Hoc, P.C. Cacciabue, E. Hollnagel (Eds.) Expertise and Technology:
Cognition & Human-Computer Cooperation, pp. 195-212, 1994

Handling Human Error

Antonio RIZZO, Donatella FERRANTE, Sebastiano BAGNARA

Introduction

- ✓ 現在の課題
  - ➢ HCI などユーザを中心とした man-machine システムのデザインアプローチ
    - ✧ エラー: 人と人工物とのコミュニケーションの断絶
      (e.g. Norman & Diaper, 1986; Rasmussen, 1986; Sheiderman, 1992)
  - ➢ エラー回復のためのガイドライン
    - ✧ ほとんどがユーザのパフォーマンスレベルで生じるエラーに関するもの
      - ● 人と人工物のインタラクションにおける高次なレベルには関連が無い
    - ✧ エラー処理の認知的プロセスについての知識が十分に確立されていない

- ✓ 2 つの理由
  - ➢ 認知心理学において，知覚と行動が独立した領域だと考えられ，とりわけ知覚は特別扱いされてきた
    - ✧ 知覚と行動の区別は当てにならない (e.g. Neisser, 1976; Arbib, 1989)
    - ✧ 知覚は行動の選択，実行およびその結果の評価に含まれる (Norman & Shallice, 1986)
      - ● 現在ではヒューマンエラーは action-perception loop の断絶だと考えられる
  - ➢ 知覚の研究者と行動の研究者の間に緊密なコミュニケーションが無かった
    - ✧ 実験的な認知心理学と認知人間工学のつながりも非常に弱かった

- ✓ 本稿の目的
  - ➢ エラー処理プロセスの認知的な分析を提供
    - ✧ 扱う問題は以下のエラーに関する研究に基づく
      - ● 実験室 (Rizzo, Bagnara, & Visciola, 1987; Rizzo, Ferrante, Stablum, & Bagnara, 1988)
      - ● 工業環境 (Gagnara, Stablum, Rizzo, Fontana, & Ruo, 1987)
      - ● 日常生活 (Rizzo, Ferrante, & Bagnara, 1992)

Human Error

- ✓ 定義 (Reason, 1990; page 9)
  - ➢ 「計画された知的または物理的な活動過程で，意図した結果が得られなかったときで，これらの失敗がほかの出来事によるものでないときの，すべての場合を包含する本質的な項目」

- ✓ 分類
  - ➢ 知識活性化のプロセスや認知制御レベルに基づいてカテゴライズ (Reason, 1990)
    - ✧ Slip: 意図と行動のミスマッチ
      - ● 意図は要件を満たすが行動が計画通り実行されない (Norman, 1981)
      - ● 行動を制御する低いレベルでの注意の失敗によって生じる
        - ➢ 自動化，過剰に訓練された振る舞い → 決まりきった状況

- ✧ Lapse: 記憶の失敗を含み，必ずしも明確な行動として現れない
  - 3 つのサブクラス
    - ➢ 意図の lapse: 実行中に意図を失念
    - ➢ 行動の lapse: 意図した行動のトリガーに失敗
    - ➢ 記憶の lapse: 意図は特定されるが，行動遂行のために必要なより詳しい情報を失念
- ✧ Rule-based mistake: 良く知ったルールまたは手続きの誤った活性化
  - よく使われるルールが適切なルールにオーバーライド
  - 状況の同定と行動の計画をマッチングするプロセスで現れる
- ✧ Knowledge-based mistake: 選択された計画または意図された目標が問題に対して不適合
  - 使用されたメンタルモデルの不完全性，原因同定の失敗

Error Handling Processes

✓ 先行研究におけるエラー処理プロセス

➢ Bagnara et al. (1987)

a) Mismatch emergence: フィードバックとアクティブな知識の枠組みとの相違の出現
- アクティブな知識の枠組み
  - ➢ アクティブな予想 (意図の充足状況)
  - ➢ 暗黙的な仮定 (e.g. 暗黙知)

b) Detection: 何が間違っているか (エラー) に気づくこと

c) Recovery: ミスマッチを減少し，または取り除くこと

➢ Sellen (1990)

a) Detection: 意識的または無意識的にエラーが起きたことを知ること

b) Identification: 何が誤ってなされたのか，本来何が行われるべきだったかを知ること

c) Correction: エラーの効果をどのようにして元に戻し，望ましい状態を達成するかを知ること

➢ Bagnara et al. (1987) と Sellen (1990) の違い → ミスマッチの役割

✧ ミスマッチの出現とエラー検出 (detection) の過程は必ずしも重ならない (e.g. Rizzo et al. 1992)

E1) コピー機がコピーとほとんど白い用紙を吐き出した．当初，上手く動作しない原因は給紙の問題であると考えたが，同じ現象が続くのでコピー機を調べたところ，以前自分が行った設定を初期化するのを忘れていたことに気づいた．

- 予期しない結果の観察によって不整合が生じる
- この結果が自分自身の誤った行動によって生じたことを考えなかった

✓ 新しい提案

➢ 先行研究を考慮して 4 つのプロセスを提案

a) Mismatch emergence: Perception-action loop の断絶

b) Detection: エラーを生じたことに気づくこと

c) Identification: 断絶の原因を知ること

d) Overcoming of the mismatch: 不具合の現象，除去，または原因を元に戻すための方略

- 2 通りのエラー検出プロセス (Norman, 1984)
  - “Self monitoring”
    - 行動が監視され予測と比較される場合，行動が計画通りでなければその時点で検出
      - 予測と実際の行動を比較するモニタリング機能を持ったフィードバック機構
  - “Deleterious outcome”
    - エラーが有害な結末を引き起こした場合，その時点でエラー検出
      - 予測と出来事の差異を検出する認知システム

Mismatch

✓ 検出されたエラーの事例紹介により以下のことを示す
- 情報のタイプ
- 情報を評価するさまざまな照合システム

Mismatch and incoming information

✓ 知識の枠組みによって照合される情報に基づいて分類

a) Inner Feedback: 情報が作動記憶上で利用可能なものであり，その情報が行動自体や行動の即時的なフィードバックに依存しない

E2) 出先でなんとなく荷物が軽いと感じて，何をもって出かけたかをチェックしたところ，30 分ほど前に立ち寄った店に大きな本を途中で置いてきてしまったことに気づいた

E3) LISP のプログラミングで “REST を 2 回行って FIRST を 1 回行う” という計画に基づいて “(RES” とコマンドを打ち込み始めた段階で，そのコマンドでは課題を解決できないことに気づいた.

b) Action Feedback: 情報が行動そのものの要素から得られる
- 行動の結果を知る以前にも矛盾を経験することはある

E4) スイッチを操作しようと手を伸ばした. 触れたスイッチが一つ隣のスイッチであることに気づいたものの，そのスイッチを操作することを避けられなかった.

c) External Outcome Feedback: 情報が実行された行動そのものではなく，環境の結末から得られる
- 外的な世界からのフィードバック
  - 実行された行動によって生成された世界の状態 (action 後すぐに検出される)
  - 実行された行動の遅延した効果
  - 実行された行動の遅延した検出

E5) ずっと足が痛いと思っていた. 暫く後，ふと自分の足が視界に入ったとき，左右で違う靴を履いていることに気づいた.
- 特殊な例としては，コミュニケーションによってもたらされるものがある

E6) 特性の異なる A と B のプラントに資材が必要であることを，オペレータが管理者に伝えようとした.オペレータは A に資材が必要であることを B に関連した文脈で説明し，B に資材が必要であることを A に関連した文脈で説明した. 資材が必要な理由を管理者が理解できなかったため，オペレータは再度説明を行った. ただし，2 度目の説明では，それぞれを正しい文脈で説明した. この時点で，管理者はオペレータに最初の説明における “スプーナリズム” を伝えた.

d) Forcing Function: 情報が環境から得られ，意図した行動の実行または望ましい状態の達成を妨げる制約にその情報が関連する

E7) HyperCard のある領域の “Visible” の属性を “False” にしたかった．そこでメッセージウィンドウでコマンドを編集し，最後の部分を “to 21, 35, 63, 85” から “to false” に変更した．すると HyperCard が “Invalid augment for command set” というメッセージを返した．メッセージボックスのコマンドを見直すと，“set rect of field X to false” と書いてあることに気づいた．“rect” を読んだのと同時に何が間違っていたかを理解した．

e) Intention Uncertainty: 次に何をするのかという情報がはっきりしない

- ✧ 実行中の意図の喪失

E8) オフィスへ書類を取りに行く途中友人と出会い，話しこんでいるうちに何をするつもりだったか失念した．

- ✧ 意図の選択時における適切な選択肢の欠如

E9) プラントの生産計画課題において，オペレータは全体の 25 % を計画したところで次にするべき事に関して不安であると感じ，手を止めた．現在のプランが間違っていることを疑い，スケジュールを再分解して新しい順番でスラブを再配列した．後日生産は適切に実行された．古いスケジュールを調べてみたところ，2 つのスラブが処理されなかったであろう事が判った．

f) Standard Check Behavior: 知識の枠組みに関する状態を制御するために，情報が積極的に探索される

- ✧ チェックの回数はさまざまな要因に依存する
  - 個人の性格，タスクに関連するリスク，環境の複雑さ，要求される精度などなど

E10) 授業中にとったノートのデータを，後日自宅で勉強しているときに教科書に載っているデータと比較した．その結果，数字をミスタイプしていたことに気づいた．

Mismatch and frame of reference

- ✓ ミスマッチが生じる原因
  - ➢ 外界から得られる情報が予測と異なる
  - ➢ 一連の行動の間に予測が変化する → 知識のアップデート

- ✓ 作動記憶上の (アクティブな) 知識のアップデート
  - ➢ 適切な知識を活性化するために，やや不完全な知識を採用することが必要な状況
    - ✧ 新しい知識は新しい枠組みを定義する
  - ➢ アップデートのソース
    - ✧ 内的なソース
      - より強い知識が実行されて上手く行かない場合に，新しい知識がアクティブになる (e.g. Norman and Shallice, 1986, Umilta, Nicoletti, Simion, Tagliabue, & Bagnara, 1992)
    - ✧ 外的なソース
      - インタラクション後に利用可能な世界の状態に関する新しい情報から生じる
        - ➢ それまでに考えられていなかった知識の活性化
        - ➢ 既にアクティブな知識の質をより高める
      - 情報は必ずしも “予想に反するもの” ではない
    - ✧ 新しい枠組みの構築において，両者は排他的ではない
- ✓ 日常生活におけるエラーの研究から知識の枠組みは 4 つに分類できる

a) Stable knowledge frame of reference: 行動実行前に活性化した枠組みが行動の実行中ずっと維持される
   - ✧ 行動と結果のフィードバックをマッチングする通常のシステム

E11) コンピュータに数式を打ち込んで計算したところ自分の予想と異なる出力が得られた. 再度数式を打ち込みなおして計算したところ, 今度は予想と同じ結果が出力された. さらにもう 1 度繰り返しても, やはり予想通りの結果が得られた. 結果的に何らかのエラーを犯したことは判ったが, それがどんなものかは判らなかった.

b) Frame of reference active after the adoption or execution of the intended actions: 行動の採用または実行後に枠組みが活性化
   - ✧ ミスマッチの出現 → 作動記憶で知識が絶え間なく更新される過程
   - ✧ 事前に確立した知識が減少すると, 動作が開始された後でも新しい知識が活性化する
     - Kahneman and Miller (1986) の立場に近い

E12) 短時間で大量のコピーを作らなければならない状況で, コピーを始めてしまってから最後のページから順にフィーディングしていけば, ソートに要する時間と仕事を節約できたことに気づく.

c) Distant frame of reference: 誤った意図や行為から時間的, 概念的に離れたところに焦点を当てている
   - ✧ アクティブな知識は外界の状態に関連している
     - 大抵は forcing function か external feedback によって提供される
     - 生成されたエラーとの明確な関連は必ずしもあるわけではない

E13) 車のトランクを掃除するためにカーステレオのケーブルを外した. 後日左のスピーカが機能せず, システムの欠陥だと考えた. 1 週間後友人と原因について話していて, 数日前に車を修理に出したときに作業者がケーブルを接続し忘れていたことを思い出した. そしてそのとき, 自分も同じようにケーブルを接続し忘れたことを思い出した.

d) Lack of meaning: 進行中の活動を支配する明確な目標がないということへの気づき

E14) 入口の箱においてある鍵を取るために, 間違って書斎 (直前に別の人との話題に上った書類がある) に行ってしまった. そこで, 何のためにそこにいて, 何を探しているのかがわからなくなった.

✓ 一般には多数の異なるタイプのミスマッチが 1 つのイベントで生じ得る

E15) 現金を引き出そうと ATM にカードを入れ暗証番号を入力した. 画面上のメッセージが “操作ボタンを押してください” と変化したのを見落として待ち続けたが, カード番号をミスタイプしたのかと思い再度入力した. それでも上手く行かないので違うカードの番号を使ったのかと思い, 新しい番号を入力しようと画面を確認したところでメッセージが変わっていることに気づいた.

After the Mismatch

✓ ミスマッチの出現 ≠ エラーの検出
   - ➢ 複雑なエラーでは mismatch と detection だけでなく, detection, identification, recovery の間でも区別することが必要
     - ✧ ここでは, 異なるステージの共起が生じる状況と, 各ステージが分離されている状況を区別する

Detection

- ✓ エラー検出の定義
  - ➢ 共通の見解は得られていない
    - ✧ 文献によっては mismatch や identification を含む
      - ● エラー処理プロセスにおいて上記がしばしばオーバーラップするため
  - ➢ 本稿での detection の定義
    - ✧ 人々が自分で生成したエラーに気づくようになること

- ✓ Mismatch とオーバーラップする場合
  - ➢ ミスマッチがエラーの結果であると自動的に考えられる
    - ✧ プログラミングや数学の問題など良定義領域における典型

- ✓ Mismatch と detection が時間的に離れている場合
  - ➢ ミスマッチの原因はシステム自体の欠陥か完全に独立した外的な要因の影響かもしれない
    - ✧ 日常生活や工業環境など複雑な状況に見られる
  - ➢ 自分が生成したエラーを仮定する前に，他の説明を排除することが必要 (Norman, 1984, 1989)
    - ✧ エラーを言いくるめる → 主要な事故を促進する重大な要因
      - ● ミスマッチが現れたときにアクティブな枠組みと，エラーを犯したときにアクティブだった枠組みの差異が大きいほどこの傾向が強い

- ✓ Mismatch と detection の同時性に関連する要因
  - ➢ 取り扱う範囲についての仮定 (開いた系 / 閉じた系)
  - ➢ アクティブな枠踏みとエラーを犯したときの枠組みとの距離

Identification

- ✓ Mismatch と detection, identification がオーバーラップする状況
  - ➢ エラーが slip であるとき
  - ➢ 行動の結果のフィードバックが直ちに利用可能なものであるとき
  - ➢ 行動の実行中，アクティブな枠組みが変化しないとき

- ✓ Identification が複雑になる状況
  - ➢ 意図の誤った活性化 → Rule-based mistake
    - ✧ 逆方向連鎖
      - ● 結果の注意深い評価に基づいて
      - ● 前に実行された行動からエラーを辿る
  - ➢ Identification に必要な知識の枠組みとミスマッチによってトリガーされた枠組みとの距離が大
    → Knowledge-based mistake
    - ✧ 順方向/逆方向解析の間で混乱
      - ● もしエラーがあるとしたらいつ，どんなエラーを生じたのか
      - ● どのようにしたら望ましくない現状を回復できるのか

- ✓ Identification が複雑になる要因
  - ➢ 推論のバイアス
    - ✧ 人間の推論はシステマチックな方法に偏っている (e.g., Kahneman, Slovic, & Tversky, 1982)
    - ✧ ある種の情報の無視と，極狭い領域への注目 (Girotto, Legrenzi, Johnson-Laird; 1993)
    - ✧ 確証バイアス (Wason, 1960)

Recovery

- ✓ Identification と recovery がオーバーラップする状況
  - ➢ “可逆” なシステム
    - ✧ 多くの場合, identification によって recovery の道筋が明らかになる

- ✓ Recovery が単独で現れる状況
  - ➢ 不可逆なシステム，または原因の同定ができないとき
    - ✧ 原因の同定とは独立に，ミスマッチを取り除く，あるいは減少しようとする
      - ● 理想的には，ネガティブな結末を取り除くことを試みる
    - ✧ 馴染みのある修復活動を通じた順方向の解析
      - ● 多くの場合，気づいた症状に基づいて既に持っている行動のセットから行動を選択する
        - ➢ 症状に基づく行動は症状自体の推論に由来する (Brehmer, 1987)
          - ✧ 利用しやすく，馴染みのある認知的ルーチンワーク

- ✓ ユーザは自分の犯したエラーを理解しようと試みる
  - ➢ そうでなければ，現状を無視して，目標を達成するための別の方法を探すだろう
    - ✧ それもできなければ，目標のほうを現状に合わせようとするだろう

Guidelines for Handling Error

- ✓ エラー処理をサポートするには
  - ➢ 知識の活性化 (Woods, 1991, pp. 253)
    - ✧ ある知識が必要になったとき，その知識が活性化されることは必ずしも保証されない
      - ● タスクが行われる状況において，関連する知識がアクセス可能かどうかという点が重要
  - ➢ 知識の活性化はヒューマンエラーを処理するプロセスをサポートする上で重要な点
    - ✧ エラー処理の問題

      → タスクが実行される条件を調整することで，関連する知識の活性化をサポートする問題

1) 行動をより認知可能なものにする (E3, 4, 6
   - ➢ 異なる目標には異なる行動を割り当てる
     - ✧ 他の行動の活性化を抑制
     - ✧ 行動自体の要素に，現在行っている活動についての明白な情報を提供させる

2) Multi-sensory feedback の利用 (E5
   - ➢ エラーを生じたとき，早期のミスマッチの生成を促進する
   - ➢ E5 は触知性の固有感覚情報がペアの靴を履くことを促進する上で効率的かを示している
     - ✧ しかし，たとえ情報が利用可能であったとしても，その人がミスマッチを経験するとは限らない

3) 高次元で具体的なメッセージ (E7
   - forcing function のサポートを試みるとき，以下の情報を提供するべき
     - どのオブジェクトが含まれるのか
     - 個々のオブジェクトに対してどの値が不正なのか
   - E7 の場合
     - デフォルトのエラーメッセージ
       - “Invalid argument for command set” ← “rect” が含まれない
     - ガイドラインを適用した例
       - “rect do not have false or true values”
       - “false is a value for x and y but not for rect”

4) 活動ログの提供 (E8, 9, 14
   - 人間は外部の記憶補助に著しく依存している
     - E8, 9, 14 からも明らか
   - ログの提供方法

     i) 時刻，行動，システムの変更点をラベリング

     ii) 出来事についてのインタラクションの形態を抽出
       - 変更されたオブジェクト単体だけでなく、変化が生じている間に潜在的に利用可能なすべての情報を記録
         - プロセス制御のような長期間に渡る複雑なセッションにおいて役に立つかもしれない

5) 比較を許容する (E10
   - 互いに関連する結果は容易に比較されるべきである
     - 重大な違いや原因を選び出すことはできないかもしれないが，複数の状態の違いを見つけることはできる
   - 一つのオブジェクトに対して複数のツールを用いて行われる活動に有効

6) フィードバックの早期評価を可能にする (E12, 13
   - フィードバックの効果
     - 所望の行動を適切に遂行することが可能
     - どんな行動をとるべきかについての考えを変えることにも重要な要因
   - フィードバックのプレゼンテーションを良くする方法
     - Exploition of layout (see Norman, 1993)
     - Exaggeration of the difference
       - 知覚的システム → オプティカルフローの変化
       - 概念的プロセス → 対象間の比率や関連性
     - Stress on the aspects relevant to the just performed task
       - 情報の一部のみが行動の制御と結果の評価に使われる
         - タスクに強く関係する情報は結果のプレゼンテーションにおいて特に強調されるべき

7) Why the result？ Why the beep？ (E11, 15
   - ミスマッチ後にエラーの同定をサポートするには，明確な回答をユーザに与えることが最良
   - 実例
     - Mathematica などでは，ユーザから意味を成さない要求があったときビープが鳴る
       - このとき，ユーザは “Why the beep” コマンドによって，限定された対話セッションを開くことができる
   - 同じようなストラテジーの採用
     - 出力がユーザの予想と違う場合にシステムに対して疑問を提示する (Why the reslut)

Conclusion

✓ ここ数年の HCI デザインの領域における成果
   - 関連した知識へのアクセスの容易化
     - GUI
     - アフォーダンスの利用
       - インターフェイスデザインにおけるメタファの利用は知識活性化問題の成功例

✓ 一方で，上記の成果は故障に対処することにおいては決定的ではない
   - メタファの副作用
     - 知識へのアクセスを容易にする側面
     - 目標に到達できる方法を制限する側面
       - この制限はデザイナーによって作られたもの
       - 同様にフィードバックやエラーメッセージ，ヘルプもデザイナーの決定に応じて提供される

✓ コンピュータベースシステムのデザインにおける Winogard and Flores の原則
   - “コンピュータツールはアプリケーション領域の故障の予測と修正を支援することができる” (1986; page.166)
     - この原則はインタラクションの意味論と語用論に関連がある
   - 本書の知見から解釈すると…
     - さまざまなミスマッチに対応するために関連する知識の活性化を容易にする

✓ ヒューマンエラー処理をサポートするためにデザインされたツールは……
   - その活動に含まれる認知的プロセスを支えるものであるべき