Error Management or Error Prevention: Two Strategies to Deal with Errors in Software Design

Michael Frese

Justus Liebig University, Dept. of Psychology,
Otto-Behaghel-Str. 10, 6300 Giessen, Germany

1. Introduction

There are two strategies that can be used to deal with the error problem in system design: a) Error prevention: good design reduces the amount of errors made by the users; b) error management: Good design does not nessarily change the number of errors but decreases the negative consequences of errors. Traditionally, most software-designers are mainly concerned with error prevention. In this contribution I want to argue that this strategy is okay but should be complemented by the strategy of error management.

2. Observational studies of user errors when working with computers: Evidence for error management

In project FAUST(1) 198 white collar workers were observed while working with a computer and 232 were interviewed with regard to errors. In all, 1749 errors were observed and categorized in a taxonomy of errors (details in Zapf, Brodbeck, Frese, Peters & Pruemper, 1990; Frese & Zapf, 1991). It is the first major project that has empirically looked at errors in human computer interaction within the natural work situation. The following results speak for adding the strategy of error management to error prevention in system design:

1) Errors are an important economic factor because according to the (conservative) estimate based on our observations about 12% of the computer work time is spent handling errors. However, not every error leads to the same economic consequences. Many errors are corrected within a few seconds, others can take many hours and even much longer. (One case was reported in which a person had worked for a whole year before he found out that he had taken the wrong approach.) For economic reasons it may be more useful to concentrate on errors with long handling times than on those with short ones.

2) Errors can contribute to stress at work. Our observations showed that long error handling time is the crucial variable that leads to stress-reactions (more in Brodbeck, Zapf, Pruemper & Frese, 1990). Thus, one should concentrate on errors with long error handling times. Error management is well suited to reduce this time.

3) One would suspect that novices make more errors than experts. It is, of course, quite difficult to differentiate between novices and experts in the field. One differentiation is certainly, whether a person knows just one or more than one system. Our study shows that experts (knowing more than one system) commit actually more errors - namely 5.8 errors per computer hour - than novices (knowing just one system), who made just 3.9 errors per computer hour (Pruemper, Zapf, Brodbeck & Frese, in press).

This means that even higher qualifications do not help to prevent errors - errors will always appear. Moreover, it may not even be good to minimize errors. If experts do not minimize errors, why should the designer try to do it? The problem of errors may not be their occurrence per se but their psychological and economic costs; thus, it is more important to reduce error handling time than the number of errors per se. Experts are indeed often better in their error handling time than novices (for details Pruemper et al., in press).

4) Errors that occur in problem solving tasks involving conscious strategies (errors on the so-called intellectual level of regulation) require particularly long error handling times of around 1.5 min. (This contrasts with errors in routine tasks which do not take more than 1 min. to handle, for details cf. Brodbeck et al., 1990.) Errors in problem solving tasks usually develop as a result of a false application of the program to a particular work task. Since programs very seldomly are designed for one particular work task and are, therefore, usually not completely adjusted to a particular work task, it is difficult to avoid errors of this kind. Thus, there are clear limits for using error prevention strategies for this type of errors.

5) Errors must be detected. Which type of information is useful to detect errors? In our research, the most important source of information for error detection was the so-called internal goal-comparison. In such a case, only the working person can detect the error because he or she compares the work result with his or her goal. For example, printing a particular letter cannot be detected by the machine as being an error. Only the person knows that he or she actually wanted to reformat the letter before printing it. Roughly 50% of all error cases were detected by this internal goal-comparison (Zapf, Lang & Wittmann, 1991).

This result is quite important because it suggests that there are limits to automatic kinds of error detection devices. Since a program usually does not "know" anything about the particular goals of the user, these errors cannot be

---

1 This contribution is part of a research project FAUST (a German acronym for "error analysis for the investigation of software and training"). The project was supported by a grant from the Work and Technology Fund of the Ministry of Research and Technology of the Federal Republic of Germany to M. Frese (01 HK 806 7) in collaboration with the Technischer Ueberwachungsverein Munich). An earlier longer version of this paper has appeared in German in Frese, Kasten, Skarpelis & Zang-Scheucher, 1991).

detected by the machine. Since error detection is often one prerequisite for the strategy of error prevention, there are limits to error prevention.

3. The concept of error management

Error management has to be distinguished from error handling. While error handling is a descriptive term and implies any type of response towards the error, error management is prescriptive and means a useful approach to the error with the goal of reducing future errors, of avoiding negative error consequences and of dealing quickly with error consequences once they occur. Error management can be more or less well supported by a computer system and it can be more or less well taught in training.

One prerequisite of error management is the differentiation between the error itself and the error consequences. Not every tripping leads to falling down. Negative error consequences can be, for example, not to notice an error, to accumulate time lost due to the error, the negative emotions following an error, the loss of resources or incorrigible damages, and additional errors appearing because of the confusion resulting from a first error. Error management does not mean to avoid the error per se but attempts to avoid the negative error consequences altogether or to reduce the amount of the negative error effects.

4. How to make error management possible in system design

Error management can be discussed from three angles: the error process, the action process, and the levels of regulation. In the following a few examples will be given (more details in Zapf, Brodbeck & Pruemper, 1989; Zapf, Frese, Irmer & Brodbeck, 1991).

4.1. The error process

The error process consists of the following components (which partly overlap, cf. Zapf, Lang & Wittmann, 1991):
a) *error detection*, i.e. the knowledge that an error has appeared; b) *error explanation*, i.e. explaining how the error came about; c) error handling, i.e. the actions for dealing with the consequences of the errors or performing actions compensating for the error.

Ad a) *Error detection*. Error detection is one prerequisite of error management. Since error detection becomes more difficult the further one is removed from the original error, it is useful to support early error detection. Two conditions help here:
- Clear feedback; this does not only imply error messages but also feedback on system conditions.
- Transparent design of the system; this means that the user can develop a good mental model of what the designer had intended to do.

Ad b) *Error explanation*. Error explanation means two things: First, the user knows how the error came about (e.g., a wrong combination of keys were pressed). Second, the user knows why the error has occurred (e.g., a particular icon does not comform to the desktop metaphor of a system, cf. Frese & Altmann,1989).

It is not necessary to explain every error. Good support of error management often means that one can correct an error without needing to explain it (e.g. by using an UNDO command or a back-up file). Routine corrections also do not require an error explanation. Error explanation is made easier, if the system is transparent and if there is good feedback. Additionally:
- Make it easy to develop hypotheses; hypotheses can be suggested by the system, e.g. by context specific help.
- Give possibilities to explore; it is often necessary to redo the error before one can learn from it. Therefore, exploration is useful (cf. Greif & Keller, 1990). Some systems give "history"-functions that allow to undo whole action strings. They allow and encourage exploration.
- Error messages and helps; they can directly explain how an error came about and what one can learn from it.

Ad c) *Error handling*. Error handling can be done in terms of forward or backward corrections. Forward correction allows, for example, to start a search process from the back of the text. Backward correction implies that one has to rewrite a text that was lost because of the error.

Error management in error handling is supported by:
- Orientation; one should know where one is and in which direction one should go. Orientation is easier if the system allows one to go to some fixed point from any part of the system (e.g. one can jump into the main menu from any submenu) or when error messages suggest a certain strategy of error correction.
- Direct correction of operations; the UNDO function or multiple UNDOs are examples.
- Support of forward correction; flexible system use allows this instead of a rigid hierarchy of menues and submenues.
- Interruption of actions; error management is supported if one can interrupt to do other things, e.g. by using windows, by being able to correct a sentence structure in the middle of a proof read program, etc.

4.2. The action process

The action process can conveniently be divided into goals, information intake and integration, planning, monitoring while executing the action, and feedback. The same steps were used in our taxonomy of errors (Zapf et al., 1989).

In the area of *goal development and decision*, error management is supported when it is easy to get to know goal conflicts, and to develop clearer goals and subgoals. Since computer systems are not well equipped to reflect goals, this issue is not of great importance for software development.

Error management in the area of *information intake and integration* is supported when there are good and consistent analogies and metaphors available. This leads to a better mental model that produces better error explanation and error correction strategies. Moreover, systems should give error information in compact and in extended forms. In case a user does not understand the compact form, he or she can turn to a longer explanation while the expert just needs a short reminder of how to proceed.

Plan development and decisions are supported by transparency and consistency of the system. Consistency follows the principle of least surprise. Lack of transparency and consistency make it hard to plan and replan. A sort of "playground"-function allows to develop new plans and to try them out before one works seriously with a plan.

Memory processes are particularly important in *Monitoring*. Help functions, menues, file managers, search function (particularly those that allow to search for specific texts across different files), etc. are useful to support memory. Since interruptions lead to memory errors, windows help to reduce these errors. Windows allow the user to keep in mind, what he or she was working on at the point of interruption. The function of *feedback* was already discussed and will, therefore, not be repeated here.

4.3. Levels of regulation

The levels of regulation is one dimension in our taxonomy of errors (Frese & Zapf, 1991; Zapf et al., 1989). The upper level of regulation implies that conscious processing of information and planning takes place, the lower levels are more routinized and therefore not conscious. When something is regulated at the lower level of regulation, one's "head" is free to think and plan other things.

Errors on the upper level of regulation are more difficult to detect and to explain. Those parameters of the system that allow a good comparison of the system conditions with the goal are good for error management, e.g. the What-You-See-Is-What-You-Get principle.

Correction of errors on the upper level of regulation is often difficult because complex thought processes are going on anyhow and the additional work of dealing with the error may be too much for the limited processing capacity. This means that unspecific methods of error handling, like the UNDO function are particularly useful here, since these unspecific methods can be used routinely and do not add strain on the conscious processing capacity.

There are also problems when dealing with errors on the lower level of regulation. Here feedback is not attended to as closely as when working in a conscious mode. Therefore it is difficult to detect errors (error correction is often quite easy at this level). It is, for example, well known how difficult it is to detect "typos." If the potential negative consequences are very high, action feedback should be quite salient in those cases. For example, when the loss of data can be the result of an error, blinking and big signs may be useful. Since feedback is often recognized very late, multiple UNDO possibilities are useful for errors on lower levels.

5. Conclusion

There are good theoretical and empirical reasons that support the notion of error management (cf. Brown & Newman's, 1985, notion of management of trouble). We think that this concept should be more systematically explored and used in software development. Moreover, there are many applications of wide importance of this concept, e.g. errors in software development, in accident research and in management decisions.

6. References

Brodbeck, P.C., Zapf, D., Prümper, J., & Frese, M. (1990). Error handling in office work with computers: A field study. München: Manuscript submitted for publication.

Brown, J.S., & Newman, S.E. (1985). Issues in cognitive and social ergonomics: From our house to Bauhaus. Human Computer Interaction, 1, 359-391.

Frese, M., Brodbeck, F.C., Zapf, D., & Prümper, J. (1990). The effects of task structure and social support on users' errors and error handling. In D. Diaper et al. (Eds.), Human-Computer Interaction - INTERACT '90 (pp. 35-41). Amsterdam: Elsevier.

Frese, M., & Altmann, A. (1989). The treatment of errors in learning and training. In L. Bainbridge & S.A. Ruiz Quintanilla (Eds.), Developing skills with information technology (pp. 65-86). John Wiley & Sons.

Frese, M., & Zapf, D. (1991). Fehlersystematik und Fehlerentstehung. In M. Frese & D. Zapf (Eds.), Fehler und Schwierigkeiten bei der Arbeit mit dem Computer. Ergebnisse und Beobachtungen und Befragungen im Bürobereich. Bern: Huber.

Frese, M., & Zapf, D. (Eds.) (1991). Fehler und Schwierigkeiten bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich. Bern: Huber.

Greif, S., & Keller, H. (1990). Innovation and the design of work and learning environments: The concept of exploration in human-computer interaction. In M.A. West & J.L. Farr (Eds.), Innovation and creativity at work (pp. 231-249). New York: Wiley.

Prümper, J., Zapf, D., Brodbeck, F.C., & Frese, M. (1990). Errors of novices and experts: Some surprising differences in computerized office work. München: manuscript submitted for publication.

Zapf, D., Brodbeck, F.C., Frese, M., Peters, H., & Prümper, J. (1990). Errors in working with computers: A first validation of a taxonomy for observed errors in a field setting. In J.

Ziegler (Ed.), GI Ergonomie und Informatik. Mitteilungen des Fachausschusses 2.3 "Ergonomie in der Informatik", 9, 3-26.

Zapf, D., Brodbeck, F.C., & Prümper, J. (1989). Handlungsorientierte Fehlertaxonomie in der Mensch-Computer Interaktion. Theoretische Überlegungen und eine erste Überprüfung im Rahmen einer Expertenbefragung. Zeitschrift für Arbeits- und Organisationspsychologie, 33, 178-187.

Zapf, D., Brodbeck, F.C., Prümper, J., & Peters, H. (1991). Softwaregestaltung und Fehlermanagement. Göttingen: Verlag für Angewandte Psychologie, Hogrefe.

Zapf, D., Frese, M., Irmer, C., & Brodbeck, F.C. (1991). Konsequenzen für die Softwaregestaltung. In M. Frese & D. Zapf (Eds.), Fehler und Schwierigkeiten bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich. Bern: Huber.

Zapf, D., Lang, T., & Wittmann, A. (1991). Der Fehlerprozeß. In M. Frese & D. Zapf (Eds.), Fehler und Schwierigkeiten bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich. Bern: Huber.