Tratamento de erros

A gênese do erro (Lewis and Norman 1986)

As situações de erro normalmente ocorrem quando o usuário faz alguma coisa à qual o sistema é incapaz de responder.

Por que estas situações são consideradas erros? De fato, o que realmente ocorre é que o sistema não consegue interpretar a informação que lhe foi enviada. E, então, isto é chamado erro ou falha cometida pelo usuário.

Esta é uma postura arrogante dos projetistas de sistemas. Uma resposta mais apropriada seria:

> "*Sinto muito, mas estou confuso. Você poderia me ajudar a sair deste impasse?*"

Considere-se o erro específico e algumas alternativas de tratamento e prevenção a seguir.

Suponha que um usuário digitou a seguinte especificação de arquivo:

> **/csl/norman/HCIbook/Lewis/Norman/designingforError**

e que o erro tenha sido o "d" invés do "D" na última expressão.

Os sistemas respondem freqüentemente de alguma das formas:

> "*syntax error near line1*"
> "*no such file or directory*"

Estas opções faria com que o usuário precisasse rescrever a expressão na íntegra, e sem qualquer auxílio no sentido de identificar e corrigir o erro feito na primeira tentativa.

Examine agora esta alternativa de resposta:

> *Não consigo achar o nome de arquivo que você especificou:*
> */csl/norman/HCIbook/Lewis/Norman/designingforError*
> *Você quer dizer:*
> 1. /csl/norman/HCIbook/Lewis/Norman/DesigningforError
> 2. design-n*otes*
>
> *Por favor, digite o número correspondente à opção desejada.*

No exemplo de especificação de arquivos, podem ter ocorrido os seguintes erros:

- o arquivo efetivamente não existir;
- ele existir mas o nome ter sido digitado errado;
- o arquivo existir mas o usuário estar trabalhando num diretório errado.

Nestas situações, pode-se pensar em:

- verificadores ortográficos;
- procedimentos inteligentes de verificação;
- procedimentos de busca em estruturas que identifiquem erros de localização.

Mas uma maneira alternativa de evitar erros está baseada no uso de outra formas de representação dos objetos.

Diretórios representados por estruturas visuais e arquivos representados por ícones na tela a e cuja especificação é exibida ao se apontar para o ícone em questão evitariam problemas de digitação de caracteres e, adicionalmente, tornariam impossível a confusão em relação à localização do arquivo na árvore.

No entanto, esta representação teria associada a possibilidade de um "*click*" no lugar errado, além de algumas limitações, como, por exemplo, a incapacidade de se especificar uma família de arquivos por meio do uso de máscaras.

Como foi dito várias vezes, o projeto de interfaces envolve um contínuo compromisso. O importante é ter em mente todos os aspectos associados a um bom *design*, de forma a tentar fazer opções sensatas.

Classificação de erros

Existem diferentes tipos de erros. Um dos tipos mais freqüentes decorre da generalização. Uma taxonomia abrangente é apresentada por Norman.

Erros decorrentes da generalização

Muitas pressuposições erradas dos usuários partem da generalização dos conceitos aprendidos no uso de outros sistemas. Isto coloca a necessidade de as interfaces revelarem tanto o seu potencial e limitações quanto a forma as diferentes tarefas devem ser realizadas.

Exemplos de generalizações são aquelas realizadas por um usuário de PC ao passar para o ambiente UNIX. Observando outro usuário eliminar um arquivo com o comando **rm**, induziu que era a abreviação de **remove**, e utilizou para apagar diretórios. A mesma pessoa observou um usuário sair do programa de **mail** com **exit**, e induziu que era o comando apropriado para sair do sistema. No entanto, ao entrar novamente verificou que as operações que ele tinha realizado não tinham sido efetivadas. De fato, a existência de várias formas de sair do programa não tinha passado pela sua cabeça.

Norman (1981) classifica os erros em dois tipos básicos: *mistakes* (erros propriamente ditos) e *slips* (distrações).

> ***Mistakes*** ocorrem por deliberação consciente. Uma ação incorreta é baseada numa decisão incorreta.
>
> Exemplo: tratar de jogar o ícone do *hard disk* no cesto de lixo no Mac. Uma seleção no menu para eliminar todos os arquivos seria a ação correta.
>
> ***Slips*** são não-intencionais. Eles ocorrem por acidente.
>
> Exemplos: Erros de digitação, de seleção errada em menu por uma ação precipitada, entre outros.

Slips podem ser subdivididos em 6 tipos: erros de captura, erros de descrição, erros orientados a entrada, erros de associação e ativação, erros de perda de ativação, e erros de modo.

erros de captura: Quando se executa um comando que compreende (captura) a função desejada mas faz algo além. Exemplo: salvar e sair, ao invés de só salvar.

erros de descrição: Quando a ação correta é executada sobre o objeto errado. Este tipo de erros são comuns por distração. Exemplo: “Clickar” no quadradinho de *zoom* ao invés de no de fechar a janela.

erros orientados a entrada: Ocorrem quando informação externa (que pode estar no meio real ou na tela) acaba interferindo na ação sendo executada. Exemplo: nomear um arquivo pelo nome que está aparecendo na barra superior da janela.

erros de associação e ativação: Ocorrem quando associações com pensamentos internos modificam a ação em execução. Exemplo: Nomear um arquivo com o nome de uma pessoa na qual se está pensando.

erros de perda de ativação: Ocorrem quando há perda de consciência do usuário em relação à tarefa que ele estava querendo que o sistema executasse.

erros de modo: Se referem ao erro de estado do sistema. Exemplo é a tentativa de executar um comando quando o sistema está em modo de entrada de dados.

Prevenção de erros

O tratamento de erros consiste de duas partes igualmente importantes: a prevenção e o tratamento ou correção posterior à ocorrência. Esta seção trata da primeira delas.

Abordando o tema de maneira geral, pode-se dizer que há várias formas de minimizar a ocorrência geral de erros, dentre as quais:

- Um bom projeto do sistema, incluindo o *layout* da interface e a seleção adequada de rótulos, fácil interpretação do status do sistema e bom *feedback*;
- Um bom modelo conceitual e uma imagem do sistema revelada e consistente com esse modelo;
- Uma interface que suavize os golfos de execução e avaliação;
- O uso de representações e linguagens naturais para o usuário.

Entre as formas específicas de evitar a ocorrência de erros podem ser citadas:

- Representação visual dos objetos (é impossível se referir a um objeto que não está representado);
- Oferecimento de listas de objetos disponíveis para evitar digitação;
- Oferta de menus para evitar erros de sintaxe.

Entre as formas possíveis de reduzir a ocorrência de erros Shneiderman (1998) cita as seguintes:

> **Eliminação, sempre que possível, dos modos de execução** (Exemplo: dados ou comando)**;**
>
> **Exibição contínua do estado do sistema** (o que ajuda também no caso de os modos serem necessários);
>
> ***Design* consistente**. (!)

Dentre as técnicas adicionais para reduzir erros, estão:

a. **“Casamento” de pares (*matching*)**

Erros comuns podem ser evitados se houver uma estratégia do sistema em orientar o usuário nas situações de pares de elementos.

Exemplos de pares de elementos podem ser os parênteses dentro de uma busca booleana, marcadores de formato em linguagens de marcação.

Computers and (PSICHOLOGY OR SOCIOLOGY)

<B> Isto deve ficar em negrito

O ambiente pode induzir à entrada correta a partir da resposta à solicitação da ação, no primeiro caso, com a exibição dos parênteses esquerdo e direito e o cursor na posição de preenchimento do espaço entre ambos, e, no segundo, de maneira análoga.

Computers and (_)

<B> _</B>

O compromisso desta abordagem na prevenção contar erros é com a flexibilidade. Se se optar pela flexibilidade, deve ser adotada, então, a abordagem de mensagens de alerta (ao se abrir um parêntese ou um marcador, entre outros).

b. Seqüências completas

Muitas vezes a execução de uma tarefa exige vários passos ou comandos. Como o usuário pode desconhecer a necessidade de um dos passos, a inter-relação entre alguns deles, ou simplesmente esquecer-se de um, o designer deve tratar as ações relacionadas de forma completa.

Exemplo de um conjunto de ações relacionadas pode ser o conjunto de ações de edição (Corte, Cópia e Cola). Uma orientação que agrupe e acompanhe a execução seqüencial destas tarefas pode ajudar na prevenção dos erros decorrentes do desconhecimento do funcionamento interno das ações.

Outros exemplos de ações que requerem vários passos são o *loggon* num sistema qualquer e a inserção de arquivos em mensagens de correio eletrônico.

O design de ações integradas não é simples, pois pode ser necessário disponibilizar as ações atômicas de forma independente, fora do ciclo completo que as engloba. Neste caso, pode-se recorrer à ativação e desativação de macros.

Sempre esta necessidade for identificada durante o processo de *design*, o sistema oferecer a possibilidade de definir **planos**, **procedimentos** e **macros** que registrem seqüências de ações normalmente executadas juntas no contexto de aplicação, liberando o usuário da necessidade de memória na solicitação da seqüência em questão.

Durante o processo de *design*, deve-se levantar informação sobre potenciais seqüências completas de ações e/ou comandos, assim como de erros típicos cometidos na sua utilização pelo usuário.

c. Solicitações de ações

Produto industriais bem-sucedidos devem prevenir contra seu uso incorreto, principalmente em casos de produtos do tipo *life-critical*.

Isto é garantido por diversos eletrônicos (entre os quais as câmeras automâticas que, entre outros mecanismos, passam o filme a cada registro, evitando que se queime uma posição pelo *click* sucessivo sem rodar o filme). Os mesmos princípios podem ser aplicados aos sistemas interativos.

Usuários solicitam comandos não-disponíveis e arquivos que não existem, e entram dados inválidos.

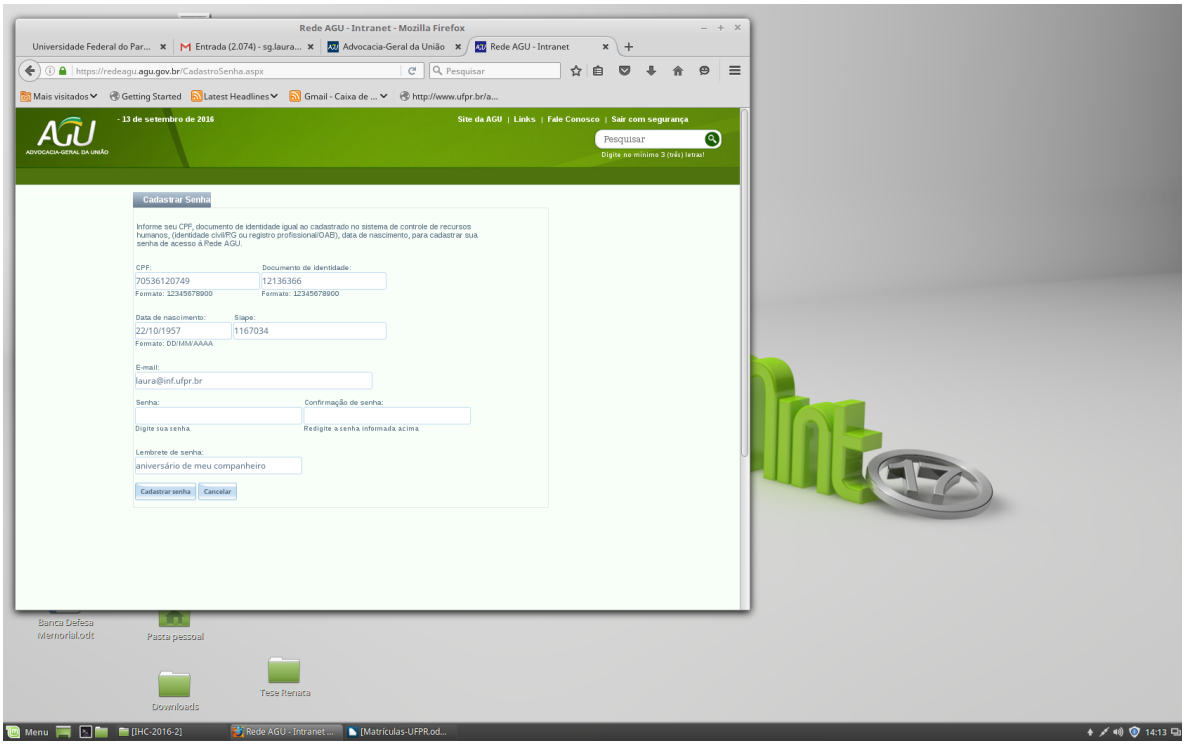
As principais causas destes erros são: digitação, abreviação incorreta, pressão apressada de tecla, engano em relação a um nome.

Uma forma de evitar este tipo de erros é a complementação do nome do comando a partir de uma parte que permite a sua identificação. Em casos de ambigüidade, esta abordagem exige a adoção de *defaults* ou listas de prioridades que podem levar à escolhas indesejáveis.

A adoção do estilo de interação por Manipulação Direta ou, mais precisamente, o paradigma WIMP (*Windows*, *Icons*, *Menus*, *Pointing devices*) evita a maior parte dos erros em questão, na medida em que concentra a interação por menus e por manipulação de representações de objetos na tela.

Além destas técnicas principais de prevenção de erros, há técnicas específicas associadas às diversas classes de erros existentes.

Exemplo de orientação para a prevenção de erros:



Minimização de erros por *slips*

Os tipos mais importantes de *slips* são os erros de captura, de descrição e de modo.

> **Erros de captura** sugerem a necessidade de um *feedback* bem resolvido. Em particular, o *feedback* deve deixar aparente o estado do sistema.
>
> **Erros de descrição** sugerem inconsistência da interface. Estruturas inconsistentes de comandos fazem com que o usuário, ao desconhecer como fazer alguma coisa, atuem por analogia, prejudicada pela inconsistência.

Erros de modo sugerem a necessidade de um *feedback* melhorado. Em particular, o estado do sistema deve ficar sempre revelado e, de preferência, em um local próprio da tela.

Correção de erros

Abordagens na correção de erros

Tão importante quanto minimizar as possibilidades de ocorrência de erros é a garantia de se recuperar deles.

A opção **undo** implementa essa facilidade.

A função *undo* é relativamente fácil de implementar em ambientes de comandos, mas extremamente difícil em interfaces onde um simples evento pode levar a inúmeras ações.

Outros sistemas são estruturados de forma a oferecer **operações inversas** naturais. Por exemplo, “apagar” como operação inversa a “desenhar” em um ambiente de desenho de figuras geométricas.

Segundo (Lewis and Norman 1986) há seis abordagens de sistemas na correção de erros: *gag*, *warn*, *do nothing*, self *correct*, *Let’s talk about it* e *teach me*.

Gag é uma abordagem que previne o erro não deixando que ele continue executando ações erradas. Embora seja “força bruta” esta abordagem é tida pelo usuário como “*A única que funciona*.”

Exemplo desta abordagem é desativar funções ou dispositivos (como o teclado) até que o usuário dê um RESET e o sistema volte ao normal;

Warn é uma mensagem de alerta que resolve algumas situações quando não são fatais. Por isso não são aconselháveis para problemas sérios, até porquê elas podem ser ignoradas pelo usuário. Estas mensagens devem deixar claro o caráter do erro, isto é, o fato de ele não ser fatal.

Donothing é uma filosofia largamente utilizada nas interfaces de manipulação direta. Quando o usuário tenta executar uma ação errada, nada acontece. Esta técnica é baseada na pressuposição de que a resposta do sistema é sempre visível. Assim, esta filosofia só deve ser utilizada quando o resultado das ações é sempre visível na interface, ou, em outras palavras, quando o golfo de avaliação é facilmente passado pelo usuário.

Um exemplo de mau uso desta abordagem está no UNIX, onde a emissão de comandos corretos não tem *feedback* e, para o comando **rcp**, que copia arquivos entre computadores, o sistema não dá mensagem de erro quando emitido com nomes de arquivos inexistentes, limitando-se apenas à não execução da ação.

Self correct é uma abordagem que leva o sistema a realizar uma ação legal próxima daquela que o usuário especificou de forma errada.

Um exemplo simples desta abordagem é a correção de um erro de digitação auto-explicativo.

O problema associado a este comportamento consiste no fato de o sistema às vezes ser exageradamente cooperativo ou “muito espertinho”.

Exemplo de excesso de uso errado da filosofia pode ser visto no Word do Windows 95, quando, ao dar um ENTER imediatamente depois de uma linha que começa com “o”, o sistema assume que o usuário cometeu um erro ao não saber usar os marcadores de itemização e itemiza o texto, frustrando o usuário.

Além disso, esta abordagem não permite que o usuário aproveite a ocasião para identificar o que tinha feito de errado, e minimizar erros subseqüentes.

Let’s talk about it é uma postura que opta por iniciar um diálogo com o usuário, auxiliando-o, por meio de informação, a identificar a causa do erro e corrigi-lo.

Teach me consiste de um refinamento da abordagem anterior, que leva o sistema a interpelar o usuário em relação ao significado de um comando ou expressão. É bastante usada em interfaces envolvendo linguagem natural.

Mensagens de erro

O registro dos erros mais freqüentes permite revisar as formas de prevenção de erros, os procedimentos de recuperação, a documentação e os manuais, alterar o auxílio online, ou, até, modificar a disponibilidade de ações no sistema.

O conjunto completo de possibilidades e de mensagens de erro correspondentes deve ser revelado na documentação ao usuário.

A melhoria das mensagens de erro constitui a forma mais fácil e eficiente de se melhorar a qualidade de um sistema existente.

Em resumo:

orientação construtiva,

tom positivo,

estilo centrado no usuário,

e formato físico apropriado

são as bases para elaborar mensagem de erros apropriadas.

Alguns exemplos comuns ilustram claramente estas necessidades.

Mensagens de erros tradicionais, tais como:

SYNTAX ERROR ou

ILEGAL DATA

WHAT?

são, além de vagas, hostis e devem ser substituídas por mensagens de erro mais específicas, de tom positivo e construtivas, tais como:

UNMATCHED LEFT PARENTHESIS ou

MENU CHOICES ARE IN THE RANGE OF 1 TO 6

Além disso, mensagens em tom imperativo aumentam a ansiedade do usuário e reduzem, assim, as chances de ele se recuperar do erro cometido.

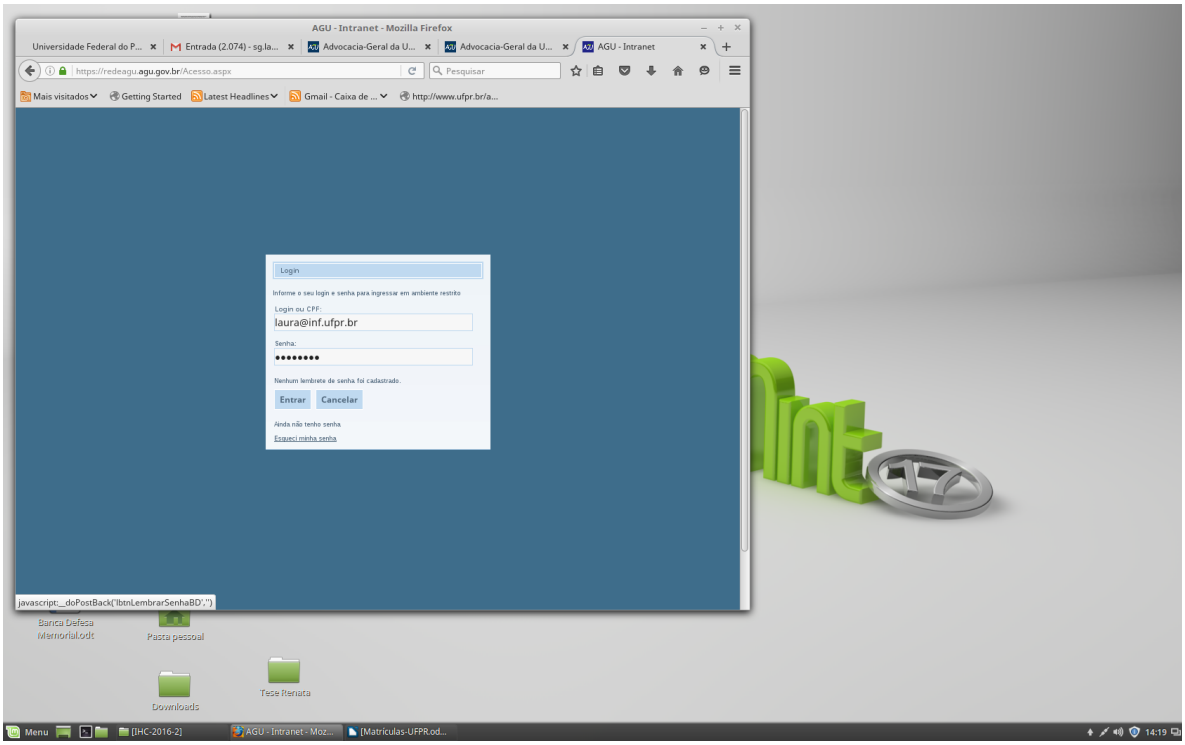
Segundo Nievergelt & Weydert (1980), o usuário precisa dispor da seguinte informação para se recuperar de um erro:

1. **o estado corrente do sistema** (relativo ao resultado da última ação e às alternativas disponíveis);
2. **donde ele veio** (não-determinístico, exige registro de ações);
3. **quais as alternativas disponíveis**.

Em alinhamento à afirmação destes autores, e com o intuito de serem cooperativas, as mensagens de erro devem informar:

1. **o erro ocorrido** (de forma clara e precisa);
2. **os possíveis motivos**;
3. **as formas possíveis de se recuperar da situação de erro**.

Exemplo de situação de impasse:



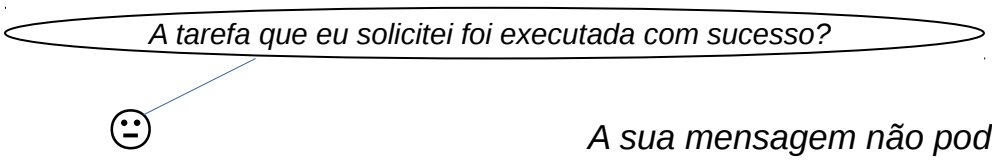
Um problema sério nas mensagens de erro ocorre quando a informação fornecida pelo sistema corresponde ao nível de execução, quando o usuário pensa no nível da intenção.

O problema está em que, em alguns casos, esse nível de mensagem é apropriado e em outros não.

A questão que o *designer* deve se fazer é:

> "*Qual é a informação útil para o usuário neste momento*?".

Certamente, o usuário necessita, por exemplo, saber se a tarefa que ele solicitou que o sistema executasse, foi executada com sucesso ou não (por exemplo, se o envio da mensagem foi efetivado ou não). Adicionalmente, é preciso proporcionar informação no nível em que o erro ocorreu, seja ele, conceitual, de realização ou físico, principalmente quando isso exigir alguma ação do usuário.



😐	*A sua mensagem não pode ser enviada. Tente novamente.*
(nível da execução*)	*Servidor da rede fora do ar. Contate o suporte técnico.*

Exemplo de níveis de informação necessários às mensagem de erro

***** *Em quais situações este nível de informação é necessária?*

Outro (contra)exemplo:

De: laura@inf.ufpr.br
Data: 13 de setembro de 2016 11:35
Assunto: Solicitação de nova senha de acesso ao simulador
Para: <ouvidoriageral@agu.gov.br>

Prezados senhores,

Na tentativa de fazer uma simulação de tempo faltante de contribuição para a aposentadoria, entrei no site da AGU e verifiquei que já tinha me cadastrado. No entanto, não lembro da senha e não consegui uma maneira de solicitar uma nova para poder efetivar a minha entrada no simulador.

Agradeço a colaboração.
Laura Sánchez García (CPF:..)

---------- ----------
De: <ouvidoriageral@agu.gov.br>
Data: 14 de setembro de 2016 09:06
Assunto: AGU Ouvidoria: Resposta da demanda 005707/2016-88
Para: laura@inf.ufpr.br

AGU Ouvidoria

Resposta da demanda 005707/2016-88

Prezado(a) Senhor(a), A Ouvidoria da Advocacia-Geral da União (AGU) agradece o envio da sua mensagem. Temos a informar que, de acordo com o artigo 131 da Constituição Federal, a Advocacia-Geral da União (AGU) é responsável pela defesa judicial e extrajudicial da União (Poderes Executivo, Legislativo e Judiciário), além de prestar assistência e consultoria jurídica aos órgãos do Poder Executivo Federal. A Ouvidoria da Advocacia-Geral da União é um canal aberto ao público interno (servidores das áreas administrativa e jurídica da AGU) e público externo (cidadãos e sociedade em geral), que possui a incumbência de receber reclamações, elogios, críticas, sugestões e denúncias relativas aos serviços prestados pela AGU, não tendo, portanto competência para atuar na demanda proposta por V. Sa. Sendo o que havia para o momento, colocamo-nos ao seu permanente dispor para apresentação de futuras demandas.

Atenciosamente, Ouvidoria da AGU Advocacia-Geral da União.

Nível de especificidade

As mensagens de erro devem ser tão objetivas e **específicas** quanto possível.

Os exemplos a seguir mostram mensagens comuns e formas mais apropriadas.

SYNTAX ERROR	unmatched left parenteses
ILLEGAL ENTRY	Type first letter: Send, Read,...
INVALID DATA	Days range from 1 to 31
BAD FILE NAME	File name must begin with a letter

Considere-se um sistema de *check-in* de um hotel, que exige que o funcionário entre uma cadeia de caracteres separados por vírgulas, incluindo: nome, habitação, cartão de crédito, e demais informações sobre o hóspede, e fornecesse mensagens de erro do tipo:

ILLEGAL DATA. RETYPE THE ENTIRE RECORD

Estas mensagens certamente frustram o usuário, por serem genéricas e não permitirem a correção específica da parte da entrada com erro.

Um sistema bem projetado deve minimizar os erros de entrada por meio de técnicas apropriadas de preenchimento de formulários, e permitir que o usuário necessite corrigir apenas o campo específico que ele tinha errado.

Sistemas que oferecem ao usuário os **códigos de erro** também são impróprios, pela indireção.

O usuário pode não ter o manual à sua disposição, ou, mesmo tendo-o , a leitura do par[agrafo correspondente pode ser longa e levar mais tempo do que o que o usuário tem disponível.

Orientação construtiva e tom positivo

Ao invés de condenar o usuário pelo que ele fez, um bom sistema deve orientá-lo no sentido de ele executar as suas tarefas da forma correta.

Seguem alguns exemplos de mensagens reais inadequadas e da correspondente modificação.

DISASTROUS STRING OVERFLOW. JOB ABANDONED	String space consumed. Revise program to use shorter strings or expand string space
UNDEFINED LABELS	Define statement labels before use
ILLEGAL STA. WRN.	Return statement cannot be used in a function subprogram

Mensagens hostis não têm justificativa. Seguem alguns exemplos retirados de sistemas reais.

FATAL ERROR, RUN ABORTED

CATASTROFIC ERROR; LOGGED WITH OPERATOR

Termos tais como: ILLEGAL, INVALID, BAD devem ser evitados.

Mensagens excessivamente informais (pelo menos para alguém da minha geração), como “*Olá, Laura!*”, tampouco são apropriadas.

Redação centrada no usuário

As mensagens do sistema devem ser centradas na ação a ser executada pelo usuário, e não no fracasso da execução da ação solicitada do ponto de vista do sistema.

Illegal phone number. Call aborted. Error number 583-2R6.9 Consulte your user manual for further information.	We are sorry, but we were unable to complete your call as dialed. Please, hang up, check the desired number, and, consult the operator for further assistance.

Formato, espaço e apresentação apropriados

Há alguns aspectos ligados ao formato da apresentação da informação que podem melhorar substancialmente a qualidade das mensagens de erro.

Quando a inclusão do **código de erro** se justificar, este não deverá iniciar a mensagem, mas, sim, ser incluído ao final, bem caracterizado, e entre parênteses.

MAIÚSCULAS devem ser reservadas para mensagens breves e sérias.

Há três abordagens para a **localização das mensagens de erro**:

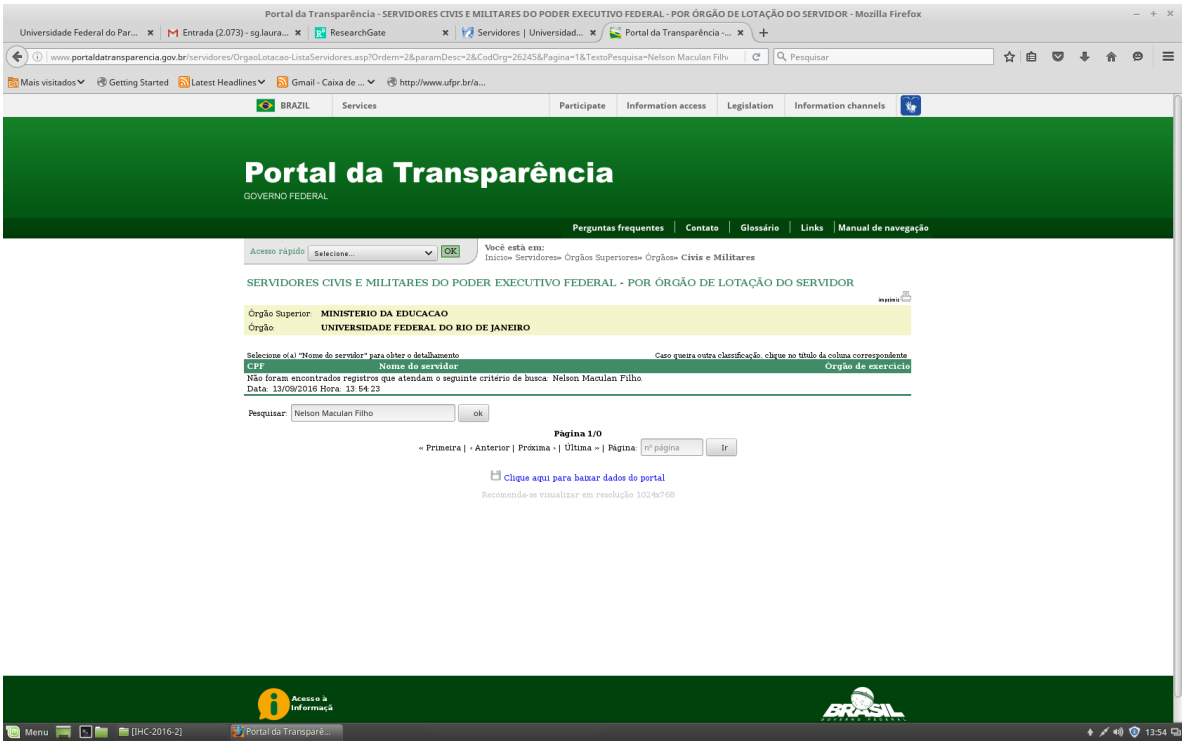
1. Uma vertente sustenta que as mensagens devem ser situadas fisicamente perto do lugar onde o erro ocorreu. Esta abordagem vem sendo largamente usada em ambientes de compiladores.

2. Outra sustenta que as mensagens distraem a leitura do todo e devem Ter um espaço consistente específico ao pé da tela.

3. A terceira abordagem abre uma janela de diálogo, móvel, provavelmente encobrindo uma parte substancial da informação presente na tela.

A adoção de sons para chamar a atenção em relação ao erro cometido é interessante, mas pode embaraçar o usuário. Portanto, este tipo de recurso deve poder ser controlado pelo usuário.

Outra questão importante é a da apropriação de recursos gráficos para chamar a atenção para a mensagem de erro.



Situações de *breakdown* determinadas pela aplicação

São situações de corte n comunicação usuário-sistema em tempo de uso que, em havendo um adequado processo de design, podem ser evitados.