# Das Konzept Fehlermanagement: Eine Strategie des Umgangs mit Handlungsfehlern in der Mensch-Computer Interaktion

Michael Frese, Caren Irmer & Jochen Prümper
Institut für Psychologie, Ludwig-Maximilians-Universität München, Leopoldstr. 13, 8000 München 40

**Zusammenfassung**
Für die Software-Gestaltung wird argumentiert, daß das Konzept der Fehlervermeidung durch das Konzept des Fehlermanagements ergänzt werden soll. Das heißt, zwischen dem Fehler und den Fehlerkonsequenzen muß differenziert werden - nur die negativen Fehlerkonsequenzen sind zu vermeiden, aber nicht notwendigerweise der Fehler selbst. Die Begründungen dafür liegen in denen im Feld erhobenen Ergebnissen des Projekts FAUST. Entsprechende Vorschläge für ein Fehlermanagement werden erläutert.

**Abstract**
It is argued that the concept of error prevention should be complemented with the concept of error management. This means, there should be a differentiation between making an error and the error consequences - only the negative consequences should be prevented, but not necessarily the error itself. The reasoning for this lies in the results of project FAUST's field study. Appropriate suggestions for error management are given.

# 1 Einleitung

Das Projekt FAUST[1] (Fehleranalyse zur Untersuchung von Software und Training), das sich mit dem Fehlerproblem in der Mensch-Computer Interaktion aus der Benutzerperspektive beschäftigt, hat unter anderem beobachtet, wieviele und welche Fehler Angestellten bei ihrer täglichen Arbeit unterlaufen, wie sie mit Fehlern umgehen, welche Hilfen sie benutzen und welche Zeit diese jeweiligen Hilfen beanspruchen. Dabei wurde untersucht, ob organisationale und Streß-Bedingungen einen Einfluß auf das Fehlergeschehen haben. Natürlich wurde auch die Frage gestellt, wie man die Behandlung von Fehlern sowohl unter dem

[1] Der vorliegende Beitrag entstand im Rahmen des Forschungsprojekts FAUST. Das Projekt wurde vom Bundesministerium für Forschung und Technologie (Arbeit und Technik) gefördert (Förderkennzeichen: 01 HK 8067). Projektmitglieder: F. Brodbeck, M. Frese (Projektleitung), C. Irmer, H. Peters (TÜV Bayern), J. Prümper und D. Zapf. Die Verantwortung für den Inhalt dieses Beitrags liegt bei den Autoren.

Gesichtspunkt der Systemgestaltung als auch dem des Trainings positiv beeinflussen kann. Die Ergebnisse dieser Untersuchungen sind ausgesprochen umfangreich und können im Rahmen eines Beitrags natürlich nicht ausführlich beschrieben werden (vgl. dazu Frese & Zapf, 1991 und Zapf, Brodbeck, Prümper & Peters, 1991).

In diesem Beitrag sollen nun die Ergebnisse und unsere Überlegungen zum Fehlerproblem unter dem Gesichtspunkt des Fehlermanagements betrachtet werden. Im Prinzip kann man Fehlermanagement immer unter dem Gesichtspunkt des Trainings oder der Software-Entwicklung betrachten. Wir konzentrieren uns hier auf die Software-Entwicklung.

# 2 Warum Fehlermanagement?

Um mit dem Fehlerproblem umzugehen gibt es zwei Strategien: *a) Fehlervermeidung*: Durch gutes Design wird die Anzahl der Fehler reduziert; *b) Fehlermanagement*: Durch gutes Design werden die negativen Auswirkungen von Fehlern reduziert. Traditionell bemühen sich die meisten Software-Designer, Systeme so zu gestalten, daß dem Benutzer möglichst wenig Handlungsfehler unterlaufen. Diese Strategie hat durchaus ihre Berechtigung; wo sich Fehler leicht vermeiden lassen, sollten sie auch vermieden werden.

Allerdings sprechen die Ergebnisse unserer Untersuchungen gegen eine ausschließliche Strategie der Fehlervermeidung[2]:

1) Fehler stellen natürlich einen ökonomischen Faktor dar und unsere (recht konservativen) Schätzungen aufgrund unserer Beobachtungen besagen, daß etwa 12% der Arbeitszeit am Computer mit Fehlern und Fehlerbehandlung verbracht werden - ein wichtiger Wirtschaftsfaktor. Aber nicht jeder Fehler hat dieselben ökonomischen Konsequenzen. Einige Fehler werden bereits nach wenigen Sekunden, andere erst nach Stunden korrigiert (z.T. wurden Fehler berichtet, aufgrund derer bis zu einem Jahr umsonst gearbeitet wurde). Das heißt, aus ökonomischen Gründen sollte man sich auf solche Fehler konzentrieren, die besonders lange Korrekturzeiten benötigen.

2) Vor allem bei Wissensfehlern und Fehlern auf der sogenannten intellektuellen Regulationsebene (also bei Fehlern, bei denen die Aufgabe denkerisch nicht richtig bewältigt wurde), entstehen besonders lange Korrekturzeiten. Nach unseren Berechnungen dauert die Korrektur bei Wissensfehlern im Durchschnitt etwas mehr als 2 Minuten, bei Fehlern auf der intellektuellen Regulationsebene mehr als

[2] Die Untersuchungsergebnisse basieren auf unseren Beobachtungen im Felde. Es wurden 198 Angestellte aus 12 Betrieben (einer bestand aus Kleinstbetrieben) 2 Stunden lang bei ihrer alltäglichen Arbeit mit dem Computer beobachtet (16 unterschiedliche Computerprogramme). Im wesentlichen nahmen Sekretärinnen, Sachbearbeiter und untere Manager an der Untersuchung teil. Dabei wurden alle Fehler mitprotokolliert und in eine Taxonomie eingestuft (Details bei Zapf, Brodbeck, Frese, Peters & Prümper, 1990).

1 1/2 Minuten und im Gegensatz dazu bei Routinehandlungen weniger als 1 Minute (näheres bei Brodbeck, Zapf, Prümper & Frese, 1990). Fehlervermeidungsstrategien wirken nun bei Wissensfehlern und auf der intellektuellen Ebene nur beschränkt, weil es hier um die Verbindung zwischen der spezifischen Aufgabe und dem Programm geht. Also um die Frage: Wie benutze ich das Programm bei dieser spezifischen Aufgabe? Da Programme im Regelfall nicht im einzelnen an die Aufgabe angepaßt werden können, sind Fehlervermeidungsstrategien dort enge Grenzen gesetzt.

3) Fehler können auch Streß erzeugen. Dabei stellt sich allerdings die Frage, unter welchen Bedingungen Fehler Streß verursachen. In unseren Beobachtungen zeigt sich, daß wiederum solche Fehler, die eine lange Korrekturzeit benötigen, Streß hervorrufen (genaueres bei Brodbeck et al., 1990). Auch dieses Datum spricht also für die Konzentration auf Fehler mit langer Korrekturzeit (vgl. Punkt 2 oben).

4) Fehler müssen entdeckt werden, und es ist eine zentrale Frage, aufgrund welcher Informationen dies geschieht. Nach unseren Untersuchungen war die bei weitem wichtigste Informationsquelle der Fehlerentdeckung der sogenannte interne Zielvergleich (in 50% der Fälle; vgl. Zapf, Lang & Wittmann, 1991). Hier kann nur die arbeitende Person selbst aufgrund ihrer Zielkonzeption feststellen, ob ein Fehler vorliegt. Von einem Programm kann z.B. nicht entdeckt werden, ob ein Benutzer aus Versehen eine unnötige Druckanweisung gegeben hat. Nur die arbeitende Person selbst weiß, daß sie eigentlich noch eine Neuformatierung machen wollte und deshalb der Druckbefehl fehlerhaft war.

Dieses Ergebnis ist wesentlich für unsere Fragestellung, weil sie aufzeigt, welche Grenzen einer automatisierten Fehlerentdeckung gesetzt sind - im Regelfall lassen sich Fehlerentdeckungen aufgrund eines internen Zielvergleichs nicht automatisieren, weil die Software die Zielvorstellungen der arbeitenden Person nicht abbilden kann (außer bei sehr taylorisierten Arbeitsplätzen, siehe unten). Da automatisierte Fehlerentdeckung aber oftmals eine Voraussetzung für die Strategie der Fehlervermeidung darstellt, sind eben auch der Fehlervermeidung enge Grenzen gesetzt.

Dies gilt im übrigen auch für das Konzept der Fehlerrobustheit, das in den DIN-Normen zu den Dialogeigenschaften (DIN, 1988) eingeführt wurde. Denn Fehlerrobustheit hat ja ebenfalls zur Voraussetzung, daß die Maschine (oder das System) den Fehler entdecken muß (also antizipieren muß), um ihn dann zu ignorieren.

5) Man würde vermuten, daß Experten weniger Fehler als Novizen unterlaufen. Es ist zwar nicht ganz einfach in einer Felduntersuchung zwischen Experten und Novizen zu differenzieren (da hier alle bis zu einem bestimmten Grad mit ihrem Programm vertraut sind), aber eine sinnvolle Unterscheidung ist sicherlich die, ob eine Person nur ein Programm beherrscht oder mehrere. Hier zeigt sich nun überraschenderweise (vgl. Prümper, Zapf, Brodbeck & Frese, 1990), daß diejenigen, die mehr als ein Programm beherrschen (die Experten), mit 5.8 Fehlern pro Computerarbeitsstunde signifikant *mehr* Fehler machen als diejenigen, die nur ein Programm beherrschen (3.9 Fehler bei diesen Novizen).

Das bedeutet, daß selbst höhere Qualifikationen nicht vor Fehlern schützen - Fehler werden immer auftreten. Darüber hinaus ist es möglicherweise gar nicht

wünschenswert, die Anzahl der Fehler per se zu minimieren. Wenn es die Experten nicht tun, warum sollten es die Software-Entwickler anstreben? Was es anzustreben gilt, ist die Kosten der Fehler zu minimieren. In unserem Fall ist das die Fehlerbehandlungszeit: Hier haben die Experten nun keine "Nachteile" mehr gegenüber den Novizen - wenn man andere Operationalisierungen von Expertise verwendet, dann benötigen die Experten sogar eine signifikant geringere Fehlerbearbeitungszeit (für Details, vgl. Prümper et al., 1990).

Bis zu diesem Punkt lassen sich also die Ergebnisse des Projekts FAUST so zusammenfassen: Es ist sinnvoll, zwischen Fehlern an sich und Fehlerkonsequenzen zu unterscheiden. Wenn es um Fehlerkonsequenzen geht, dann sind Fehler mit hoher Fehlerkorrekturzeit in der Mensch-Computer Interaktion ein besonderes Problem. Diese Konsequenzen gilt es zu minimieren - das genau versucht das Konzept des Fehlermanagements. Neben diesen eher empirischen Ergebnissen gibt es zusätzlich noch die folgenden theoretischen Gesichtspunkte, die für Fehlermanagement sprechen:

6) Bei komplexen Tätigkeiten entstehen Fehler oft wegen mangelnder Qualifikation. Deshalb wird häufig der Versuch unternommen, durch verstärkte Arbeitsteilung, d.h. weniger komplexen Tätigkeiten, Fehler zu vermeiden. Die Qualifikationsanforderungen sind dadurch auch reduziert. Wir vermuten, daß eine Fehlerreduktion dadurch nicht unbedingt funktioniert. Personen, die geringere Qualifikationsanforderungen bewältigen müssen, machen zwar wahrscheinlich weniger Fehler (Frese, Brodbeck, Zapf & Prümper, 1990) - es dürften sich aber Formen der organisationalen Rigidisierung einschleichen, die dann zu organisationaler Ineffizienz und zu einer verringerten Fehlererkennung auf der organisationalen Ebene führen. Darüber hinaus vermuten wir, daß Fehler auf den oberen Regulationsebenen dann zwar nicht mehr von den Experten vor Ort, aber von den Arbeitsvorbereitern und den Vorgesetzten gemacht werden. Diese Fehler haben aber den zusätzlichen Nachteil, daß man sie nur schwer erkennen kann, weil diese Personen auch von Rückmeldungen (die wiederum die Person vor Ort erhält) weiter entfernt sind.

7) Fehler lassen sich nicht vermeiden. Da letztlich Menschen die Arbeitsprozesse determinieren - selbst wenn dieser Einfluß nicht immer unmittelbar ist - werden stets menschliche Fehler auftauchen (vgl. Perrows, 1988, Fallstudien, in denen die fast unendlich große Menge an Fehlermöglichkeiten gezeigt wird - auch in sehr genau durchgeplanten Systemen, wie Atomkraftwerken). Zwar ist es einerseits genau der Vorteil des menschlichen Denk- und Handlungsapparats, daß er sehr schnell eine Menge Informationen integrieren und auch unter Ungewißheit handeln kann, aufgrund von Erfahrung schnell stereotype Routinen herausbildet und selbst im Zweifelsfall noch "handlungsfähig" bleibt; andererseits führen aber genau diese Vorteile unter bestimmten Bedingungen zu Fehlern.

Auch wenn es durchaus eine sinnvolle Strategie ist, Fehler zu vermeiden, verringert sich der Grenznutzen zunehmend. Während bei völlig neuen Systemen anfangs noch jeder Schritt der Fehlervermeidung wirkliche Fehlervermeidung produziert, kann bei sophistizierteren Systemen eine Fehlervermeidung zur Fehlerverlagerung mit möglicherweise schlimmeren Konsequenzen führen. Am Beispiel der technischen Automatisierung, bei der Fehler durch technische Appa-

rate verhindert werden sollen und damit zunehmend den Menschen von der direkten Handlung entfernen, läßt sich dies veranschaulichen. Fehlervermeidung ist immer verbunden mit einer Verringerung der aktiven Handlungen und damit mit einer Atrophie des Handlungswissens des Menschen, Verringerung des Lernens aus Fehlern, Verringerung der Erwartung, daß Fehler auftauchen (und damit der Wachsamheit) und einer Komplizierung des Gesamtsystems.

# 3 Das Konzept von Fehlermanagement

Begrifflich ist Fehlermanagement von Fehlerbewältigung abzugrenzen. Während Fehlerbewältigung jede Art von Herangehensweise an einen Fehler beinhaltet - und damit einen deskriptiven Begriff darstellt, impliziert Fehlermanagement einen präskriptiven Sinn: Fehlermanagement bedeutet also sinnvolles Herangehen an einen Fehler mit den Zielen, Folgefehler zu vermeiden, die negativen Effekte der Fehler nicht aufkommen zu lassen und die Fehlerfolgen schnell zu beseitigen.

Man sollte zwischen einem Fehler und den negativen Fehlerfolgen differenzieren. Im Prinzip folgt nicht auf jeden Fehltritt ein Fall. Nicht jedem Fall folgt ein Bruch des Armes, usw. Fehlermanagement beinhaltet also die Vermeidung oder Reduktion der negativen Konsequenzen von Fehlern.

Was sind nun die negativen Konsequenzen? Im folgenden eine - sicherlich unvollständige - Liste:

- Ein Fehler wird nicht bemerkt; dadurch werden die negativen Konsequenzen akkumuliert (z.B. man merkt nicht, daß dies der falsche Programmieransatz war und verschwendet deshalb noch mehr Zeit für das Programmieren mit diesem Ansatz).
- Fehler führen zu negativen Emotionen; diese negativen Emotionen müssen bewältigt werden und benötigen dazu einen Teil der sowieso schon beschränkten Informationsverarbeitungskapazität.
- Fehler führen zu Zeitverlust; dies passiert z.B. dann, wenn ein Fehler wieder vollständig korrigiert werden muß.
- Fehler führen zu Verlust an anderen Ressourcen; wenn z.B. aufgrund einer irrtümlichen Reformatierung der Festplatte Programme verloren gehen, die dann neu erstanden werden müssen.
- Fehler führen zu weiteren Fehlern; Arbeitende müssen nach einem Fehler oftmals von der reinen Routine auf eine eher intellektuelle Problemlösung umschalten. Diese Umschaltprozesse sind immer schwierig, denn dadurch wird die Überwachung von Handlungen und das weitere Vorausplanen gestört; es kann deshalb zu weiteren Folgefehlern kommen. In unseren Untersuchungen sind uns immer wieder "Fehlerkaskaden" aufgefallen, bei denen ein Fehler zum nächsten führte - auch Fehler, die sonst normalerweise nicht passieren würden.

- Fehler führen zu nicht korrigierbaren Schäden. Besonders deutlich im Umweltbereich oder bei Unfällen mit Verletzungen oder Toten haben Fehler Konsequenzen, die nicht wieder gut zu machen sind.

Zur Vermeidung dieser negativen Konsequenzen kann Fehlermanagement durch Instrumente und Werkzeuge, durch organisationale Maßnahmen, sowie durch Training unterstützt werden (vgl. Irmer, Pfeffer & Frese, 1991). Wir konzentrieren uns hier auf die Unterstützung durch die Software. Computer sind eigentlich sehr gut geeignet, Fehlermanagement zu unterstützen, weil sie zwischen einer virtuellen und der wirklichen Welt unterscheiden. Beim Schreiben dieses Artikels erscheint ein Fehler zunächst nur auf dem Bildschirm und nicht gleich auf dem Papier. Welche Prozesse können nun dazu beitragen, Fehlermanagement zu ermöglichen?

# 4 Wie kann man Fehlermanagement ermöglichen?

Fehlermanagement kann unter drei Gesichtspunkten betrachtet werden: unter dem *Fehlerprozeß*, dem *Handlungsprozeß* und den *Regulationsebenen*. Im folgenden werden dazu einige Beispiele gegeben (vgl. ausführlich Zapf, Brodbeck, Prümper & Peters, 1991; Zapf, Frese, Irmer & Brodbeck, 1991).

## 4.1 Fehlerprozeß

Der Fehlerprozeß besteht aus den folgenden Bestandteilen (die sich z.T. überlappen und nicht immer geordnet auftreten; vgl. Zapf, Lang & Wittmann, 1991): *a) Fehlerentdeckung*, also das Wissen um die Tatsache, daß ein Fehler gemacht wurde; *b) Fehlererklärung*, also die Einordnung des Fehlers in einen größeren Erklärungszusammenhang; *c) Fehlerbehebung*, also die Handlungen der Beseitigung oder der Kompensation des Fehlers.

*Zu a) Fehlerentdeckung.* Ganz offensichtlich gehört zum Fehlermanagement zunächst die Entdeckung des Fehlers. Da mit dem zeitlichen Abstand zum usprünglichen Fehler die Tendenz sinkt, den Fehler zu erkennen, ist rechtzeitige Fehlerentdeckung besonders wichtig. Zwei Voraussetzungen helfen hier:

- Klares Feedback; ohne Feedback ist es nur sehr schwer, einen Fehler zu entdecken. Dabei ist nicht nur an Fehlermeldungen gedacht, sondern auch an allgemeines Feedback, das einen Vergleich des Systemzustands mit den aufgestellten Zielen ermöglicht.

- Transparentes Design eines Systems, d.h. der Benutzer kann sich ein gutes Abbild von den Intentionen und den Ausführungen des Designers machen und weiß deshalb genau, an welchem Punkt er sich befindet.

*Zu b) Fehlererklärung.* Nicht jeder Fehler verlangt eine Fehlererklärung. Es ist sogar so, daß gute Unterstützung von Fehlermanagement oft gerade die Möglichkeit eröffnet, auch ohne Fehlererklärung eine Fehlerkorrektur durchzuführen (z.B. die UNDO-Taste oder der Back-up File). Bei Routinekorrekturen gibt es auch keine abgrenzbare Fehlererklärung, da man sofort nach dem Auftreten eines Fehlers die Korrektur ausführt, z.B. man vertippt sich in der Textverarbeitung und benützt die "backspace"-Taste.

Fehlererklärung kann zum einen bedeuten, daß der Betroffene weiß, wie dieser Fehler zustandekam, z.B. es wurde die falsche Tastenkombination gedrückt. Ein zweiter Aspekt beinhaltet das "Warum", also das Einordnen des Fehlers in einen größeren Zusammenhang, z.B. der Benutzer entdeckt, daß an einem bestimmten Punkt die Schreibtischmetapher nicht durchgehalten wurde; der Handlungfehler weist dann auf die Grenzen dieser Metapher hin.

Für Fehlererklärung gelten dieselben Prinzipien wie für die Fehlerentdeckung: Ein transparentes System und gute Rückmeldungen erhöhen das Wissen um das System und um die Fehlermöglichkeiten. Zusätzlich gelten:

- Aufstellen von Hypothesen erleichtern; transparente Systeme erleichtern dies. Hypothesen können auch durch das System selbst vorgeschlagen werden, z.B. im Rahmen von Fehlermeldungen oder von (kontextspezifischen) Hilfen.
- Explorationsmöglichkeiten zur Verfügung stellen; oftmals kann man erst dann feststellen, was man falsch gemacht hat und warum, wenn man die fehlerhafte Handlung noch einmal wiederholen bzw. ähnliche Handlungen ausführen kann. Es ist also notwendig zu explorieren (vgl. Greif & Keller, 1990). Ein System erleichtert dies, wenn Bedingungen leicht rückgängig gemacht werden können. Es gibt z.B. bei manchen Programmen eine history-Funktion, die es dem Benutzer erlaubt, auf einen bestimmten Punkt zurückzugehen und von dort aus noch einmal neu zu starten.
- Fehlermeldungen und Hilfesysteme; sie können direkt erklären, wie der Fehler zustandekam und was man daraus lernen kann.

*Zu c) Fehlerbehebung.* Fehlerbehebung kann als Vorwärts- und Rückwärtskorrektur (Zapf, Lang & Wittmann, 1991) geschehen. Bei einer Vorwärtskorrektur wird der Fehler behoben, ohne schon ausgeführte Teilhandlungen noch einmal ausführen zu müssen; z.B. wenn die Suchfunktion den Benutzer fälschlicherweise an das Ende des Textes geführt hat und er dann einfach eine Suchfunktion nach oben einschalten kann. Bei einer Rückwärtskorrektur muß z.B. ein verlorengegangener Text noch einmal geschrieben werden.

Das Fehlermanagement wird unterstützt durch:

- Orientierung; d.h. durch das Wissen, wo man sich befindet und in welche Richtung eine Fehlerbehebung gehen kann. Die Orientierung wird durch das System dann erleichtert, wenn man leicht zu einem bestimmten Fixpunkt gelangt (z.B. aus jedem Submenü zum Hauptmenü gehen kann) und wenn die

Fehlermeldungen bereits mögliche Strategien der Fehlerbehebung vorschlagen (z.B. bei Rechtschreibeprogrammen wird eine mögliche richtige Alternative bereitgestellt).
- Direkte Korrektur von Teilhandlungen. Die UNDO-Funktion und das multiple Zurückgehen von Schritten im Sinne einer multiplen UNDO-Funktion wurden bereits angesprochen.
- Unterstützung der Vorwärtskorrektur. Systeme unterstützen dann vorwärtskorrigierende Strategien, wenn sie Verzweigungen anbieten, z.B. bedeutet eine starre, stark hierarchisch gegliederte Menüstruktur, daß der Benutzer von einem irrtümlich aufgerufenen Submenü nicht mehr in ein anderes gehen kann, sondern erst wieder zum Ausgangspunkt zurückkehren muß.
- Handlungen unterbrechen können; Fehlermanagement wird dann unterstützt, wenn man Handlungen unterbrechen kann (z.B. mit Hilfe von Windows andere Teilhandlungen ausführen, oder bei einem bereits laufendem Spellprogramm durch Unterbrechung eine falsche Satzstruktur verbessern).

## 4.2 Der Handlungsprozeß

Hier kann auf den Handlungsprozeß rekurriert werden, der auch unserer Taxonomie von Fehlern unterliegt (Frese & Zapf, 1991; Zapf, Brodbeck & Prümper, 1989): *Ziele, Informationsaufnahme und -integration, Pläne, Monitoring* und *Feedback.*

Im Bereich der *Zielentwicklung und -entscheidung* sind genaue Zielvorstellungen (einschließlich Unterzielbildung) und Überlegungen zu möglichen Zielkonflikten wesentlich für Fehlermanagement. Zum Beispiel erlaubt eine genaue Spezifikation der Unterziele eine bessere Einordnung von Rückmeldungen und ein schnelles und präzises Wissen darüber, ob man noch auf dem Weg zum Ziel ist. Da wir der Meinung sind (Frese & Peters, 1988), daß Zielbildungen im Computer nicht abgebildet werden können, kann in diesem Bereich das System nur wenig unterstützend wirken.

Im Bereich der Informationsaufnahme und -integration läßt sich Fehlermanagement durch gute und konsistente Metaphern und Analogien unterstützen. Wenn ein System eine Analogie genau durchhält, ist es leichter, ein gutes mentales Abbild zu entwickeln, das wiederum dabei hilft, Abweichungen vom Ziel zu entdecken, zu erklären und Fehler zu korrigieren.

Darüber hinaus soll das System Informationen sowohl in komprimierter als auch in aufgelöster Form darbieten. Komprimierte Information (möglichst mit Leitinformation der wichtigsten Variablen) ist notwendig, um einen schnellen Überblick zu erlangen und die begrenzte Informationsverarbeitungskapazität des Menschen nicht zu überfordern. Andererseits werden für die Fehlererklärung und die Korrektur oft auch detaillierte Informationen benötigt.

Ein System unterstützt die *Planentwicklung und -entscheidung* durch hohe Transparenz und Konsistenz. Transparenz bedeutet die Durchschaubarkeit der

Systemlogik, und Konsistenz beinhaltet das Prinzip der geringsten Verwunderung über Systemreaktionen. Fehlende Transparenz und Konsistenz bringen geordnete Pläne durcheinander und erschweren Umplanungen. Wesentlich für das Fehlermanagement ist die Unterstützung von Neuplanungen. Hier helfen alternative Pläne weiter, die oft zunächst erst einmal ausprobiert werden müssen. Die Software kann dies z.B. durch eine "Spielwiese-Funktion" unterstützen, unter der man dann gefahrlos eine bestimmte Strategie und deren Resultate ausprobieren kann.

Beim *Monitoring* sind Gedächtnisprozesse wesentlich. Gedächtnisprobleme können durch Hilfefunktionen, Menülisten, durch Dateimanager und durch Suchfunktionen reduziert werden. Letzteres ermöglicht es dem Benutzer z.B. nicht nur nach einem Wort in einem Text, nach einem File auf einer Festplatte, sondern auch nach inhaltlichen Konzepten über verschiedene Files hinweg zu suchen - etwa wenn man den Dateinamen eines Briefes vergessen hat und nun z.B. durch Angabe von Datum, Adresse, usw. den Brief finden kann. Windows sind für Monitoring besonders wichtig, denn oft führen Unterbrechungen dazu, Merk-, Vergessensfehler und Unterlassensfehler zu provozieren. Mit der Windowtechnik kann man den Ausgangszustand zum Zeitpunkt der Unterbrechung konstant präsent halten. Wenn eine Sachbearbeiterin z.B. während ihrer Tätigkeit angerufen wird und eine Adressensänderung vornehmen muß, macht sie das durch Aufrufen eines zweiten Windows und weiß dann nach dieser Tätigkeit wieder, welche Arbeiten sie auf dem ersten Window unterbrochen hat. Wenn sie nun die Stelle, an der sie sich vorher befand, vergessen hat, zeigt die Cursor-Stellung, wo die Arbeit unterbrochen wurde.

Die Funktion von Feedback für das Fehlermanagement wurde bereits schon besprochen und wird deshalb hier nicht wiederholt.

## 4.3 Regulationsebenen

Fehlermanagement läßt sich auch unter dem Gesichtspunkt der Regulationsebenen diskutieren. Auch die Regulationsebenen liegen unserer Taxonomie zugrunde (Frese & Zapf, 1991; Zapf et al. 1989). Man kann zwischen den bewußten oberen Regulationsebenen und den routinemäßigen unteren Regulationsebenen unterscheiden. Ein noch unbekanntes Problem wird auf den oberen Regulationsebenen gesteuert. Hingegen regulieren erfahrene Autofahrer die meisten Handlungen, z.B. das Schalten und Steuern auf den unteren Ebenen. Wann immer wir etwas auf den unteren Regulationsebenen steuern, haben wir den Kopf für anderes frei. Den höheren Regulationsebenen sind prinzipiell in ihrer Kapazität Grenzen gesetzt - das Arbeitsgedächtnis kann nur eine beschränkte Anzahl von Informationen gleichzeitig bewältigen.

Auf den oberen Regulationsebenen sind Fehler oft schwer zu entdecken und zu erklären, weil die Vergleiche zwischen Ziel und augenblicklicher Lage aufgrund der komplexen Situation schwieriger sind. Hier unterstützen solche Systemberei-

che das Fehlermanagement, die den Vergleich des augenblicklichen Zustands mit dem Ziel erleichtern.

Fehlerbehebung ist auf den oberen Regulationsebenen ebenfalls oft erschwert, weil hier sowieso schon komplexe Prozesse reguliert werden müssen und jedes weitere Problem, z.B. ein Fehler, die Grenzen beschränkter Kapazität sprengt. Aus diesem Grund erleichtern unspezifische Methoden das Fehlermanagement - hier sei wieder auf die Vorzüge einer UNDO-Taste verwiesen. Denn unspezifische Methoden können routinemäßig eingesetzt werden, sie werden also von unteren Ebenen reguliert.

Bei Wissensfehlern müssen Nachprüf- und Nachschlagemöglichkeiten zur Verfügung stehen, die dem Benutzer eine schnelle Fehlerbehebung ermöglichen. Oftmals sind jedoch die Manuale der Systembeschreibung alles andere als günstig für das Fehlermanagement (vgl. Peters & Bichler, 1989).

Andererseits gibt es auch auf den unteren Regulationsebenen Erschwernisse. Diese beziehen sich v.a. auf die Fehlerentdeckung, weniger auf die Fehlererklärung und -behebung. Auf den unteren Regulationsebenen wird Feedback weniger stark beachtet, da die Handlungen oftmals so schnell ausgeführt werden, daß Rückmeldungen gar nicht mehr verarbeitet werden können. Bei potentiell sehr negativen Konsequenzen sollte deshalb die Handlungsrückmeldung besonders deutlich und "aufdringlich" sein. Beispielsweise sollte vor einer möglichen Reformatierung einer Festplatte, die zum Verlust sämtlicher Daten führt, mit extra großen und evt. deutlich blinkenden Zeichen gewarnt werden.

Da Rückmeldungen "verspätet" aufgegriffen werden, sind hier oft multiple UNDO-Möglichkeiten besonders brauchbar, da das Zurücknehmen lediglich eines Schrittes oft nicht genügt, weil bereits mehrere Schritte aufgrund der verzögert wahrgenommenen Rückmeldung falsch gemacht wurden.

# 5 Abschluß

Für das Konzept von Fehlermanagement scheinen uns viele gute empirische und theoretische Gründe zu sprechen. Es wurde natürlich auch schon von anderen Forschern in der einen oder anderen Weise angesprochen. Wir meinen aber, daß es sich lohnt, das Konzept systematisch zu explorieren und in der Software-Entwicklung einzusetzen.

Darüber hinaus meinen wir, daß es eine große Palette von Anwendungen dieses Konzeptes gibt, die es noch zu erforschen gilt, z.B. im Bereich der Software-Entwicklung (in der die Frage, wie man mit Spezifikations- und Programmfehlern umgeht, besonders gravierend ist), für die Unternehmensführung (wie kann man erreichen, daß Managementfehler nicht in Katastrophen einmünden?), in der Unfallforschung, usw.

# 6 Literatur


Brodbeck, F. C., Zapf, D., Prümper, J., & Frese, M. (1990). Error handling in office work with computers: A field study. München: Manuskript, zur Publikation eingereicht.

DIN 66 234. Teil 8. Normenausschuß Informationsverarbeitungssysteme (NI) (1988). Bildschirmarbeitsplätze. Grundsätze der Dialoggestaltung. Berlin: Deutsches Institut für Normung e. V.; Beuth-Verlag.

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., & Peters, H. (1988). Zur Fehlerbehandlung in der Software-Ergonomie: Theoretische und praktische Überlegungen. Zeitschrift für Arbeitswissenschaft, 42, 9-18.

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

Frese, M. & Zapf, D. (Hrsg.) (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.

Irmer, C., Pfeffer, S., & Frese, M. (1991). Praktische Konsequenzen für das Training. In M. Frese & D. Zapf (Hrsg.), Fehler und Schwierigkeiten bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich. Bern: Huber.

Perrow, C. (1988). Normale Katastrophen. Die unvermeidbaren Risiken der Großtechnik. Frankfurt: Campus.

Peters, H., & Bichler, S. (1989). Benutzerfehler und Nutzungsprobleme bei der Arbeit mit Software: Welchen Beitrag leisten die Handbücher? In S. Maaß & H. Oberquelle (Hrsg.), Software Ergonomie '89 (S.233-243). Stuttgart: Teubner.

Prümper, J., Zapf, D., Brodbeck, F. C., & Frese, M. (1990). Errors of novices and experts: Some surprising differences in computerized office work. München: Manuskript, zur Publikation eingereicht.

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 (Hrsg.), 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 (Hrsg.), 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 (Hrsg.), Fehler und Schwierigkeiten bei der Arbeit mit dem Computer. Ergebnisse von Beobachtungen und Befragungen im Bürobereich. Bern: Huber.




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