* (pp. 243–258). Berlin: Springer-Verlag.

Rice, A.K. (1958). *Productivity and social organization: The Ahmedabad experiment*. London: Tavistock.

Shneiderman, B. (1987). *Designing the user interface*. Amsterdam: Addison-Wesley.

Smith, S.L., & Mosier, J.N. (1986). *Guidelines for designing user interface software* (U.S. Department of Commerce "National Technical Information Service." Report-No. AD-A177 198). Bedford: MITRE-Corporation.

Spinas, P. (1987). *Arbeitspsychologische Aspekte der Benutzerfreundlichkeit von Bildschirm systemen*. Doctoral dissertation, Bern.

Strohm, O. (1990). Benutzerorientierung und Probleme bei der Software-Entwicklung. *Output, 7*, 27–31.

Tomaszewski, T. (1981). Struktur, Funktion und Steuerungsmechanismen menschlicher Tätigkeit. In T. Tomaszewski (Ed.), *Zur Psychologie der Tätigkeit* (pp. 11–33). Berlin: Deutscher Verlag der Wissenschaften.

Troy, N. (1980). Zur Bedeutung der Stresskontrolle: Experimentelle Untersuchungen ü)ber Arbeit unter Zeitdruck. *Zeitschrift für Arbeitswissenschaft 34*, 103–108.

Troy, N. (1986). *Die Funktionsweise von 'intelligenten' Maschinen in der Vorstellung ihrer Benutzer.* Unpublished diploma thesis, University of Zürich.

Ulich, E. (1987). Some aspects of user-oriented dialogue design. In P. Docherty, K. Fuchs-Kittowski, P. Kolm, & L. Matthiassen (Eds.), *System design for human development and productivity: Participation and beyond* (pp. 33–47). Amsterdam: Elsevier.

Ulich, E. (1989). *Regarding computers as tools and the consequences for human-centered work design*. Paper presented to the 3rd International Conference on Human-Computer Interaction, Boston.

Verein Deutscher Ingenieure VDI. (1988). *Software-Ergonomie in der Bürokommunikation. Richtlinienentwurf*. Berlin: Beuth.

Volpert, W. (1974). *Handlungsstrukturanalyse als Beitrag zur Qualifikationsforschung*. Köln: Pahl-Rugenstein.

Volpert, W. (1987). Psychische Regulation von Arbeitstätigkeiten. In U. Kleinbeck & J. Rutenfranz (Eds.), *Arbeitspsychologie* (pp. 1–42). Göttingen: Hogrefe.

Whiteside, J., Jones, S., Levy, P.S., & Wixon, D. (1985). User performance with command, menu, and iconic interfaces. In L. Borman & B. Curtis (Eds.), *Human factors in computing systems-II* (pp. 185–191). Amsterdam: North-Holland.

Zapf, D., & Frese, M. (1989). Benutzerfehler im Kontext von Arbeitsaufgabe und Arbeitsorganisation. In S. Maass & H. Oberquelle (Eds.), *Software-Ergonomie '89. Aufgabenorientierte Systemgestaltung und Funktionalität* (pp. 211–222). Stuttgart: Teubner.