BEHAVIOUR & INFORMATION TECHNOLOGY, 1992, VOL. 11, NO. 6, 319–328

Some surprising differences between novice and expert errors in computerized office work

JOCHEN PRÜMPER, DIETER ZAPF, FELIX C. BRODBECK and MICHAEL FRESE

Department of Psychology, University of Giessen, Otto-Behaghel-Str. 10, D-6300 Giessen, Germany

Abstract. This paper investigates the impact of different levels of expertise on errors in human–computer interaction. In a field study 174 clerical workers from 12 different companies were observed during their normal office work and were questioned on their expertise with computers. The level of expertise was determined by (*a*) the length of time an employee had worked with a computer (computer expertise); (*b*) the number of programs s/he knew (program expertise); and (*c*) the daily time s/he spent working with the computer (daily work-time expertise). These different operationalizations of novices and experts led to different results. In contrast to widespread assumptions, experts did not make fewer errors than novices (except in knowledge errors). On the other hand, experts spent less time handling the errors than novices. A cluster analysis produced four groups in the workforce: occasional users, frequent users, beginners, and general users.

1. Introduction

In order to develop effective computer systems, it is important to gain insight into the problems which may arise for those using software on a daily basis in the workplace (e.g., Frese *et al.* 1987). Many studies have shown that the character of a problem is closely related to users' level of expertise (e.g., Chi *et al.* 1988). Consequently, it is profitable to analyse the different problems arising at work for computer users with different levels of expertise.

1.1. *Studies about expertise*

The existing literature about expertise has chosen to differentiate between novices and experts as follows: experts have greater domain knowledge (Johnson *et al.* 1981, Voss and Post 1988); experts have greater ability to organize their knowledge (Chase and Simon 1973, deGroot 1966, Soloway *et al.* 1988); experts tend to explore problems to a greater degree (Schaub and Strohschneider 1992); experts see and represent a problem in their domain at a more fundamental level (Chi *et al.* 1981); experts tend to reflect more on their own actions (Miyake and Norman 1979, Simon and Simon 1978); experts tend to apply exceptionally efficient heuristic strategies (Putz-Osterloh and Lemme 1987); and experts have more realistic ideas of potential problems (Chi *et al.* 1982). In addition, there is a small number of studies about errors in human–computer interaction which reach similar conclusions (e.g., Cuff 1980, Davis 1983, Lang *et al.* 1981, Wiedenbeck 1985, Youngs 1974). Overall, these studies have proven that experts are simply better and make less errors than novices.

Yet there seems to be one other common result: existing studies have concentrated formally on complex, often artificially-created experimental situations, for example the reconstruction of a position in a game of chess (e.g., deGroot 1965) or a computer program (e.g., Vessey 1988) or the management of simulated scenarios (e.g., Dörner 1987, Funke 1988). Obviously, these studies have shown little interest in the question of how people cope with a daily environment dominated by routine actions rather than complex or artificial problem solutions.

By analysing an empirical field study from an action-theoretical point of view, we attempt to show that experts do not necessarily reach better results: that they do not make fewer errors than novices.

1.2. *Operationalization of expertise*

Different studies use different empirical definitions of experts and novices. However, the majority of studies on novice/expert research in the field of human–computer interaction has chosen to concentrate on laboratory situations, comparing problem solutions found by professors, teachers, and experienced programmers, with those by students and less experienced programmers (e.g., Adelson 1984, Allwood and Eliasson 1987, Barfield 1986, Bateson *et al.* 1987, Cooke and Schvaneveldt 1988, Shneiderman 1976, Soloway *et al.* 1988, Vihmalo and Vihmalo 1988, Weiser and Shertz 1983). To our knowledge there are hardly any novice/expert studies concentrating on observations of people solving non-standardized problems in their common work situation.

Thus, there are usually two criteria used for the differentiation between novices and experts: degrees (e.g., comparing students and teachers) and the time spent working with a particular system (e.g., students with a few vs. those with many courses) (of course, there is a large overlap between degrees and time worked). However, a review of the literature also showed that the differentiation between novices and experts 'has been used to mean anything' (Allwood 1986: 634) and that 'there are currently no well-established methods for determining programmer expertise' (Vessey 1988: 158). In general, there is a lack of investigations in the actual workplace, as well as little use of multiple criteria to differentiate different levels of expertise. Consequently, we attempt herein to identify and analyse errors made by novices and experts when interacting with a computer during normal office work, and to discuss various criteria that determine the level of expertise. The following three criteria were considered: (*a*) the total length of time that the user had worked with computers (*computer expertise)*; (*b*) the number of programs known *(program expertise)*; and (*c*) the amount of time working with a computer *(daily work-time expertise).* Since the subjects in the present study were all experienced in their work tasks, it is assumed that all subjects have enough computer-independent task knowledge. The suggested operationalizations refer to the users' device representations (Kieras and Polson 1985). These include task-relevant knowledge (What task can be performed with the computer?), computer behaviour knowledge (How does it react?), and knowledge about the internal structures and functions of the computer system (How does it work?).

1.3. *An action-oriented error taxonomy*

To distinguish between errors made by novices and experts, it is useful to differentiate specific error classes. A taxonomy was developed for this purpose (Frese and Zapf 1991, Zapf *et al.* 1992, Zapf *et al.* 1989). For this article, the following distinctions are important: *usability problems*[1] (errors and problems that result from a mismatch between the user and the computer), *inefficiency problems* (some detour is taken), and *functionality problems* (mismatch between computer system and the task).

According to an action–theoretic approach (Hacker 1986, Semmer and Frese 1985, Volpert 1982, 1987) which assumes an hierarchic–sequential structure of goals and plans, several levels of action regulation can be differentiated. Usability problems can occur at each level of action regulation. The intellectual level of regulation (the high level) implies that situations and problems are consciously analysed, elaborate plans are monitored, and feedback is judged. Errors occur on the intellectual level of regulation because of goals and plans are inadequately developed. Because the plans are complex or because the conditions of when to use a subplan are not specified, a part of the action may not be done at the right time. Finally, there may be difficulties in interpreting feedback by the system. Errors at the intellectual level of action regulation are similar to Norman's (1981) and Reason's (1990) definition of mistakes. Lower level errors can be found at the level of flexible action patterns and at the level of sensorimotor regulation. Both levels regulate routinized and highly practised actions. The level of flexible action patterns implies that relatively reliable structures are used which can be changed flexibly. Errors at this level occur for example, when the correct action is executed in the wrong situation or some sign or signal is overlooked. At the sensorimotor level stereotypical, routinized, and automatic sequences of movements are regulated without conscious attention, e.g., typing errors or incorrect movements with the mouse. Errors at these levels correspond to the definition of action slips (Norman 1981, Reason 1979, 1990). In addition, there is the knowledge base for regulation which provides the material used to regulate actions. Knowledge errors may appear because of information deficits or misconceived information, for example, not knowing a particular command, the mean-

[1]The term *usability* is used to mean many different things (Booth 1989). We use the term in a narrower sense than, e.g., Shackel (1986), whose concept comprises effectiveness, learnability, flexiblity, and attitude, and thus includes aspects we would subsume to functionality (namely the question whether the functions of a system allow to attain a certain goal).

ing of a particular function key, etc. (for details, see Frese and Zapf 1991, Zapf *et al.* 1989, 1992).[2]

There are two causes for *inefficiency:* inefficiency due to lack of knowledge and inefficiency due to habit (Zapf *et al.* 1989). The former implies that the user follows an inefficient strategy because he or she does not know a better way. Inefficiency due to habit means that the person habitually uses inefficient routines, although he or she knows that there are more efficient ways. The difference between knowledge errors and inefficiency due to lack of knowledge is that the latter allows the goal to be achieved without any correction, however a shorter route to goal attainment is available. In contrast, knowledge errors can have three consequences: (1) the user can not achieve the goal at all; (2) the user can only achieve the goal by external support: or (3) the user has to redo an action step to reach the goal. Novices should be more inefficient than experts in the knowledge subcategory of inefficiency.

Functionality problems, as we use the term, comprise both soft and hardware problems. They imply that computer devices do not support the users' goals in an appropriate way. Functionality problems can be classified into different categories: action blockade (the user either has to give up or has to change a goal), action repetition (a part of one's work is lost and needs to be redone), action interruption (the user is interrupted by the system, but is able—usually after some additional work—to continue), and action detour (the user knows the weaknesses of the software and compensates them). We did not develop abstract suppositions regarding functionality, but were interested in determining empirically the occurrence of functionality problems with novices and experts.

1.4. *Errors of novices and experts*

The taxonomy allows us to develop hypotheses regarding different types of errors depending on the operationalization of expertise. People who know many programs (program experts) probably have a good general mental model of computers and programs. This may reduce knowledge errors. However, those experts have to deal with many incompatible programs. Since they are better acquainted with some programs than with others—and have routinized their actions—they will apply their routines incorrectly and thus make more habit errors (errors on the lower levels of regulation). Their well-practised actions which successfully apply to one system may turn out to be wrong for another one.

The other operationalization of expertise—defined as the length of time since starting to learn to use a computer (computer expertise)—may imply similar hypotheses.

Action regulation for a particular task changes with increasing experience (Frese and Altmann 1989, Frese and Stewart 1984). The more experienced somebody is, the more parts of a task he or she will carry out routinely. This implies that more actions are regulated at the lower levels. Therefore, the chance increases to commit errors these levels. Similarly, Lewis and Norman (1986) argue that the highly practised, automated behaviour of the expert leads to a lack of focused attention. This increases the likelihood of some forms of slips which correspond with errors on lower levels of regulation. Program experts regulate more actions on lower levels than novices. Thus, they should show more errors on lower levels of regulation.

Finally, expertise defined by daily length of computer use may have different implications. For people who work long hours with a special program, it is important to be efficient. For them daily inefficiency leads to a higher loss of time than for a person who just works briefly with the computer. Therefore, they might be particularly motivated to increase their knowledge and their efficiency. Thus experts (in terms of daily computer time) should be more efficient than daily work-time novices.

While we have developed different hypotheses for the different groups of novices and experts regarding to types of errors, we expect a uniform picture for error-handling time. Error-handling time (that is, the time elapsed from the detection of the error to the successful correction or abandonment of correction should be shorter for experts, regardless of how they are operationalized, since they have more experience in using the programs than novices.

2. Method

2.1. *Subjects*

Two hundred and fifty-nine users of 16 different software programs from 12 companies in the Federal Republic of Germany comprising 16 different departments and seven small firms participated in our study. The size of the companies analysed ranged from the office department of a large public administration department with 260 employees to a small brokerage firm

[2]In addition, errors can also be differentiated according to different stages of the action process (Frese and Stewart 1984, Norman 1984, 1986) which was done by us as well (Frese and Zapf 1991, Zapf *et al.* 1989, 1990). However, theory just offers us hypotheses for the levels of regulation and not for the action process. Therefore, the analysis collapsed errors according to the levels of action regulation.

operated by two family members. The array of work tasks ranged from low level data entry, text editing, graphic design, and electronic mailing, to managerial tasks. An essential component of each subject's job was computer work (average percentage of daily computer work was 50–60%). The average time of employment in the company was 3–5 years, the average experience with computers was 1–2 years, and on average 1·8 computer programs were known. The age of the subjects ranged from 16 to 60 years, average age was 31 years; 72·9% were female. The subjects were observed in the office and were asked to fill out a questionnaire. However, not all of the 259 subjects could be observed and not all those observed returned the questionnaire. Our report is based upon observation data from 198 subjects, questionnaire data from 232 subjects, and 174 subjects with both sets of data.

2.2. *Measures and procedures*

The subjects were observed, interviewed and given a standardized questionnaire which asked for demographic data, work characteristics, and their level of computer expertise.

The observers participated in a three-day training session which included a discussion of the theory and procedure, as well as an evaluation of a set of errors which we had collected previously. Observers practised for one and a half days, and observed an experienced observer before working with the subjects.

The subjects were observed during their routine work process with the computer. The observation period lasted for 2 h. The observer sat next to or behind the subject in order to see both the screen and the keyboard. A structured protocol sheet was used to record and classify all errors. Each error was described briefly. Based on these descriptions, the errors were rerated by two raters. Interrater agreement for these two raters was 74.7%. This is equivalent to Cohen's (1960) kappa coefficient of 0.73. Only those errors the reraters agreed upon were included ($n = 1306$).

Error-handling time is defined as the time it takes to correct or abandon correcting an error after it has been detected. German companies did not allow the use of stop watches for time-keeping. Therefore, error handling time was recorded with a rough observer estimate on a five-point scale: immediately (coded: 15 s), to 2 min (coded: 1 min), to 5 min (coded: 4 min), to 10 min (coded: 8 min) and more than 10 min (coded: 12 min). Actually, the 10 min and above category leads to a conservative estimate of the total handling time, because in some cases subjects actually needed more than 30 min. To assess the reliability of error-handling time, a small study with 23 subjects was carried out with two observers (Prümper 1991a). Since the ratings of error handling time were ordinal data, a Kendall's (1948) *tau* of 0·69 could be computed.

As an *a priori* classification, we operationalized different kinds of novices/experts by a cut-off point on their answers to the following questions: (*a*) 'How long ago did you start working with a computer for the very first time?' (called '*computer expertise*'). The cut-off point was one year; i.e., 'computer novices' had up to one year computer experience, 'computer experts' more than one; (*b*) 'How many computer programs do you work with?' ('*program expertise*') users who knew only one computer program were considered 'program novices'; (*c*) 'How many percentage of your working hours do you spend working with a computer?' ('*daily work-time expertise*'). Those who spent less than 50% of their time working with the computer were classified as 'daily work-time novices', the others as experts.[3]

3. Results and discussion

3.1 *Results of the* a priori *classification of novices and experts*

Table 1 shows the results for the three different novice/expert groups. Commonsense suggests that novices make more errors than experts. However, our results suggest otherwise. There were no significant differences between computer novices and experts in the total number of errors (usability problems, functionality problems, and inefficiencies). Program experts even made significantly more errors than novices. On the other hand, there was a significant higher number of errors in daily work-time novices. Invariably, this leads to two conclusions. First, commonsense, is sometimes wrong; and second, the answer to how many errors are made by experts or novices strongly depends on which criterion is used for defining experts and novices.

In line with our hypotheses, daily work-time novices made significantly more errors on the knowledge base for regulation and computer novices made significantly more errors on the knowledge base for regulation, and significantly more inefficiencies due to lack of knowledge then experts. Again, the choice of criterion was important. In contrast to computer experts and daily work-time experts, program experts made significantly more errors not only on the higher level but on lower levels of

[3]The cut-off points might seem to be a bit severe. However, additional analyses were carried out with trichotomized samples leading to similar results (Prümper 1991b).

Table 1. Average number of errors per computer hour for novices and experts concerning computer, program, and daily work-time expertise.

	Computer			Program			Daily Work-Time		
	Novices *n*=51		Experts *n*=123	Novices *n*=95		Experts *n*=79	Novices *n*=82		Experts *n*=91
Total number of errors	5·05		4·64	3·87	***	5·83	5·37	**	4·07
Usability problems									
Errors in the knowledge base of regulation	0·61	*	0·34	0·39		0·45	0·52	*	0·33
Errors on higher level of regulation	0·79		0·81	0·64	*	1·00	0·79		0·82
Errors on lower levels of regulation	2·12		1·89	1·55	***	2·44	2·05		1·88
Inefficiency									
Lack of knowledge	0·77	**	0·22	0·47		0·28	0·47		0·31
Inefficiency out of habit	0·30		0·31	0·25		0·37	0·37		0·17
Functionality problems	0·48	**	1·08	0·58	*	1·29	1·18	*	0·56

Note: ***$p<0{\cdot}001$; **$p<0{\cdot}01$; *$p<0{\cdot}05$ (one-tailed t test).

Table 2. Error handling time for novices and experts (minutes per error).

	Computer			Program			Daily Work-Time		
	Novices		Experts	Novices		Experts	Novices		Experts
Total error handling time	2·31	*	1·51	1·72		1·79	2·09	*	1·46
	n=48		*n*=111	*n*=85		*n*=74	*n*=75		*n*=83
Handling time for usability problems									
Errors in the knowledge	2·82	*	1·65	2·06		2·19	2·39		1·74
base of regulation	*n*=22		*n*=33	*n*=30		*n*=25	*n*=32		*n*=23
Errors on higher	1·98		1·65	1·64		1·87	2·34	**	1·22
level of regulation	*n*=28		*n*=62	*n*=46		*n*=44	*n*=43		*n*=47
Errors on lower	0.76	*	0.51	0.60		0.56	0.71	**	0.47
levels of regulation	*n*=41		*n*=102	*n*=73		*n*=70	*n*=66		*n*=75
Handling Time for	4·79		2·72	3·69		2·75	3·54		3·15
functionality problems	*n*=10		*n*=29	*n*=21		*n*=18	*n*=16		*n*=22

Note: Error categories that needed no error handling (inefficiency and action detours) are excluded from the analysis.
The different *n*s in the separate error categories are due to the fact that error handling time per error is the result of number of errors per person divided by error error handling time per person. In case of no error event this would mean a division by 0. In this instances the subject is excluded from the analysis.
Note: **$p<0{\cdot}01$; *$p<0{\cdot}05$ (one-tailed t test).

regulation as well. Thus, learning produced a reduction, as well as an increase of errors depending on the kind of error and the way expertise is operationalized. Learning increases the knowledge of the system and the capability to apply this knowledge to the task. This leads to a reduction of knowledge errors and errors at the intellectual level of regulation. However, by growing experience the routinized actions increase. This leads to a higher chance of routinized errors at the lower levels of action regulation. Apparently, errors in routinized actions were independent of how long one has worked with a computer in general and of how long one has worked with the computer on a daily basis. However, the number of errors is dependent on the amount of programs somebody knows. The reason for this lies in the incompatibilities between programs; routinizing the commands for one leads to more problems with the next program.

Computer and program experts had significantly more functionality problems than computer and program novices. Experts probably have to cope with problems of higher complexity and, therefore, pushed the limits of their software, producing more functionality problems, which develop as a result of the mismatch between the computer system and the actual work task. In contrast, computer and program novices performed simple tasks which were well covered by the standard features of the software. However, the results are not completely consistent because daily work-time novices had significantly more functionality problems than daily work-time experts.

Table 3. Four cluster solutions for 'computer expertise', 'program expertise', and 'daily work-time expertise'.

	Occasional Users $n=74$	Frequent Users $n=66$	Beginning Users $n=27$	General Users $n=6$
Computer Expertise	2–3 years	2–3 years	3–6 month	2–3 years
Program Expertise	1·9	1·6	1·2	5·2
Daily Work-Time Expertise	20–30%	80–90%	50–60%	40–50%

Table 2 presents quite a cohesive picture on error-handling time. Whenever there is a significant difference, the novices, regardless of operationalizations, needed more time to correct errors.

In summary, our hypotheses were partly confirmed. The overall picture concerning the number of errors depends very much on the specific operationalization of experts and novices. Computer novices made more knowledge errors and were less efficient due to lack of knowledge. A similar picture evolved for daily work-time novices. As predicted, there were more errors on lower levels for program experts. Functionality problems were higher in computer and program experts, but this was not true for daily work-time experts.

In contrast, the picture regarding error-handling time is much clearer and confirms our hypothesis. Novices always needed more time to handle errors than experts, regardless of the operationalization of novices and experts.

3.2 *Results of the cluster analysis*

Up to this point, we have been concerned with *a priori* operationalizations of novice and expert status. This does not yet tell us anything about the existence of natural groupings within the workforce. One could, for example, argue that those people who knew many programs, probably have also worked with a computer for a longer period of time and for a longer time each day than computer novices.[4] In order to find out, which empirical groupings appear at work, we established a cluster analysis with the same three variables computer expertise, program expertise and daily work-time expertise. Using the average linkage between groups method[5] the following result emerged (see table 3):

The largest group consisted of employees who had 2–3 years computer experience, knew 1·9 computer programs on average and spent 20–30% of their daily working time with the computer. They had a fairly long computer experience, their computer knowledge was relatively specific and they did not spend much work time with the computer on a daily basis. We called this group *occasional users.*

The second group also had 2–3 years experience and they knew 1·6 computer programs on average. However, they were different from the occasional users in that they spent more time working with the computer on a daily basis than any other group (80–90%). This group was called *frequent users.*

The third group knew only 1·2 computer programs on average. At the same time they spent 50–60% of their daily work time with the computer. They differed from all the other groups in that they had used the computer 3–6 months. We called this group *beginning users.*

The smallest group consisted of employees who also had 2–3 years computer experience and spent 40–50% of their daily working time with the computer. However, they knew many more computer programs than any other group (5·2 on average). This group can be called *general users.*

Generalizing this natural separation, we can define four groups of users in the computerized office. Group comparisons were made with Scheffé's Multiple Range Test ($p<0{\cdot}01$):

- The *occasional user* group is the one, which has significantly less daily work-time experience than any other group.
- The *frequent user* group is the one, which has significantly more daily work-time experience than any other group.
- The *beginning user* group is the one, which has significantly less computer experience than any other group, but significantly more daily work-time experience than the occasional user group.
- The *general user* group is the one, which has significantly more program experience than any other group.

Table 4 shows the results for occasional, frequent, beginning and general users concerning usability problems, inefficiencies and functionality problems. Group com-

[4] Actually we performed an analysis of variance to find out whether there were any interactions for the different operationalizations of novices/experts. However, there was only one significant interaction which suggests to be a chance finding.

[5] The '*average linkage between groups method*', often called UPGMA (unweighted pair-group method using arithmetic averages) was preferred, because it differs from the linkage methods in that it uses information about all pairs of distances, not just the nearest or the furthest (see Anderberg 1973).

Table 4. Analysis of variance for number of errors per computer hour for occasional-, frequent-, beginning- and general users.

	Occasional Users $n=74$	Frequent Users $n=66$	Beginning Users $n=27$	General Users $n=6$	*F* Ratio
Total number of errors	5·05	3·80	5·30	7·21	2·81*
Usability problem					
Errors in the knowledge base of regulation	0·45	0·23	0·77	0·56	3·77**
Errors on higher level of regulation	0·68	0·70	0·98	2·70	6·99***
Errors on lower levels of regulation	1·93	1·94	1·94	2·73	0·34
Inefficiency					
Lack of knowledge	0·30	0·24	1·01	0·24	3·80**
Inefficiency out of habit	0·34	0·15	0·26	0·54	0·85
Functionality problems	1·36	0·54	0·34	0·45	2·19

Note: ***$p<0·001$; **$p<0·01$; *$p<0·05$.

Table 5. Analysis of variance for error handling time per error for occasional-, frequent-, beginning- and general users (minutes per error).

	Occasional Users	Frequent Users	Beginning Users	General Users	*F* Ratio
Total error handling time	2·09 $n=67$	1·21 $b=59$	2·34 $n=26$	1·46 $n=6$	1·86
Handling time for usability problems					
Errors in the knowledge base of regulation	1·91 $n=25$	1·41 $n=14$	3·12 $n=14$	2·67 $n=2$	1·33
Errors on higher level of regulation	2·29 $n=35$	1·10 $n=33$	2·13 $n=16$	1·18 $n=6$	3·06*
Errors on lower levels of regulation	0·65 $n=59$	0·47 $nf=55$	0·74 $n=22$	0·37 $n=6$	1·49
Handling Time for functionality problems	4·43 $n=17$	2·41 $n=13$	3·53 $n=5$	0·50 $n=3$	1·19

Note: Error categories that needed no error handling (inefficiency and action detours) are excluded from the analysis. The different *ns* in the separate error categories are due to the fact that error handling time per error is the result of number of errors per person divided by error error handling time per person. In case of no error event this would mean a division by 0. In this instances the subject is excluded from the analysis.
*$p<0·05$.

parisons were made with the Duncan's Multiple Range Test ($p<0·05$).

The most interesting results were that the general users, in spite of their expert status, made the most errors and that the frequent users made the fewest errors in most cases. Beginning users had the most knowledge problems (knowledge errors and inefficiency due to knowledge) and occasional users the most functionality problems. General users made significantly more overall errors than frequent users. Concerning usability problems, beginning users made significantly more knowledge errors than frequent users. General users made significantly more errors at the higher level of regulation than the other groups. Beginning users are significantly more inefficient because of a lack of knowledge than the others.

Table 5 presents the results on error-handling time. Here, significant differences only appeared within the field of the higher level of regulation. In general, beginning users seem to need the most time to handle errors, frequent users the least by comparison.

In summary, the cluster analysis has shown three important results. First, the *a priori* classification with three classes used before is not redundant. There is not just one dimension with experts on the one hand, who use computers for many years, who know a lot of programs and work with them for many hours every day, and novices on the other hand who only have recently begun to work with computers, know one program and use it only occasionally. This means, that the category of casual users (e.g. Cuff 1980) is not just an intermediate stage between novices and experts.

Second, there was a large group of frequent users who habitually used just one program, have learned this program a long time ago, and used it quite a long time each day. They probably did not have a good general model of computers. They could actively work with one specific program very well. Because of their little general understanding, they often might had greater difficulties transferring to a second software system, as some of our qualitative observations showed.

Third, the frequent users proved to make the least amount of errors.

4. Discussion

Commonsense suggests that the overall number of errors would be higher for novices. This was not the case in our research. In fact, program experts made significantly more overall errors than the program novices. Apparently, errors *per se* are not an indication of a novice status. Thus, one has to distinguish different operationalizations of novices and experts as well as different error types.

While there were differences in the number of errors depending upon operationalization, the picture for error handling was quite clear and relatively uniform—novices showed significantly longer error-handling times than experts. They obviously had less skills to cope efficiently with errors.

The data also showed that it pays off to develop a differentiated taxonomy of errors. For example, program experts made more errors on the lower levels of regulation than program novices because experts have developed more habits that turn out to cause some trouble when inconsistent software was used. This result fits well with the cognitive literature which shows that experts have more abstract knowledge and may, therefore, commit errors in concrete situations (Adelson 1984).

Generally, computer and program experts have to deal with more functionality problems than novices. One possible reason might be that experts push the limits of the system because more problematic areas of the software are used.

From a methodological point of view the question may arise whether the date are trivial because the observers of our study got some training in understanding the background theories and the application of the error taxonomy. However, this is not very likely. The observers were trained to apply the error taxonomy but they did not know the hypotheses regarding novices and experts. They also could not always be sure whether the persons they observed were experts or novices, because these data were collected by the standardized questionnaire after the observation period.

The practical implications of these results are not only interesting, but also potentially controversial. The data of this study suggest that practice can achieve only very specific reductions in errors. Only errors in the knowledge base for regulation and inefficiencies due to lack of knowledge are reduced with longer computer experience. This gain in competence, however, is off-set by other errors.

We think that the data fit nicely with our reasoning on the concept of error management (Frese and Altmann 1989, Frese *et al.* 1991). Traditionally, the main emphasis has been to reduce the number of errors rather than to facilitate error management. Expertise does not reduce errors *per se* (except knowledge errors). So why should this be the exclusive goal? The error management strategy suggests that the goal of software design and training should not be so much to reduce the number of errors *per se*, but to reduce the negative effects of errors. One negative effect of errors is the time spent handling them. Expertise does reduce error-handling time, thus the most important aspects of error management are becoming aware of potential errors, being able to interpret errors, knowing strategies to recover from them, learning from one's errors, and developing good strategies of error diagnosis.

The suggested concept of error management is useful for all user groups, because the data showed that in many cases it is not just the pitiable novice but also the highly respectable expert who gets into trouble in the real computerized office world.

Acknowledgement

This contribution was produced as part of the research project FAUST (a German acronym of 'Error analysis for the investigation of software and training'). FAUST was supported by a grant from the Humanization of Work Fund of the Ministry of Research and Technology of the Federal Republic of Germany to M. Frese (01 HK 806 7) in collaboration with Technischer Überwachungsverein (TÜV), Munich.

References

Adelson, B. 1981, Problem-solving and the development of abstract categories in programming languages, *Memory and Cognition*, **9**, 422–433.

Adelson, B. 1984, When novices surpass experts: the difficulty of a task may increase with expertise, *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **10**, 483–495.

Allwood C. M. 1986, Novices on the computer: a review of the literature, *International Journal of Man-Machine Studies*, **25**, 633–658.

Anderberg, M. J. 1973, *Cluster Analysis for Applications* (Academic Press, New York).

Anderson, J. R. 1982, Acquisition of cognitive skill, *Psychological Review*, **89**, 369–406.

Barfield, W. 1986, Expert–novice differences for software: implications for problem solving and knowledge acquisition, *Behaviour and Information Technology*, **5**, 15–29.

Bateson, A. G., Alexander, R. A. and Murphy, M. D. 1987, Cognitive processing differences between novice and expert computer programmers, *International Journal of Man–Machine Studies*, **26**, 649–660.

Bonar, J. and Soloway, E. 1985, Pre-programming knowledge: a major source of misconceptions in novice programmers, *Human–Computer Interaction*, **1**, 133–161.

Booth, P. A. 1989, *An Introduction to Human–Computer Interaction* (Lawrence Erlbaum, Hove).

Chase, W. G. and Simon, H. A. 1973, Perception in chess, *Cognitive Psychology*, **4**, 121–152.

Chi, M. T. H., Feltovich, P. J. and Glaser, R. 1981, Categorization and representation of physics problems by experts and novices, *Cognitive Science*, **5**, 121–152.

Chi, M. T. H., Glaser, R. and Rees, E. 1982, Expertise in problem solving, in R. Sternberg (ed.), *Advances in the Psychology of Human Intelligence*, Vol. 1 (Lawrence Erlbaum, Hillsdale, NJ), 17–76.

Chi, M. T. H., Glaser, R. and Farr, M. J. 1988, *The Nature of Expertise* (Lawrence Erlbaum, Hillsdale, NJ).

Cohen, J. 1960, A coefficient of agreement for nominal scales, *Educational and Psychological Measurement*, **20**, 37–46.

Cooke, N. J. and Schvaneveldt, R. W. 1988, Effects of computer programming experience on network representations of abstract programming concepts, *International Journal of Man–Machine Studies*, **29**, 407–427.

Cuff, R. N. 1980, On casual users, *International Journal of Man–Machine Studies*, **12**, 163–187.

Davis, R. 1983, User error or computer error? Observations on a statistics package, *International Journal of Man–Machine Studies*, **19**, 359–376.

Dörner, D. 1987, On the difficulties people have in dealing with complexity, in J. Rasmussen, K. Duncan and J. Leplat (eds), *New Technology and Human Errors* (Wiley, Chichester), 97–109.

Frese, M. and Stewart, J. 1984, Skill learning as a concept in life-span developmental psychology: an action theoretic analysis, *Human Development*, **27**, 145–162.

Frese, M. and Altmann, A. 1989, The treatment of errors in learning and training, in L. Brainbridge and S. A. Ruiz Quintanilla (eds), *Developing skills with information technology* (John Wiley, New York), 65–86.

Frese, M. and Zapf, D. 1991, Fehlersystematik und Fehlerentstehung: Eine theoretische Einführung, in M. Frese and D. Zapf (eds), *Fehler bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich* (Huber, Bern), 14–31.

Frese, M., Ulich, E. and Dzida, W. 1987, *Psychological Issues of Human-Computer Interaction in the Work-place* (North-Holland, Amsterdam).

Frese, M., Irmer, C. and Prümper, J. 1991, Das Konzept Fehlermanagement: Eine Strategie des Umgangs mit Handlungsfehlern in der Mensch-Computer Interaktion, in M. Frese, Chr. Kasten, C. Skarpelis and B. Zang-Scheucher (eds) *Software für die Arbeit von morgen. Bilanzen und Perspektiven anwendungsorientierter Forschung* (Springer-Verlag, Berlin), 241–251.

Funke, J. 1988, Computer-simulated scenarios: a review of studies in the FRG, *Simulation & Games*, **19**, 277–303.

Greif, S. and Gediga, G. 1987, A critique and empirical investigation of the 'one-best-way-models' in human–computer interaction, in M. Frese, E. Ulich, and W. Dzida (eds), *Psychological Issues of Human-Computer Interaction in the Workplace* (North-Holland, Amsterdam), 357–377.

Gugerty, L. and Olson, G. M. 1986, Debugging by skilled and novice programmers, *Proceedings of the CHI '86 Conference on Human Factors in Computing Systems* (ACM, New York).

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

Johnson, P. E., Duran, A. S., Hassebrock, F., Moller, J. H., Prietula, M., Feltovich, P. J. and Swanson, D. B. 1981, Expertise and error in diagnostic reasoning, *Cognitive Science*, **5**, 135–283.

Kendall, M. G. 1948, *Rank Correlation Methods* (Griffin, London).

Kennedy, T. C. S. 1975, Some behavioural factors affecting the training of naive users of interactive computer system, *International Journal of Man–Machine Studies*, **7**, 817–834.

Kieras, D. and Polson, P. 1985, An approach to the formal analysis of user complexity, *International Journal of Man–Machine Studies*, **22**, 365–394.

Lang, T., Lang, K. and Auld, R. 1981, A longitudinal study of computer-user behaviour in a batch environment, *International Journal of Man–Machine Studies*, **14**, 251–268.

Larkin, J. H. 1983, The role of problem representation in physics, in D. Genter and A. L. Stevens (eds), *Mental Models* (Lawrence Erlbaum, Hillsdale, NJ), 75–98.

Lewis, C. and Norman, D. A. 1986, Designing for error, in D. A. Norman and S. W. Draper (eds), *User-Centered System Design* (Lawrence Erlbaum, Hillsdale, NJ), 411–432.

Miller, L. A., 1974, Programming by non-programmers, *International Journal of Man–Machine Studies*, **6**, 237–260.

Miyake, N. and Norman, D. A. 1979, To ask a question one must know enough to know what is known, *Journal of Verbal Learning and Verbal Behavior*, **18**, 357–364.

Norman, D. A. 1981, Categorization of action slips, *Psychological Review*, **88**, 1–15.

Norman, D. 1984, Stages and levels in human–computer interaction, *International Journal of Man–Machine Studies*, **21**, 365–375.

Norman, D. 1986, Cognitive engineering, in D. A. Norman and S. W. Draper (eds), *User-Centered System Design* (Lawrence Erlbaum, Hillsdale, New Jersey), 31–61.

Prümper, J. 1991a, Die Inter-Rater-Reliabilität von Fehlerbeobachtungen im Feld, in M. Fese and D. Zapf (eds), *Fehler bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich* (Huber, Bern), 47–59.

Prümper, J. 1991b, Handlungsfehler und Expertise, in M. Frese and D. Zapf (eds), *Fehler bei der Arbeit mit dem Computer: Ergebnisse von Beobachtungen und Befragungen im Bürobereich* (Huber, Bern), 118–130.

Putz-Osterloh, W. and Lemme, M. 1987, Knowledge and its intelligent application to problem solving, *The German Journal of Psychology*, **11**, 286–303.

Rafaeli, A. and Sutton, R. I. 1986, Word processing technology and perceptions of control among clerical workers, *Behaviour and Information Technology*, **5**, 31–37.

Reason, J. T. 1979, Actions not as planned: the price of automation, in G. Underwood and R. Stevens (eds), *Aspects of Consciousness* Vol. 1 (Academic Press, London), 76–89.

Reason, J. 1990, *Human Error* (Cambridge University Press).

Schaub, H. and Strohschneider, S. 1992, Die Auswirkungen unterschiedlicher Problemlöseerfahrung auf den Umgang mit einem unbekannten komplexen Problem, *Zeitschrift für Arbeits- und Organisationspsychologie,* **36,** 117–126.

Semmer, N. and Frese, M. 1985, Action theory in clinical psychology, in M. Frese and J. Sabini (eds), *Goal-Directed Behavior: The Concept of Action in Psychology* (Lawrence Erlbaum, Hillsdale, New Jersey), 296–310.

Shackel, B. 1986, Ergonomics and the design for usability, in M. D. Harrison and A. F. Monk (eds), *People and Computers: Designing for Usability, Proceedings of the Second Conference of the BCS HCI Specialist Group* (Cambridge University Press, Cambridge), 44–64.

Shneiderman, B. 1976, Exploratory experiments in programming behavior, *International Journal of Computer and Information Sciences,* **5,** 123–143.

Simon, D. P. and Simon, H. A. 1978, Individual differences in solving physics problems, in R. Siegler (ed.), *Children's Thinking: What Develops?* (Lawrence Erlbaum, Hillsdale, New Jersey), 325–348.

Soloway, E., Adelson, B. and Ehrlich, K. 1988, Knowledge and processes in the comprehension of computer programs, in M. T. H. Chi, R. Glaser and M. J. Farr (eds), *The Nature of Expertise* (Lawrence Erlbaum, Hillsdale, New Jersey), 129–152.

Spohrer, J. C., Soloway, E. and Pope, E. 1985, Where the bugs are, *Proceedings of the CHI '85 Conference on Human Factors in Computing Systems* (San Francisco, ACM), 47–53.

Vessey, I. 1988, Expert-novice knowledge organization: an empirical investigation using computer program recall, *Behaviour and Information Technology,* **7,** 153–171.

Vihmalo, A. and Vihmalo, M. 1988, Utilization of subject's background knowledge in computer program comprehension, *Zeitschrift für Psychologie,* **196,** 401–413.

Volpert, W. 1982, The model of the hierarchical-sequential organization of action, in W. Hacker, W. Volpert and M. Cranach (eds), *Cognitive and Motivational Aspects of Action* (Deutscher Verlag der Wissenschaften, Berlin), 35–51.

Volpert, W. 1987, Psychische Regulation von Arbeitstätigkeiten, in U. Kleinbeck & J. Rutenfranz (eds), *Arbeitspsychologie. Enzyklopädie der Psychologie, Themenbereich D, Serie III, Band 1* (Hogrefe, Göttingen), 1–42.

Voss, J. F. and Post, T. A. 1988, On the solving of ill-structured problems, in M. T. H. Chi, R. Glaser and M. J. Farr (eds), *The Nature of Expertise* (Lawrence Erlbaum, Hillsdale, New Jersey), 261–285.

Weiser, M. and Shertz, J. 1983, Programming problem representation in novice and expert programmers, *International Journal of Man-Machine Studies,* **19,** 391–398.

Wiedenbeck, S. 1985, Novice-expert differences in programming skills, *International Journal of Man-Machine Studies,* **23,** 383–390.

Youngs, E. A. 1974, Human errors in programming, *International Journal of Man-Machine Studies,* **6,** 361–376.

Zapf, D., Brodbeck, F. C. and Prümper, J. 1989, Handlungsorientierte Fehlertaxonomie in der Mensch-Computer Interaktion, *Zeitschrift für Arbeits- und Organisationspsychologie,* **33,** 178–187.

Zapf, D., Brodbeck, F. C., Frese, M., Peters, M. and Prümper, J. 1992, Errors in working with office computers: a first validation of a taxonomy for observed errors in a field setting, *International Journal of Human-Computer Interaction,* **4,** 311–339.