UNIVERSIDADE FEDERAL DE SANTA CATARINA
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO
ÁREA DE SISTEMAS DE CONHECIMENTO

MARCOS DOUGLAS DA SILVA GOMES

Interfaces Ergonômicas Para Software Educativo

Desenvolvimento de ferramentas de concepção e avaliação a partir da mesma base de conhecimento

Dissertação submetida à Universidade Federal de Santa Catarina como requisito final para a obtenção do grau de Mestre em Ciência da Computação área de Sistemas de Conhecimento, ênfase em informática Educativa

Orientador: Prof. Dr. Walter de Abreu Cybis

Florianópolis, outubro de 2002

Interfaces Ergonômicas Para Software Educativo

Desenvolvimento de ferramentas de concepção e avaliação a partir da mesma base de conhecimento

Marcos Douglas da Silva Gomes

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Conhecimento, ênfase em Informática Educativa e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Fernando Álvaro Ostuni Gauthier
Coordenador do Programa

Banca Examinadora:

_____

Prof. Dr. Walter de Abreu Cybis
Orientador

_____

Profa. Dra. Edla Maria Faust Ramos

_____

Prof. Dr. Leandro José Komosinski

“Eu creio que se nós, como povo, fôssemos educados para a tolerância recíproca, para o respeito à autoridade, para o trabalho persistente, sem conflitos entre empresários e trabalhadores, se nós todos nos uníssemos para compreender as necessidades desses valores espirituais na vida de cada um ou de cada grupo social, nós teríamos um país extremamente venturoso”

(Emmanuel)

DEDICATÓRIA

Dedico este trabalho a meus pais, Benedito Corrêa Gomes e Nadir Barata da Silva; a meus irmãos, Valério Alan, Marcelino Augusto e Regina Márcia, pois nos momentos difíceis sempre encontro paz e equilíbrio forjados no seio familiar.

Dedico à minha esposa, Rosa Helena Oliveira, pelas incontáveis e, principalmente, incansáveis horas de nossas vidas consumidas na construção deste trabalho.

AGRADECIMENTOS

- A Deus, por todos os momentos bons e por todos os que achamos ser ruins;
- Ao Prof. Dr. Walter de Abreu Cybis, pela verdadeira orientação prestada, sempre de forma sincera e tranqüila, mesmo em seus momentos mais complicados;
- À Profa. Dra. Edla Maria Faust Ramos, pela grande colaboração e compreensão durante sua orientação;
- À Profa. Conceição Rangel Fiuza de Melo por ser a "culpada" de minha inserção na vida acadêmica e ter apostado em minha capacidade.
- À profa. Ms. Elisa Maria Gomes Schiocchet por seu incansável incentivo e dedicação, durante e após a Coordenação do Mestrado, em Belém.
- Ao Centro de Ensino Superior do Pará - CESUPA pelo apoio estrutural e financeiro disponibilizados na realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho aborda interfaces ergonômicas para software educativo, mais especificamente focado nos processos de concepção e avaliação, objetivando proporcionar uma interação mais eficiente, eficaz e confortável em ambientes educacionais informatizados. A partir da fundamentação teórica foi possível a análise de diversas ferramentas desenvolvidas para este objetivo, tais como: guias de estilo, guias de recomendações e técnicas de inspeção ergonômicas. Tais ferramentas são concebidas a partir de uma base específica de conhecimentos para cada ambiente analisado, essa característica não permite que haja uma sincronização entre a concepção e a avaliação, causando uma possível diminuição na qualidade da interface. A pesquisa propõe a utilização, durante o ciclo de vida de usabilidade para software educativo, de ferramentas desenvolvidas para orientar tanto a concepção como a avaliação da interface concebida a partir da mesma base de conhecimentos, a fim de que possam atuar sob as mesmas características e critérios, proporcionando um aumento na qualidade da interface. A pesquisa desenvolve, ainda, um guia de recomendações ergonômicas que utiliza a mesma base de conhecimentos encontrada em uma técnica de inspeção ergonômica concebida para ambientes educacionais informatizados.

O referido guia apresenta um total de vinte e três critérios, com recomendações comentadas, exemplos e referências bibliográficas.

ABSTRACT

This work shows the ergonomical interface for educational software's, more specifically focused in the conception and evaluation process, objecting an efficient, effective and comfortable educational computerized environment. From the scientifical studies of theoretical subjects was possible to analyze the different tools developed for this aim, as well as: stylish guides, technical guides for recommendations in ergonomics inspections. That tools are created from a specific knowledge base for each analyzed area, and this characteristic does not allowed an synchronization between the conception and the exams, caused by a possible shorten of quality of the interface. The research proposes an utilization of tools developed to orient the conception of how the interface exams works from the same knowledge base to keep the original criteria and characteristics during the life cycles of the educational software, to offer an increment in quality of the interface. The research develops a recommendation of ergonomics that uses the same base found in a technical ergonomics inspection created for computerized educational environment. The referred guide shows twenty-three criteria's with commented recommendations, examples and bibliographical references.

CAPÍTULO I

1. INTRODUÇÃO

Tendo a Educação um papel importante na formação e no desenvolvimento de uma sociedade e percebendo que a mesma passa por um processo informacional, seja de forma imposta ou reconhecendo a necessidade da utilização de novas tecnologias, se faz necessário um criterioso estudo nos diversos aspectos e características próprias do ambiente educacional que devem ser valorizadas neste processo.

Devem ser considerados aspectos como: a teoria pedagógica apropriada para os diversos tipos de ambientes a serem desenvolvidos, o devido tratamento para os erros durante a interação do usuário, a concepção de ambientes cooperativos, ou seja, o desenvolvimento de interfaces ergonômicas que colaborem para o processo de aprendizagem, entre outros.

De maneira geral, existem diversas propostas com o objetivo de tratar a qualidade ergonômica de software, algumas estão voltadas para a concepção e outras para a avaliação; contudo, ambos os casos não são apropriados para ambientes educacionais se não levarem em consideração suas particularidades. Nestes casos a interface tem função vital, uma vez que ela pode interferir no processo de ensino-aprendizagem. Como exemplo, pode-se citar o tratamento de erros: se ele não for bem concebido, pode fazer com que o usuário (aluno) desista de continuar a interagir com a aplicação.

Entre as técnicas voltadas para a concepção de interface pode-se destacar o guia de recomendações e o guia de estilo, com a função de conduzir a elaboração de interfaces ergonômicas. No caso de avaliação ergonômica, encontram-se técnicas prospectivas, técnicas analíticas, técnicas empíricas que, se aplicadas em conjunto, têm maior eficiência no processo avaliativo.

O que se observa é que técnicas de concepção e técnicas de avaliação trabalham baseadas em conhecimentos distintos, ou seja, cada uma delas é elaborada a

partir de uma **base de conhecimento – Conjunto de critérios que caracteriza a especificidade ergonômica da interface –** própria gerada a partir de contextos específicos que diferem entre si, causando uma descontinuidade nas fases de elaboração e avaliação de uma determinada interface.

Essa independência de base de conhecimentos pode gerar uma interface baseada em um grupo específico de critérios e que, ao ser avaliada com base em um grupo de critérios diferenciados, poderá ter uma extensa parte de seu projeto reprovada ou não considerada pelos avaliadores.

Essas observações serão constatadas através da análise de duas técnicas desenvolvidas, tendo como foco softwares educativos:

- GEPESE (Guia de recomendações específico para software educacional) (VALIATI *et al.*, 2000).
- TICESE (Técnica de inspeção de conformidade ergonômica de software educacional) (Games, 1998).

Sob o ponto de vista da engenharia do processo, a falta de sincronicidade é preocupante, a despeito de saber qual o conjunto de critérios seria o mais adequado. Tal desconformidade pode prejudicar sua etapa de avaliação forçando revisões sucessivas em seu ciclo de vida.

1.1. Definição do tema e do tipo de pesquisa

Desenvolvimento de um Guia de Recomendações para conduzir o projeto de elaboração da interface ergonômica de softwares educativos, utilizando a mesma base de conhecimentos de uma Técnica de Inspeção Ergonômica.

A pesquisa pode ser classificada como teórica, pois pretende realizar um estudo das condições necessárias para se estabelecer a sincronia entre as etapas de desenvolvimento e avaliação de projetos para desenvolvimento de interfaces de aplicativos educacionais, tendo como base o levantamento do estado da arte envolvendo técnicas de concepção/condução e avaliação ergonômicas.

1.2. Panorama atual

Observa-se atualmente um aumento de inserção das novas tecnologias em diversos contextos sociais causado pelo processo tecnológico que se lança no mundo.

Contudo, as dimensões de acesso em que se processam tais avanços são totalmente díspares quando comparados entre si. Esta situação é observável, por exemplo, nas telecomunicações, onde o aparato tecnológico permite que se tenha informações de um determinado fato no exato momento de seu acontecimento, independente de onde o mesmo está ocorrendo. Na área médica, os avanços atuais permitem diagnósticos precisos e antecipados. Em contrapartida, alguns setores da sociedade de igual importância vêm sofrendo um lento desenvolvimento nessa área, como exemplo a Educação.

É próprio considerar-se como fundamental para o desenvolvimento de uma sociedade setores vitais como saúde, telecomunicações e educação, entre outros. Partindo dessa premissa, uma longa lista de perguntas permeia a discussão sendo, talvez, uma das mais relevantes perceber qual o motivo que leva a Educação a um plano secundário, uma vez que esta concentra em suas raízes o pilar de qualquer sociedade.

Diversas tentativas, nesse sentido, vêm sendo feitas por profissionais das mais diferentes áreas do conhecimento: Pedagogia, Psicologia, Filosofia e, entre elas, a Informática. Prova disso está no elevado número de aplicativos e produtos educacionais que surgiram e surgem ao longo dos últimos anos, com o objetivo de promover um maior e melhor desempenho na aprendizagem de conceitos nas mais diversas áreas do conhecimento, desde as primeiras séries do ensino básico até o ensino superior e pela educação especial.

Possivelmente, uma maior divulgação dos resultados obtidos em pesquisas relacionadas à área de informática educativa, possa expressar a real situação que se encontra seu processo de expansão tecnológica.

A questão é: como estão sendo desenvolvidos estes produtos e que nível de qualidade apresentam? E aqui cabem diversas observações. Os tópicos a seguir não indicam ordem de prioridade, apenas estão organizadas em tópicos para melhor compreensão.

- O nível de compromisso pedagógico do produto – diversos produtos são encontrados no mercado, seja em prateleiras de lojas especializadas, ou desenvolvidos para atender um determinado objetivo, com a titulação de software educativo. Nesse universo de programas, uma pequena parcela prioriza o processo

pedagógico tendo como objetivo primordial a aprendizagem ser alcançada pelo usuário.

- Questão financeira – outro ponto que merece relevância é a baixa qualidade do produto em função da redução de custos. Para que os custos no desenvolvimento de software não sejam elevados abre-se mão de uma equipe multidisciplinar de profissionais, de recursos tecnológicos adequados, de ambientes de desenvolvimento apropriados.

- Conhecimento pedagógico aliado a conhecimento técnico – se faz necessário que haja uma aproximação das duas grandes áreas que envolvem esse processo. Por um lado se observa um grande contingente de técnicos e projetistas, capazes de desenvolver ferramentas computacionais com recursos de alta qualidade, por outro lado, pedagogos e estudiosos da Educação cada vez mais discutem e fundamentam processos e teorias para tentar obter avanços na qualidade da educação como um todo. O esforço na direção de um trabalho conjunto é relativamente recente e já se observam bons frutos, demonstrando as possibilidades desse tipo de trabalho.

- Falta de critério técnico no desenvolvimento – outro fator que pode e deve ser levado em conta diz respeito à qualidade de software quanto à usabilidade – utilização do sistema. O que se observa é um zelo no aspecto funcional do produto, porém um descuido no que se refere à qualidade da interface em função do usuário final, principalmente em produtos com fins educacionais, os quais podem ter seus objetivos pedagógicos seriamente comprometidos em função de uma interface mal elaborada.

O tema em questão envolve diversos aspectos além dos acima citados e se faz necessária uma ampla discussão, a fim de se obter uma maior conscientização por parte dos envolvidos no processo de criação e desenvolvimento de software.

Dentre esses fatores, dar-se-á enfoque à questão da interface e de seus aspectos ergonômicos – Ergonomia de Interfaces Humano-Computador (IHC). De acordo com WISNER *apud* CYBIS (1997):

> *"A ergonomia é o conjunto de conhecimentos científicos relativos ao homem e necessários à concepção de instrumentos, máquinas e dispositivos que possam ser utilizados com o máximo de conforto, segurança e eficácia"*.

A análise da interface baseada em estudos ergonômicos, de um modo geral, vem despertando o interesse de diversos pesquisadores nas últimas décadas, aumentando tanto em abrangência como em profundidade o nível da discussão. Os principais focos das pesquisas apontam para as seguintes direções:

1. Fundamentos da ergonomia de IHC – que abrange:

- Bases teóricas referentes à:
- Psicologia cognitiva, onde são consideradas "*...as características humanas no tratamento da informação, ligadas as suas habilidades e capacidades em termos cognitivos*", CYBIS (1997).
- Semiótica - que estuda as características dos tratamentos simbólicos, tanto do ser humano como do computador e o modo como esses simbolismos se relacionam.
- Bases metodológicas, que através da ergonômica do trabalho buscam identificar os pontos a serem observados sobre os tipos de tarefas e atividades a serem executadas levando em consideração seu funcionamento e utilização.

2. Critérios e componentes de ergonomia – formam as bases do conhecimento necessário para a elaboração e concepção de interfaces ergonômicas.
3. Técnicas de projetos – São responsáveis pelo processo de desenvolvimento, conduzindo cada uma das fases na elaboração de IHC – análise, concepção, projeto, desenvolvimento, implantação, teste e manutenção.
4. Técnicas de concepção/condução ergonômica.
5. Técnicas de avaliação ergonômica – São utilizadas para possibilitar a avaliação da qualidade ergonômica da interface.
6. Aplicações ergonômicas – Introdução da ergonomia de IHC nos mais diversos tipos de software; aplicativos comerciais, jogos, software educativo, entre outros.

A engenharia de usabilidade busca pesquisar, aplicar e analisar métodos, procedimentos e ferramentas no sentido de melhorar a interação do usuário com o ambiente, aumentando as possibilidades de sucesso no que se refere a conforto, segurança, eficácia e eficiência, em produtos informáticos, sejam estes educacionais ou não.

O desenvolvimento de interfaces ergonômicas, quando realizado de maneira eficiente, pode transmitir ao usuário clareza e segurança necessárias para garantir a interação com o ambiente de qualquer aplicação. No caso de ambientes educacionais sua importância se acentua no fato de que, além de garantir uma boa interação entre usuário e software, a ergonomia de IHC tem uma forte ligação com o processo de aprendizagem, o qual pode ser seriamente comprometido se não houver a devida atenção aos critérios ergonômicos.

Do ponto de vista pedagógico, corre-se o risco de comprometer o objetivo final, que vem a ser o aprendizado de um dado conteúdo, se o produto em questão não for desenvolvido com uma análise ergonômica criteriosa.

CYBIS (*apud* GAMEZ, 1998) ao abordar a qualidade ergonômica de software educativo, afirma:

> *"Um software educacional possui características peculiares e sua utilidade é avaliada através de sua didática e pedagogia. Ele deve ser inserido em um contexto de aprendizado pré-definido e proporcionar autonomia, cooperação, criatividade, pensamento crítico, descoberta e construção de conhecimento. Dessa forma, além de ser intuitivo, fácil e eficiente de usar, o software educacional deve ser didático".*

A inserção da informática na educação, além de sua utilização como ferramenta capaz de aumentar a rapidez no processamento e permitir o armazenamento de um grande número de informações carrega consigo uma possibilidade muito peculiar, que é proporcionar aprendizagem utilizando-se de aplicações como forma de mediação no processo educacional. É nessa perspectiva que se deve concentrar esforços de profissionais em informática, professores, educadores e demais envolvidos, afim de que seja obtido o máximo de seu desempenho para que não se corra o risco de mais um fracasso tecnológico educacional. Essa afirmação é respaldada por HACK *et al* (1999) quando afirma que:

> *"É fundamental que se realize uma reflexão profunda de todos os aspectos envolvidos na relação pedagógica, tendo o computador como ferramenta de mediação que possibilita a troca generalizada de saberes. É necessário ter claro qual a concepção de aprendizagem que está por detrás, ao fazer uso dessa tecnologia e, muito mais do que isso, é preciso que os educadores, professores, vivenciem o uso dessa tecnologia na perspectiva de reconhecimento autogerenciado, móvel e contextual das competências".*

Com base nessas considerações, o presente trabalho é delimitado pelos seguintes objetivos:

1.3. Objetivo Geral

- Elaborar um ambiente de Engenharia de Usabilidade para Software Educacional.

1.4. Objetivos Específicos

O objetivo geral pode ser desdobrado nos seguintes objetivos específicos:

- Elaborar um guia de recomendações para o desenvolvimento de interfaces ergonômicas de software educativo.
- Utilizar a mesma base de conhecimentos aplicada na elaboração da TICESE.
- Estabelecer a sincronicidade entre o guia de recomendações a ser elaborado e a técnica de inspeção (TICESE) a partir do mesmo grupo de critérios utilizados.

1.5. Contexto do Problema

A qualidade é fator importante na aceitação de produtos e serviços que buscam suprir as mais diversas necessidades do homem. Cada vez mais se observa um movimento no sentido de buscar não somente a solução para um determinado problema, objetiva-se também obter qualidade, contemplando tanto a eficácia como a eficiência no resultado final.

O desenvolvimento de ambientes informacionais, sejam eles produtos ou serviços, vêm sofrendo alterações no decorrer de sua evolução, apresentando um processamento cada vez mais rápido e robusto em ambientes mais amigáveis e

interativos, com a utilização de recursos gerados pela multimídia, buscando satisfazer o usuário final com um software fácil e agradável de ser usado.

Contudo, ao se fazer uma análise mais criteriosa desses softwares, é provável que se perceba o aparecimento de algumas inconsistências que podem causar uma queda de rendimento no desempenho do processo, tais como: ações redundantes, desvio de conduta, tratamento de erros de forma inconsistente, interfaces ergonomicamente incorretas, entre outros. Quase sempre os critérios utilizados são gerais, permitindo o surgimento de falhas em itens específicos que requerem tratamentos diferenciados.

O trabalho em questão tem como campo de pesquisa o ambiente educacional e como foco a produção de software educativo, mais precisamente os processos de concepção e avaliação da ergonomia de interface desses produtos.

1.6. Problematização

Ao observar as diversas técnicas, tanto de concepção, através da utilização de guias de recomendações ou de estilo, como de avaliação, com a aplicação de checklists para interfaces ergonômicas, constatou-se que, na maioria dos casos, a base de conhecimento que as geram são diferenciadas em diversos critérios de análise, podendo gerar resultados insuficientes na verificação de aplicabilidade e conformidade das recomendações envolvidas.

Tal situação é descrita por VALIATI *et al.*(2000), ao propor um guia de recomendações específico para software educacional. Após o desenvolvimento e a aplicação do guia de recomendações em um determinado software, foi realizada a avaliação através da aplicação da TICESE. Os resultados obtidos não foram satisfatórios em relação à conformidade da interface. Este exemplo será mais bem detalhado posteriormente.

É possível supor que, se a base de conhecimento for a mesma utilizada tanto na concepção como na avaliação, o resultado terá seu nível de acerto melhorado, descartando possíveis critérios incompatíveis. Com isso pode-se ter um ciclo de vida para o desenvolvimento de interfaces mais curto e preciso, gerando vantagens em diversos segmentos do projeto.

1.7. Metodologia

A pesquisa terá sua fundamentação a partir de estudos bibliográficos envolvendo os aspectos relativos à engenharia de software, usabilidade, ciclo de vida da engenharia de usabilidade, bem como as diversas técnicas de concepção e avaliação de interfaces para software educativo.

a) Seleção da Técnica de Inspeção Ergonômica.

Será selecionada uma técnica de inspeção ergonômica para ambientes educacionais informatizados para que sua base de conhecimentos seja utilizada no desenvolvimento de um guia de recomendações.

b) Desenvolvimento do Guia de Recomendações

Durante seu desenvolvimento, será observada a sincronia entre guia de recomendações e técnica de inspeção, com o objetivo de manter a mesma base de conhecimentos entre ambos.

1.8. Apresentação da dissertação

A dissertação está dividida em seis capítulos descritos a seguir:

a) Introdução

Contém a definição do tema e o tipo de pesquisa, apresentando um panorama geral sobre o desenvolvimento de software educacional e suas principais questões na busca da qualidade para produtos educacionais informatizados, além dos objetivos da pesquisa, sua contextualização e relevância.

b) Engenharia de Software

Este capítulo apresenta os elementos fundamentais que compõem a engenharia de software e seus principais tipos de ciclo de vida, além de algumas ferramentas utilizadas para apoiar o desenvolvimento de sistemas informatizados tais como guias de estilos, guias de recomendações e checklists.

c) Usabilidade

Neste capítulo é apresentada a definição de usabilidade, além de seus requisitos e critérios de qualidade de acordo com o tipo de usuário. Apresenta, ainda, a evolução da interface do usuário desde a década de quarenta até os dias atuais e suas principais características de usabilidade.

d) Engenharia de Usabilidade

Neste capítulo são apresentados a engenharia de usabilidade e seu modelo de ciclo de vida, além dos principais estágios que envolvem este ciclo durante a elaboração de um produto de software.

e) Desenvolvimento do Guia de Recomendações Ergonômicas para Interfaces de Software Educativo

Este capítulo apresenta a descrição de todo o processo de desenvolvimento da dissertação, desde a detecção do problema até a solução proposta.

Contém a descrição do estado da arte que envolve ferramentas de concepção e avaliação ergonômicas de interfaces para ambientes educacionais informatizados, além da análise de alguns dos principais guias de estilo, guias de recomendações e checklists existentes e suas contribuições para a pesquisa.

Apresenta a estratégia adotada para solucionar o problema e suas justificativas e a descrição da metodologia utilizada na elaboração do Guia de Recomendações proposto.

f) Conclusão e Trabalhos Futuros

São traçadas as considerações finais sobre a pesquisa, enfocando suas dificuldades e limitações, além de propor trabalhos futuros que possam dar prosseguimento à linha de pesquisa.

g) Guia de Recomendações Ergonômicas Para Software Educativo

O Guia de Recomendações Ergonômicas Para Software Educativo é parte integrante deste trabalho e apresentado na forma de anexo.

CAPÍTULO II

2. ENGENHARIA DE SOFTWARE

O atual estágio em que se encontra o desenvolvimento de software, tendo suas aplicações, em um curto espaço de tempo, penetrado nas mais diversas áreas de atuação, desde aplicações comerciais até o campo da Medicina passando pela Educação e, mais recentemente, no ambiente doméstico, gera uma questão que a cada dia se torna fundamental: qual o nível de qualidade em um software?

Diante da produção atual de software, o grande diferencial pode ser a qualidade em uso. Essa qualidade visa garantir, entre outros quesitos, a eficácia, eficiência e o conforto do usuário, satisfazendo suas necessidades explícitas e implícitas.

PAULA FILHO (2001) afirma que "*geralmente a qualidade de um produto decorre diretamente da qualidade do processo efetivamente utilizado na produção deste*". Sendo assim, a qualidade do software depende diretamente da qualidade de seu processo de elaboração.

Durante muito tempo o software foi produzido de maneira a satisfazer uma aplicação específica e sob medida para realizar uma determinada tarefa, o que levava a uma distribuição bastante limitada do produto. Seu desenvolvimento era realizado quase que de forma empírica e sem nenhuma metodologia sistematizada, gerando um processo mais próximo da arte do que da ciência.

A Engenharia de Software surge com o objetivo de garantir a qualidade em um processo sistematizado de elaboração de software unindo métodos mais eficazes e abrangentes de desenvolvimento, ferramentas capazes de automatizá-los, técnicas que garantam a qualidade do produto, aliada a uma filosofia de coordenação, controle e administração do processo.

As primeiras práticas da Engenharia de Software apareceram durante a década de 80, quando surgiram questões relativas à produção de software. Alguns

problemas como a demora na conclusão de programas, custos elevados, falhas após a entrega da versão definitiva, entre outros, despertaram nos profissionais a necessidade de desenvolver processos sistematizados, mais eficientes, capazes de garantir qualidade ao produto de software.

De acordo com PRESSMAN (1995), a primeira definição de Engenharia de Software proposta por Fritz Bauer afirma que: "é *o estabelecimento e o uso dos sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais*".

Basicamente a Engenharia de Software é composta de três elementos fundamentais – métodos, ferramentas e procedimentos – responsáveis pelo desenvolvimento de software com qualidade.

Os ***métodos*** são responsáveis pelo detalhamento do processo de "como fazer" um software partindo de importantes tarefas que englobam, segundo PRESSMAN(1995): "*planejamento e estimativa do projeto, análise de requisitos, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção*".

As ***ferramentas*** proporcionam apoio aos métodos, atuando em cada uma das tarefas citadas anteriormente. Como exemplo têm-se as ferramentas CASE – Computer-Aided Software Engineering, ou seja, Engenharia de Software Auxiliada por Computador, onde são estabelecidas relações entre cada ferramenta, promovendo uma integração de modo que a informação gerada por uma será usada por outra, alimentando um sistema de suporte durante o desenvolvimento.

Os ***procedimentos*** funcionam como a ligação entre os métodos e as ferramentas, definindo: a seqüência em que os métodos serão aplicados, quais os produtos serão entregues, quais os controles serão utilizados para assegurar a qualidade e controlar as possíveis mudanças e quais os pontos de referência na avaliação do progresso do desenvolvimento.

Como o objetivo é desenvolver processos capazes de conduzir a produção de software, seja ele voltado a um determinado cliente ou um software de prateleira para ser comercializado no mercado aberto, a Engenharia de Software propõe paradigmas que envolvem métodos, ferramentas e procedimentos. Esses paradigmas são também denominados de ciclos de vida.

> "*Como todo produto industrial, o software tem um ciclo de vida: ele é concebido a partir da concepção de uma necessidade; é desenvolvido, transformando-se em um conjunto de itens entregue a um cliente; entra em operação, sendo usado dentro de algum processo de negócio e sujeito a atividades de manutenção; é retirado de operação, ao final de sua vida útil*".(PAULA FILHO, 2001).

O Ciclo de Vida em Engenharia de Software deve considerar o objetivo do projeto e da aplicação, os métodos e as ferramentas adequadas para cada situação e as principais características do produto a ser entregue. Os principais paradigmas para os Ciclos de Vida serão apresentados a seguir.

2.1. Tipos de Ciclo de Vida

- Ciclo de vida clássico ou em cascata
- Modelo de prototipação e teste
- Modelo espiral
- Modelo de prototipagem evolutiva
- Técnicas de quarta geração

2.1.2. Ciclo de Vida Clássico ou Modelo em Cascata

Este modelo necessita de uma abordagem sistemática, seqüencial ao desenvolvimento do software, tem em sua divisão as seguintes etapas: análise e engenharia de sistemas, análise de requisitos de software, projeto, codificação, teste e manutenção (Fig. 1).

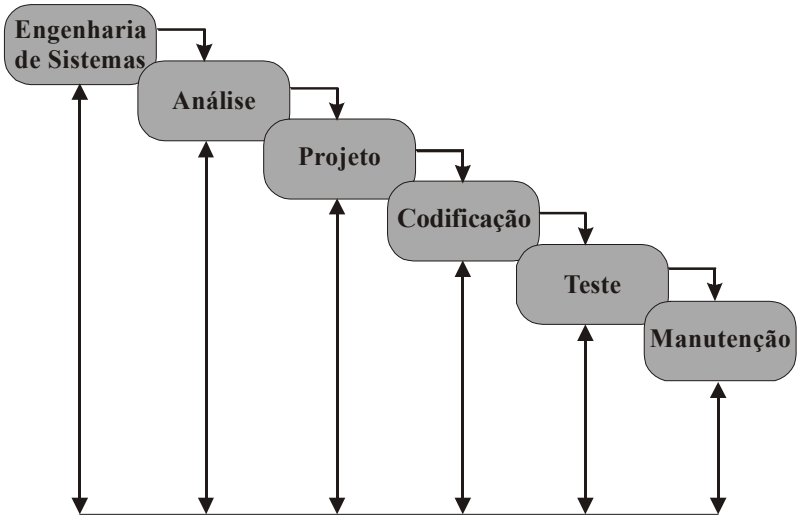


FIGURA 1 – Ciclo de Vida clássico ou modelo em cascata.

Análise e engenharia de sistemas – Nesta fase é feita uma análise do sistema em que o software será inserido, estabelecendo todos os seus elementos e requisitos. Além disso, ocorre a atribuição dos principais requisitos, necessária na interação do software com o sistema analisado. "*Essa visão do sistema é essencial quando o software deve fazer interface com outros elementos, tais como hardware, pessoas e banco de dados*". (PRESSMAN, 1995).

Análise de requisitos de software – Ocorre uma intensificação no processo de coleta dos requisitos com enfoque específico no software. O objetivo é a descoberta, refinamento, análise e especificação de quais desses requisitos serão necessários para definir a função, o desempenho e a interface exigidos para a elaboração do produto.

Projeto – Considerado como o ponto de partida na fase de desenvolvimento, pode ser definido como:

> *"...o processo de se aplicar várias técnicas e princípios ao propósito de se definir um dispositivo, um processo ou um sistema com detalhes suficientes para permitir sua realização física*". (TAYLOR, 1995).

O projeto é composto por diversas etapas que têm como objetivo básico quatro atributos distintos do programa: estrutura de dados, arquitetura de software, detalhes procedimentais e caracterização de interface. Nesta etapa, as exigências do software podem ser apresentadas para uma avaliação prévia antes do início de sua codificação.

Codificação – Nesta etapa o projeto é convertido em programa através de linguagens de programação, possibilitando que o mesmo seja compreendido e executado em computadores. Dependendo da qualidade do projeto e de seu nível de detalhamento, a codificação pode se tornar um processo automático.

Testes – Têm como objetivo avaliar e detectar possíveis erros durante o desempenho do software, a fim de garantir que o projeto idealizado seja realmente funcional, verificando se as saídas obtidas são compatíveis com os resultados esperados.

Manutenção – Praticamente todo software passa por um processo de manutenção e os motivos podem ser os mais diversos, correções após a fase de teste, alterações exigidas pelo usuário em virtude de mudanças ocorridas durante a execução do projeto, futuras atualizações, entre outras. Deve-se observar que para a realização da manutenção de um software, deverão ser reaplicadas cada uma das etapas precedentes do ciclo de vida.

2.1.3. Modelo de Prototipação e Teste

Esta abordagem é geralmente aplicada em projetos cujo nível de detalhamento é superficial, tendo sido levantados apenas os objetivos mais gerais do software. Problemas como eficiência dos algoritmos, adaptabilidade da interface e até mesmo requisitos de entrada, processamento e saída serão depurados através da prototipação (Fig. 2).

Segundo PRESSMAN (1995), este modelo apresenta três variações:

> "*(1) um protótipo em papel ou modelo baseado em PC que retrata a interação homem-máquina de uma forma que capacita o usuário a entender quanta interação ocorrerá; (2) um protótipo de trabalho que implementa algum subconjunto de função exigida do software desejado; ou (3) um programa existente que executa parte ou toda a função desejada, tendo outras características a serem melhoradas em um novo esforço de desenvolvimento".*

Neste modelo o projetista e o cliente definem, em conjunto, todos os requisitos básicos para o software, levando em conta os objetivos globais e as exigências conhecidas por ambos. Com base nesse levantamento, é desenvolvido um

protótipo que tem como característica apresentar ao usuário os formatos de entrada e saída. Esses aspectos serão avaliados e refinados, atendendo as suas necessidades e, ao mesmo tempo, capacitando o projetista para os ajustes necessários.

É importante destacar que o protótipo tem a função de definir os requisitos necessários para o projeto e que, após os diversos refinamentos, provavelmente este será descartado dando lugar ao software real final desenvolvido com a qualidade e funções desejadas.

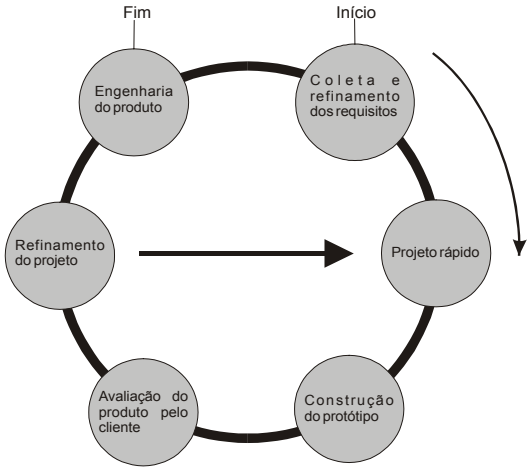


FIGURA 2 – Modelo de Prototipação e Teste.

De acordo com PRESSMAN (1995), o modelo em espiral é a concatenação dos modelos clássico e de prototipação, mais a inserção de um novo elemento ausente nos métodos anteriores – a análise de risco. O autor ainda define quatro atividades importantes neste processo:

> **"*Planejamento*** *– determinação de objetivos, alterações e restrições;* ***análise de riscos*** *– análise de alternativas e identificação/solução de riscos;* ***engenharia*** *– desenvolvimento do produto no "nível seguinte";* ***avaliação pelo cliente*** *– avaliação dos resultados da engenharia*".

Neste modelo são desenvolvidos vários níveis de iteração, sendo que cada nível representa uma volta na espiral. Esse processo permite observar novas características de acordo com as necessidades detectadas em cada ciclo da espiral, permitindo acrescentar melhorias com relativa redução no tempo de elaboração do

software. Durante o primeiro giro da espiral são identificados os objetivos, definidas as alternativas e restrições, além da identificação e análise dos riscos. Em cada novo ciclo ocorre o refinamento do produto com a manutenção e a identificação de novos problemas. É importante destacar que o modelo em espiral necessita de gestão bastante sofisticada para alcançar um nível confiável de eficiência (Fig. 3).

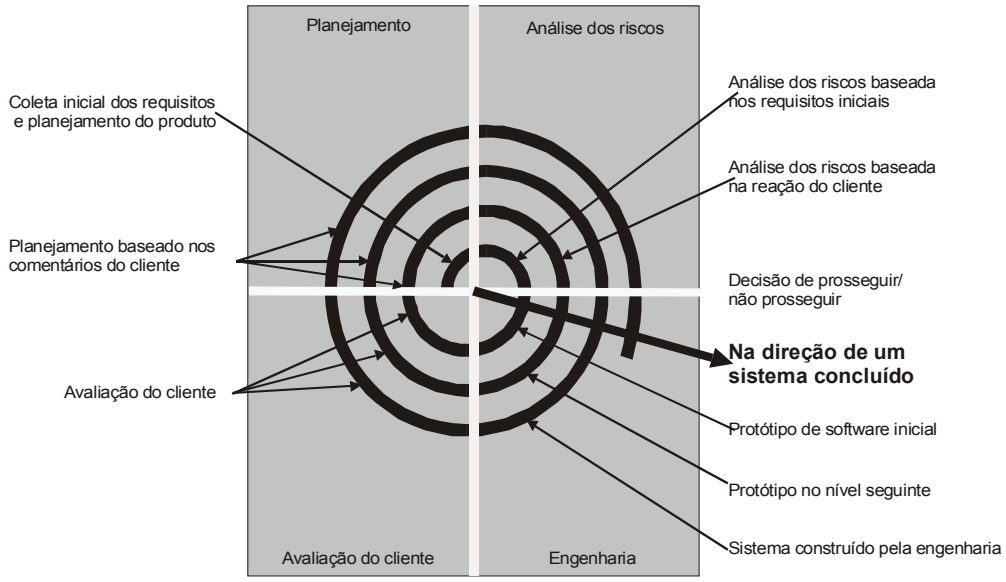


FIGURA 3 – Modelo em Espiral.

2.1.4. Modelo de Prototipagem Evolutiva

Este paradigma é uma combinação entre os modelos clássico, prototipação e espiral, na qual, segundo PAULA FILHO (2001), "*...a espiral é usada não para desenvolver o produto completo, mas para construir uma série de versões provisórias que são chamadas de protótipos*". Sua característica fundamental é permitir que os requisitos sejam definidos progressivamente, apresentando alta flexibilidade e visibilidade para o cliente. Como no modelo em espiral, também necessita de uma gestão de qualidade para não haver perda na estruturação do produto no desenvolvimento dos diversos protótipos (Fig. 4).

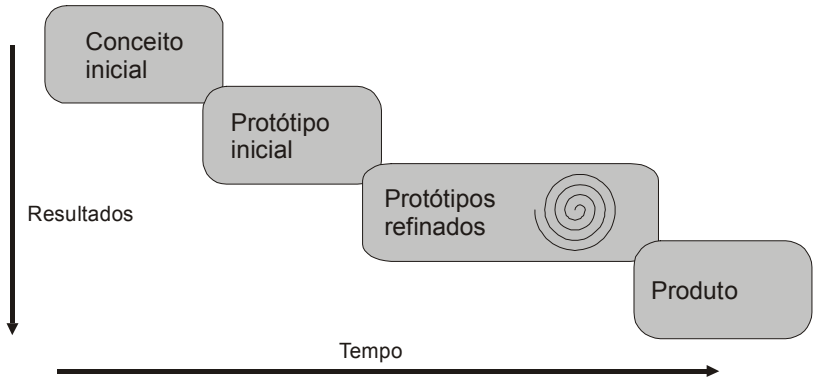


FIGURA 4 – Modelo de Prototipagem Evolutiva.

2.1.5. Técnicas de Quarta Geração

As técnicas de quarta geração, também conhecidas como 4GT, possuem um amplo conjunto de ferramentas de software que apresentam um recurso em comum: a possibilidade do desenvolvedor em especificar uma determinada característica do software em um nível elevado gerando automaticamente, em seguida, o código-fonte completo do nível analisado.

A rapidez de sua modelagem cresce proporcionalmente ao nível de especificação do software, ou seja, quanto mais alto o nível de especificação, mais rápido será construído o programa.

Segundo PRESSMAN (1995),

> "*O paradigma 4GT da engenharia de software concentra-se na capacidade de se especificar software a uma máquina em um nível que esteja próximo à linguagem natural ou de se usar uma notação que comunique uma função significativa*".

O paradigma de quarta geração também tem seu início a partir da coleta de dados, como mostra a Fig. 5. Teoricamente, os requisitos descritos pelos clientes poderiam ser imediatamente utilizados para a geração do protótipo; contudo, esses dados coletados precisam ser analisados e depurados em função de prováveis inconsistências nas informações, como acontecem nos demais paradigmas.

Em alguns casos de desenvolvimento de pequenas aplicações, é possível utilizar os dados coletados numa fase inicial, diretamente utilizados para a

implementação do protótipo operacional, utilizando uma “linguagem de quarta geração” (4GL). Contudo, o uso da linguagem de quarta geração sem planejamento, principalmente em grandes projetos, pode acarretar os mesmos problemas de má qualidade, dificuldades na sua manutenção e desaprovação por parte dos usuários.

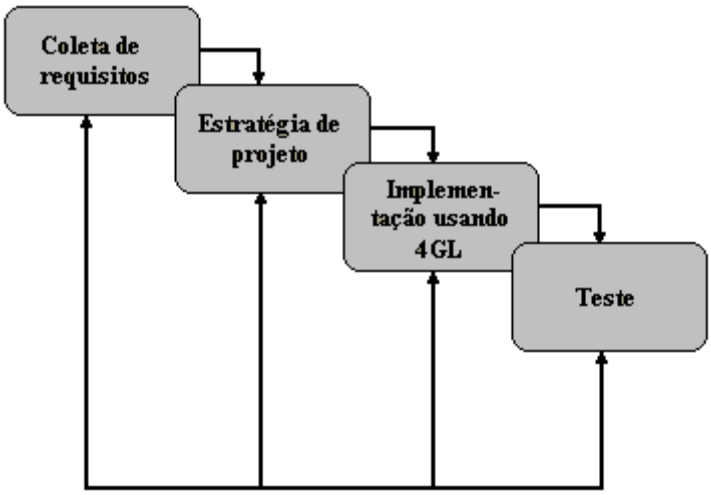


FIGURA 5 – Quarta geração – 4GT.

A vantagem de se utilizar uma linguagem de quarta geração está na possibilidade de gerar, de forma automática, todos os códigos necessários capazes de representar os resultados desejados para um sistema. As informações que darão subsídios para a geração dos códigos devem estar adequadamente disponíveis para a linguagem, tendo que passar por todos os níveis de transição encontrados nos demais modelos de paradigmas.

A grande maioria das aplicações baseadas no paradigma de quarta geração é voltada para sistemas comerciais que apresentam grandes bancos de dados; contudo, algumas exceções, como no caso das ferramentas CASE, suportam aplicações voltadas para engenharia e sistemas em tempo real.

Alguns casos analisados em empresas que utilizam o paradigma de quarta geração, indicam que há uma redução, tanto no tempo de produção de um software, como na quantidade de planejamento e análise para pequenas aplicações. Em contrapartida, grandes aplicações requerem igual ou maior análise, planejamento e teste para se ter uma redução considerável de tempo alcançado durante o processo de codificação automática.

2.2. Utilização Simultânea de Paradigmas

Em muitos casos a utilização de mais de um paradigma no mesmo projeto combinando suas potencialidades individuais podem trazer uma maior eficiência na elaboração de projetos. De acordo com PRESSMAN (1995), "*o paradigma do modelo espiral realiza isso diretamente, combinando a prototipação e elementos do ciclo de vida clássico numa abordagem evolucionária*".

Para os casos de utilização simultânea de paradigmas, o início do processo é semelhante aos demais como mostra a Fig. 6, devendo começar com as definições dos objetivos, alternativas e restrições. O próximo passo será definido pelas características do projeto a ser desenvolvido:

- Se a especificação do projeto for totalmente concluída no início, pode-se seguir as etapas do ciclo de vida clássico.
- Para requisitos indefinidos, pode-se utilizar como solução a prototipação e, posteriormente, retornar para as etapas do ciclo de vida clássico.
- Técnicas de quarta geração podem implementar um protótipo durante a codificação do ciclo de vida clássico.

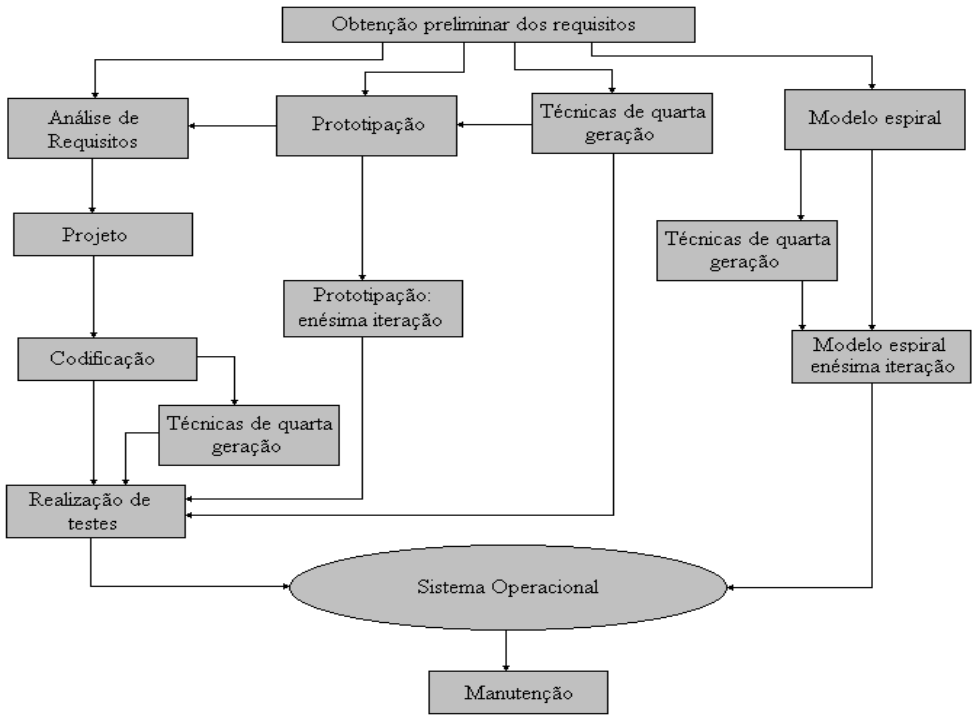


FIGURA 6 – Possibilidades de combinações de paradigmas.

Não se pode mais considerar o software como um simples conjunto de linhas de comando, ordenadas corretamente com o objetivo de solucionar um problema específico. O crescimento dinâmico da indústria de software e a evolução dos usuários, com um perfil cada vez mais exigente, fez surgir uma série de necessidades aliadas a novos e complexos problemas que fizeram do software um obstáculo na evolução de sistemas computacionais.

A engenharia de software apresenta os paradigmas e suas variações como um conjunto de técnicas que envolvem métodos, ferramentas e procedimentos necessários à análise e concepção da maioria dos sistemas existentes e futuros.

CAPÍTULO III

3. USABILIDADE

A busca da qualidade envolve diversos aspectos na produção de software, desde qualidades básicas como funcionalidade, confiabilidade e segurança de uso, até as chamadas qualidades extras ou implícitas como flexibilidade, adaptabilidade e facilidade de entendimento. Dentro desse conjunto de critérios está a interface que é responsável pela interação do usuário com o computador a qual deve primar pela qualidade da comunicação entre ambos.

Diversas são as denominações que tentam definir ou expressar a interação entre o homem e o computador. Siglas como ICH (Interação Computador-Homem), IHC (Interação Homem-computador), DCU (Design Centrado no Usuário), HF (Human-Factors ou Fatores Humanos), EF (Ergonomic-Factors ou fatores ergonômicos) entre outras, são comumente encontradas em publicações do gênero nas últimas décadas. Cabe ressaltar que, do ponto de vista da usabilidade, os fatores humanos e ergonômicos são mais abrangentes do que a simples interação do homem com o computador, pois fatores como faixa etária, nível de conhecimento do usuário, tipo de aplicação e entre outros, podem influenciar na qualidade da interface.

PRESSMAN (1995) descreve a usabilidade da seguinte maneira:

A conhecida expressão user friendliness (amigável ao usuário), tornou-se onipresente em discussões sobre produtos de software. Se um programa não for user friendly, freqüentemente estará destinado ao fracasso, mesmo que as funções executadas por ele sejam valiosas. A usabilidade é uma tentativa de se qualificar a user friendliness e pode ser medida segundo quatro características:

- A habilidade física e/ou intelectual exigida para se aprender o sistema;
- O tempo exigido para se tornar moderadamente eficiente no uso do sistema;
- O aumento líquido da produtividade (sobre a abordagem que o sistema substitui) medido quando o sistema é usado por alguém que seja moderadamente eficiente;

Uma avaliação subjetiva (em alguns casos, obtida por meio de questionário) das atitudes dos usuários em relação ao sistema.

É necessário situar em que parte do processo de desenvolvimento de um software encontra-se a usabilidade. De acordo com NIELSEN (1993) "*A aceitabilidade global de um sistema está dividida entre aceitabilidade social e aceitabilidade prática*", onde a primeira se caracteriza pela aceitação, por parte dos usuários, da necessidade e da relevância do papel social proposto por um determinado sistema. Quanto a sua aceitação prática, ela se subdivide em critérios como custo, confiança, segurança, compatibilidade, flexibilidade, dentre os quais encontra-se a utilidade.

A utilidade pode ser medida como sendo a propriedade de poder alcançar o resultado desejado, ou seja, que o sistema tenha a capacidade de solucionar o problema para o qual foi elaborado (NIELSEN, 1993). O critério utilidade se divide em dois aspectos importantes do software: se ele é útil, apresentando uma real necessidade capaz de justificar seu desenvolvimento, e sua usabilidade, que busca aplicar características que facilitem sua interação com o usuário.

A usabilidade tem como objetivo elaborar interfaces capazes de permitir uma interação fácil, agradável, com eficácia e eficiência.

Segundo a Norma ISO 9241– Parte 11 (CYBIS, 1997), referente à especificação da usabilidade dos sistemas, define usabilidade como sendo:

> *"...aquelas características que permitem que o usuário alcance seus objetivos e satisfaça suas necessidades dentro de um contexto de utilização determinado. Desempenho e satisfação do usuário são especificados e medidos a partir do grau de realização de objetivos perseguidos na interação (eficácia), pelos recursos alocados para alcançar estes objetivos (eficiência) e pelo grau de aceitação do produto pelo usuário (satisfação)".*

Ela deve capacitar a criação de interfaces transparentes de maneira a não dificultar o processo, permitindo ao usuário pleno controle do ambiente sem se tornar um obstáculo durante a interação.

3.1. Modelos de qualidade de Usabilidade

3.1.1. Heurísticas de Usabilidade de Nielsen

Estas heurísticas se baseiam na medição da facilidade de aprendizado, tendo como foco os efeitos da interação entre a interface e o usuário. Apresentam dez indicadores básicos:

1. Status do sistema visível – O sistema deve manter o usuário informado sobre seu estado atual, utilizando para isso mensagens claras e tempos razoáveis.
2. Correspondência entre o sistema e o mundo real – O sistema deve manter uma comunicação com o usuário de maneira a reproduzir termos e conceitos comuns a este, em relação ao seu mundo real, evitando a utilização de linguagem específica de sistemas.
3. Controle e liberdade do usuário – O sistema deve permitir que o usuário tenha um nível de controle elevado de seus comandos e tarefas, além de proporcionar um alto grau de liberdade em relação à forma de execução estes.
4. Padrão e consistência – O sistema deve apresentar telas padronizadas e consistentes para que dados semelhantes possam ser reconhecidos em telas diferentes.
5. Prevenção de erro – A quantidade de erros apresentados pelo sistema e a possibilidade do usuário executar uma operação errada devem ser as mais reduzidas possíveis.
6. Facilitar o reconhecimento - O sistema deve ser de fácil reconhecimento em sua utilização, permitindo sua a aprendizagem através de ações, objetos e opções visivelmente dispostos, tanto para usuários sem experiência como para ocasionais, com intervalos de tempo longos de utilização.
7. Flexibilidade e eficiência no uso – O sistema deve apresentar opções configuráveis que permitam sua utilização tanto por usuários novatos como experientes, além de ser eficiente em seu desempenho apresentando um alto nível de produtividade.
8. Tela minimalista e com desenho estético – Os diálogos devem apresentar somente informações que contenham algum grau de relevância, pois, uma unidade a mais de informação em um diálogo, concorre com as demais informações, causando uma redução da visibilidade destas.

9. O usuário deve poder identificar, diagnosticar e corrigir erros – As mensagens de erros devem apresentar clareza sem utilização de códigos. Deve conter a definição precisa do problema em questão juntamente com a(s) solução(ões) possível(eis).
10. Ajuda e Documentação – O usuário deve dispor, quando necessário, de documentação complementar ou de ajuda para a execução de alguma tarefa. Estas devem ter seus focos direcionados especificamente na tarefa que o usuário deseja realizar, apresentando, de forma clara, os passos necessários para realiza-la com o cuidado de não ser muito extensa.

3.1.2. O Modelo Ergonômico de SCAPIN & BASTIEN

Tem a função de conduzir o projeto de usabilidade objetivando analisar as causas que interferem no projeto ergonômico de interface, propõem o conjunto de oito critérios:

1. Condução – O software ergonomicamente correto deve conduzir o usuário na interação com o computador, utilizando para sua orientação mensagens, alarmes, rótulos entre outros, permitindo que usuário possa:

- Saber, a qualquer hora, onde está durante uma sequência de interações ou durante a execução de uma tarefa;
- Conhecer as possíveis ações e seus efeitos;
- Obter informações suplementares ou complementares.

Ao apresentar boa condução, o software permite uma aprendizagem rápida através de uma fácil utilização, melhorando seu desempenho e diminuindo o número de erros. Este critério possui subdivisões: presteza, feedback imediato, legibilidade e agrupamento/distinção de itens.

1.1. Presteza - Este critério engloba os meios utilizados para levar o usuário a realizar determinadas ações como, por exemplo, a entrada de dados, os mecanismos ou meios que permitam ao usuário conhecer as alternativas, em termos de ações, do estado ou contexto nos quais ele se encontra bem como a apresentação das ferramentas de ajuda e seu modo de acesso.

1.2. Feedback Imediato - Feedback Imediato diz respeito às respostas do sistema às ações do usuário. Estas ações podem ir do simples pressionar de uma tecla até a uma seleção dentro de uma lista de comandos. Em qualquer caso, as respostas do computador devem ser fornecidas, de forma rápida, no momento apropriado

e de forma consistente com cada tipo de transação. Devem ser fornecidas respostas rápidas com informação sobre a transação solicitada e o seu resultado.

1.3. Legibilidade - diz respeito às características lexicais das informações apresentadas na tela que possam dificultar ou facilitar a leitura desta informação (brilho do caráter, contraste letra/fundo, tamanho da fonte, espaçamento entre palavras, espaçamento entre linhas, espaçamento entre parágrafos, comprimento da linha, etc.). Por definição, o critério *Legibilidade* não abrange mensagens de erro ou de *feedbac*k..

1.4. Agrupamento/Distinção de Itens - O critério *Agrupamento/Distinção de Itens* diz respeito à organização visual dos itens de informação de alguma maneira relacionados entre si. Este critério leva em conta a topologia (localização) e algumas características gráficas (formato) para indicar as relações entre os vários itens mostrados, para indicar se eles pertencem ou não a uma dada classe, ou ainda para indicar diferenças entre classes. Este critério também diz respeito à organização dos itens de uma classe. O critério *agrupamento/distinção de itens* está subdividido em dois critérios: *agrupamento/distinção por localização* e *agrupamento/distinção por formato*.

1.4.1. Por localização - O critério de *Agrupamento/Distinção por Localização* diz respeito ao posicionamento relativo dos itens, estabelecido para indicar se eles pertencem ou não a uma dada classe ou, ainda, para indicar diferenças entre classes. Este critério também diz respeito ao posicionamento relativo dos itens dentro de uma classe.

1.4.2. Por formato - O critério de *Agrupamento/Distinção por Formato* diz respeito mais especificamente às características gráficas (formato, cor, etc.) que indicam se os itens pertencem ou não a uma dada classe, ou que indicam distinções entre classes diferentes, ou ainda distinções entre itens de uma dada classe.

2. Carga de trabalho - O critério Carga de Trabalho diz respeito a todos elementos da interface que têm um papel importante na redução da carga cognitiva e perceptiva do usuário e no aumento da eficiência do diálogo. O critério Carga de Trabalho está

subdividido em três sub-critérios: Brevidade (o qual inclui Concisão e Ações Mínimas) e Densidade Informacional.

2.1. Brevidade - O critério *Brevidade* diz respeito à carga de trabalho perceptiva e cognitiva, tanto para entradas e saídas individuais, quanto para conjuntos de entradas (i.e., conjuntos e ações necessárias para se alcançar uma meta). *Brevidade* corresponde ao objetivo de limitar a carga de trabalho de leitura e entradas e o número de passos. O critério de *Brevidade* está subdividido em dois sub-critérios: *Concisão* e *Ações Mínima*s.

2.1.1. Concisão - O critério *Concisão* diz respeito à carga perceptiva e cognitiva de saídas e entradas individuais. Por definição, *Concisão* não diz respeito às mensagens de erro e de *feedbac*k.

2.1.2. Ações Mínimas - O critério *Ações Mínimas* diz respeito à carga de trabalho em relação ao número de ações necessárias à realização de uma tarefa. O que temos aqui é uma questão de limitar tanto quanto possível o número de passos pelos quais o usuário deve passar.

2.2. Densidade Informacional - O critério *Densidade Informacional* diz respeito à carga de trabalho do usuário de um ponto de vista perceptivo e cognitivo, em relação ao conjunto total de itens de informação apresentados ao usuário e não a cada elemento ou item individual.

3. Controle explícito - O critério Controle Explícito diz respeito tanto ao processamento explícito pelo sistema das ações do usuário, quanto do controle que os mesmos têm sobre o processamento de suas ações pelo sistema. O critério *Controle Explícito* subdivide-se em dois critérios: *Ações Explícitas do Usuário* e *Controle do Usuário*.

3.1. Ações Explícitas do Usuário - O critério *Ações Explícitas do Usuário* refere-se às relações entre o processamento pelo computador e as ações do usuário. Esta relação deve ser explícita, i.e., o computador deve processar somente aquelas ações solicitadas e somente quando solicitado para o fazer.

3.2. Controle do Usuário - O critério *Controle do Usuário* refere-se ao fato de que os usuários deveriam ter sempre no controle sobre o processamento do sistema (i.e., interromper, cancelar, suspender e continuar, avançar, retroceder, ou parar

a apresentação). Cada ação possível deve ser antecipada e devem ser oferecidas opções apropriadas.

4. Adaptabilidade - A *adaptabilidade* de um sistema diz respeito à sua capacidade de reagir conforme o contexto e conforme as necessidades e preferências do usuário. Dois sub-critérios compõem a adaptabilidade: a *flexibilidade* e a *consideração da experiência do usuário.*

4.1. Flexibilidade - A *flexibilidade* refere-se aos meios colocados à disposição do usuário que lhe permite personalizar a interface a fim de levar em conta as exigências da tarefa, de suas estratégias ou hábitos de trabalho. Ela corresponde também ao número das diferentes maneiras à disposição do usuário para alcançar um certo objetivo. Em outros termos, trata-se da capacidade da interface em se adaptar às variadas ações do usuário.

4.2. Consideração da Experiência do Usuário - A *consideração da experiência do usuário* diz respeito aos meios implementados que permitem que o sistema respeite os níveis de experiência individuais.

5. Gestão de erros - A *gestão de erros* diz respeito aos mecanismos que permitem evitar ou reduzir a ocorrência de erros e, quando eles ocorrem, que favoreçam sua correção. Os erros são aqui considerados como entrada de dados incorretos, entradas com formatos inadequados, entradas de comandos com sintaxes incorretas, etc. Considera-se erro também, as respostas inadequadas ao sistema mediante os recursos de verificação de aprendizagem dos conteúdos no *software* educacional. Três sub-critérios participam da manutenção dos erros: a *proteção contra os erro*s, a *qualidade das mensagens de erro* e a *correção dos erro*s.

5.1. Proteção Contra Erros - A *proteção contra os erros* diz respeito aos mecanismos empregados para detectar e prevenir os erros de entradas de dados, comandos, possíveis ações de consequências desastrosas e/ou não recuperáveis.

5.2. Qualidade da Mensagem de Erro - A qualidade das mensagens refere-se à pertinência, legibilidade e exatidão da informação dada ao usuário sobre a natureza do erro cometido (sintaxe, formato, etc.) e sobre as ações a executar para corrigi-lo.

5.3. Correção dos Erros - O critério *correção dos erros* diz respeito aos meios colocados à disposição do usuário com o objetivo de permitir a correção dos seus erros.

6. Homogeneidade/Coerência - O critério *homogeneidade* refere-se ao modo como as escolhas na concepção da interface (códigos, denominações, formatos, procedimentos, etc.) são conservadas idênticas em contextos idênticos e diferentes em contextos diferentes, de uma tela para outra.
7. Significado dos Códigos e Denominações - O critério *significado dos códigos e denominações* diz respeito à adequação entre o objeto, informação apresentada ou pedida e sua referência. Códigos e denominações significativas possuem uma forte relação semântica com seu referente. Termos pouco expressivos para o usuário podem ocasionar problemas de condução que o podem levar à seleção de uma opção errada.
8. Compatibilidade - O critério *compatibilidade* refere-se às relações que possam existir entre as características do usuário (sexo, idade, formação e competência) na realização de suas tarefas (conforme suas convenções, habilidades, preferências, expectativas e estratégias) e a organização das apresentações, das entradas e do diálogo de uma dada aplicação. Ela diz respeito também ao grau de similaridade entre diferentes ambientes e aplicações.

3.1.3. Norma ISO 9241 – Parte 10 (*apud* CYBIS, 1995)

Trata dos Princípios de Diálogo e apresenta um conjunto de sete critérios:

1. **Adaptabilidade à tarefa** - Um diálogo é adaptável à tarefa quando dá suporte ao usuário na realização efetiva e eficiente da tarefa.

Aplicações:

- O diálogo deve ser apresentado ao usuário somente com informações relacionadas a realização da tarefa .
- Informação de ajuda deve ser dependente da tarefa.
- Quaisquer ações que possam ser executadas automaticamente de forma apropriada devem ser levadas a efeito pelo software sem envolvimento do usuário.
- Quando do projeto do diálogo deve-se considerar as habilidades e capacidades do usuário face à complexidade da tarefa.

- O formato de entrada e saída deve ser apropriado à tarefa e às solicitações dos usuários.
- O diálogo deve dar suporte ao usuário quando da realização de tarefas recorrentes.
- Se numa tarefa existirem possibilidades de entradas default (ex. Valores default padrão), não deve ser necessário que o usuário tenha de entrar com tais valores. Também deve ser possível substituir valores default por outros valores ou por outros valores default.
- Durante a realização de uma tarefa na qual dados são modificados, os dados originais devem permanecer acessíveis, se isto for necessário.
- O diálogo deve evitar passos desnecessários à tarefa.

2. **Autodescrição** - Um diálogo é auto descritivo quando cada passo é imediatamente compreendido através do feedback do sistema, ou quando sob demanda do usuário.

Aplicações:

- Se for apropriado, depois de qualquer ação do usuário, o diálogo deve oferecer feedback. Se consequências graves possam resultar das ações dos usuários, o sistema deve oferecer explicações e solicitar confirmação antes de efetuar a ação.
- O feedback ou as explanações devem ser apresentados numa terminologia adequadamente derivada do ambiente da tarefa e não da tecnologia do sistema.
- Como um possível complemento ao treinamento de usuários, o feedback ou as explanações devem prestar assistência ao usuário no sentido deste obter um entendimento geral do sistema de diálogo.
- O feedback ou as explanações devem ser baseadas no nível de conhecimento esperado para o usuário típico.
- O sistema deve fornecer feedback ou explanações variando em tipo e extensão conforme as necessidades e características do usuário.
- Para aumentar o seu valor para o usuário, o feedback ou a explicação devem estar relacionados estritamente com a situação na qual eles são necessários. O feedback ou da explicação de qualidade podem minimizar a consulta a manuais do usuário, e outro tipo de informação externa, evitando deste modo, frequente mudanças de mídias.

- Se existem valores default para uma determinada tarefa eles devem estar disponíveis ao usuário.
- O usuário deve ser informado sobre mudanças no status do sistema que são relevantes para a tarefa.
- Quando uma entrada é solicitada o sistema deve fornecer informações sobre a entrada esperada.
- As mensagens devem ser formuladas e apresentadas em um estilo compreensível, objetivo e positivo e seguindo uma estrutura consistente. As mensagens não devem conter julgamentos, tais como 'Esta entrada não tem sentido'.

3. **Controlabilidade** - O diálogo é controlável quando o usuário é capaz de iniciar e controlar a direção e o ritmo da interação até que seu objetivo seja atingido.

Aplicações:

- A velocidade da interação não deve ser ditada pelo sistema. Ela deve estar sempre sob o controle do usuário, de acordo com suas necessidades e características.
- Ao usuário deve ser dado o controle sobre como continuar o diálogo.
- Quando da retomada do diálogo após uma interrupção, o usuário deve ter a habilidade de determinar o ponto de reinicio, se a tarefa o permitir.
- Se existem interações reversíveis e a tarefa permite, deve ser possível desfazer no mínimo o último passo do diálogo.
- Diferentes características e necessidades dos usuários requerem diferentes níveis e métodos de interação.
- O modo como as entradas e saídas de dados são representadas (formato e tipo) devem estar sob controle do usuário.
- Se o controle da quantidade de dados apresentados é útil para uma determinada tarefa, o usuário deve estar habilitado a exercê-lo.
- Quando dispositivos alternativos de entradas e saídas coexistem o usuário deve poder escolher qual utilizar.

4. **Conformidade com as expectativas do usuário** - O diálogo adapta-se às expectativas do usuário quando ele é consistente e corresponde a suas

características, tais como conhecimento da tarefa, educação, experiência e convenções.

Aplicações:

- O comportamento e a aparência do diálogo no sistema deve ser consistente.
- Ações de mudança de estado devem ser implementadas consistentemente.
- O aplicativo deve usar um vocabulário que fosse familiar ao usuário na execução de uma tarefa.
- Os diálogos realizados para tarefas similares devem ser similares de maneira que o usuário possa desenvolver procedimentos padronizados.
- O feedback imediato de entradas do usuário deve ser fornecido sempre que apropriado com as expectativas dos usuários. Ele deve se basear no nível de conhecimento do usuário.
- O cursor deve estar colocado onde a entrada está sendo esperada.
- Se o tempo de resposta provavelmente desvie do tempo esperado, o usuário deve ser informado disto.

5. **Tolerância a erros** - Um diálogo é tolerante à erros se a despeito de erros evidentes de entrada, o resultado esperado pode ser alcançado com mínimas ou nenhuma ação corretivas por parte do usuário.

Aplicações:

- O aplicativo deve apoiar o usuário na detecção, evitando os erros de entrada. O sistema deve prevenir qualquer entrada do usuário que instabilize o sistema ou que ocasionem falhas no diálogo.
- Os erros devem ser explicados de maneira a ajudar os usuários a corrigi-los.
- Dependendo da tarefa pode ser desejável aplicar esforço especial na apresentação de técnicas para melhorar o reconhecimento de situações de erro e a sua recuperação.
- Nos casos onde o sistema seja capaz de corrigir erros automaticamente ele deve avisar o usuário da execução da correção e fornecer a oportunidade de anular as correções.
- Necessidades e características do usuário podem requerer que situações de erro sejam postergadas deixando ao usuário a decisão de quando lidar com elas.

- É interessante fornecer sob demanda, explicações adicionais durante a correção do erro.
- A validação e verificação de dados devem se realizar antes de processar as entradas. Controles adicionais devem ser fornecidos para comandos com sérias consequências.
- Onde a tarefa permitir, a correção de erros deve ser possível sem a mudança de estado do sistema.

6. **Adequação à individualização** - O sistema é capaz de individualização quando a interface pode ser modificada para se adaptar as necessidades da tarefa, as preferências individuais e as habilidades dos usuários.

Aplicações:

- Mecanismos devem ser fornecidos para permitir ao sistema se adaptar a linguagem e cultura dos usuários, assim como seu conhecimento individual, experiência no domínio da tarefa, habilidades perceptivas, senso-motoras e cognitivas.
- O sistema deve permitir que o usuário escolha entre modos alternativos de apresentação de acordo com suas preferências individuais e de acordo a complexidade da informação a ser processada.
- A quantidade de explicação (detalhes nas mensagens de erros, informação de ajuda) deve ser modificável de acordo com o nível de conhecimento do usuário.
- O usuário deve ser capaz de incorporar o seu próprio vocabulário para estabelecer nomes de objetos ou ações, se isto segue o contexto da tarefa. Também deve ser possível ao usuário adicionar comandos individuais.
- O usuário deve ser capaz de estabelecer parâmetros operacionais de tempo para satisfazer suas necessidades individuais.
- Os usuários devem ser capazes de escolher entre diferentes técnicas de diálogo para diferentes tarefas.

7. **Adequação ao aprendizado** - O sistema é adequado ao aprendizado quando apóia e conduz o usuário no aprendizado do sistema.

Aplicações:

- Regras e conceitos subjacentes que são úteis para o aprendizado devem estar disponíveis para o usuário, permitindo que ele construa seus próprios grupos de estratégias e regras de memorização de atividades.
- Estratégias relevantes de aprendizado (compreensão orientada, aprendizado pela ação, aprendizado por exemplos) devem ser fornecidos.
- Facilidades de reaprendizado devem ser fornecidas.
- Um número de diferentes meios de ajuda para o usuário tornar-se familiar com os elementos do diálogo deve ser fornecido.

3.2. Categorias de usuários e tipos de interfaces

Outro fator de grande importância que deve ser considerado são as categorias de usuários e suas diferenças individuais. A Fig. 7 mostra, segundo NIELSEN (1993), as três principais dimensões que diferem o usuário em sua perícia: com o sistema, com computadores em geral e com o domínio da tarefa.

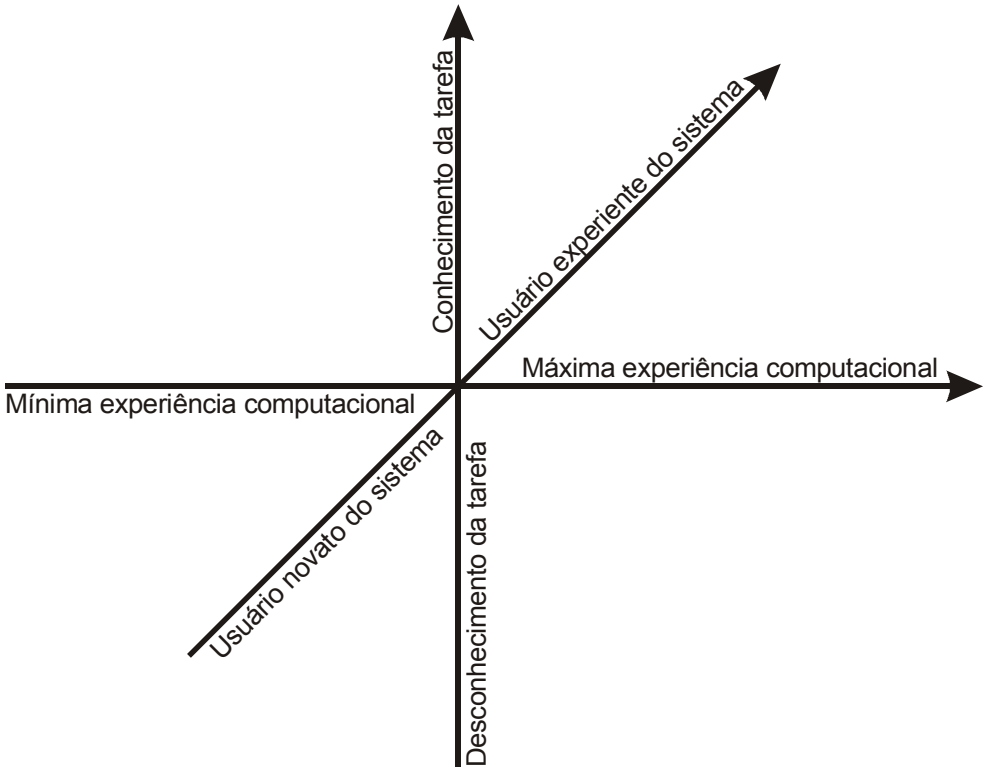


FIGURA 7: As três dimensões que representam a experiência do usuário.

Quando se discute perícia do usuário, deve-se levar em consideração a experiência deste em relação a uma interface específica e, normalmente, são classificados em novatos ou experientes ou se localizam em algum lugar entre ambos.

A curva de aprendizagem que ocorre tanto com o usuário novato como com o experiente ou perito pode ser representada pelo gráfico da Fig. 8.

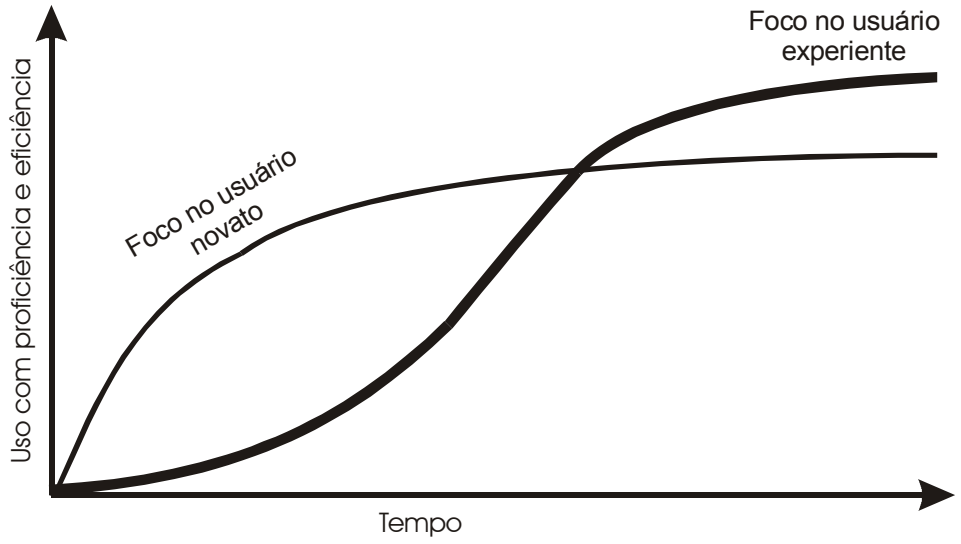


FIGURA 8: Curva de aprendizagem para usuário novato e experiente

Em um sistema hipotético para um usuário novato, o mesmo deve apresentar fácil aprendizagem e pouca eficiência na utilização. Para o caso de usuários experientes o sistema deve apresentar alta eficiência, contudo, maior dificuldade de aprendizagem em sua utilização.

Há vários elementos de interface que podem colaborar com o aumento da perícia do usuário sem interferir nas ações pretendidas de usuários novatos:

- Teclas de atalho – opções que são indicadas nos menus, na forma de combinação de duas ou mais teclas que apresentam a mesma função desejada.
- Barra de ferramentas – conjunto de ícones que executam a mesma tarefa encontrada em diversas opções nos menus.
- Macros – utilização de um conjunto de comandos com o objetivo de realizar uma tarefa repetitiva com um determinado nível de complexidade.
- Sistemas de ajuda Online – permite a qualquer nível de usuário aprofundar seus conhecimentos nas técnicas de interação disponíveis em um determinado

ambiente, sugerindo melhores alternativas e caminhos para alcançar um objetivo. Colabora com a transição do usuário novato para experiente.

Existem interfaces desenvolvidas com o objetivo de proporcionar a interação com usuários novatos em situações onde, quase sempre, serão utilizadas apenas uma vez. São os casos de quiosques que possuem informativos multimídias e estão dispostos em lugares públicos com dados gerais de localização, entre outros. Como exemplo pode-se citar os *shopping centers*, grandes magazines, museus, bosques, entre outros.

Outras necessitam de uma série de leituras de instruções passo a passo, basicamente apresentando o mesmo formato, que conduzem o usuário na execução de uma tarefa objetiva, tais como programas de instalação, rotinas de formatação de discos, copias de arquivos, etc.

Alguns sistemas apresentam interfaces com dois grupos distintos de menus. Os menus curtos, desenvolvidos para usuários novatos, que apresentam funções básicas, porém com recursos limitados e os menus longos criados para os usuários experientes, apresentando uma grande variedade de recursos. Este modelo permite a interface disponibilizar uma vasta gama de recursos aos usuários experientes sem interferir na interação do novato.

É importante destacar que, em sistemas que apresentam um certo grau de complexidade, nenhum usuário apresenta perícia completa em um nível de abrangência total, ou seja, até mesmo um usuário experiente deve apresentar características de usuário novato ao interagir com partes específicas do sistema aos quais não esteja familiarizado. Desta forma, mesmo usuários experientes podem necessitar de auxílio na compreensão de tarefas desconhecidas e interfaces não utilizadas comumente.

Outro fator que influencia no desenvolvimento de interfaces é o nível de domínio da tarefa e do sistema por parte do usuário. De acordo com o domínio do usuário, pode-se usar terminologias específicas e uma alta densidade de informação na tela. Como exemplo pode-se citar os usuários experientes ou especialistas.

Nos casos em que o conhecimento do sistema e/ou do domínio da tarefa é diminuto, se faz necessário o desenvolvimento de interfaces mais explicativas com telas menos densas e terminologia mais geral (NIELSEN, 1993). Um exemplo desta situação

ocorre com os quiosques multimídias instalados em lugares públicos, onde os usuários são ocasionais.

Deve-se levar em consideração, também, o fator de aceitação/rejeição da informática por parte dos usuários. Existem usuários que dedicam uma grande parcela de tempo no aprendizado total de seu sistema, buscando o domínio amplo do conhecimento em todos os seus níveis. Em posição contrária, alguns usuários são extremamente avessos ao computador e seus recursos, podendo gerar processos de rejeição em diversos níveis de intensidade, para estes usuários a interface deve ser projetada para apresentar um ambiente o mais amigável possível, objetivando minimizar ou reverter tais processos.

Observa-se, partindo do exposto acima, que a usabilidade aplicada à interface depende de diversos fatores de ordem subjetiva, tais como: nível de qualificação do usuário, tipo de aplicação, capacidade de aprendizagem, aceitação/rejeição, domínio da tarefa, complexidade do sistema, entre outros. Além disso, em alguns casos a interface é gerada a partir de suposições empíricas do projetista ou baseada na opinião do usuário, o qual pode não apresentar qualificação para tal. Essa prática, na maioria das vezes, contempla poucos ou nenhum dos critérios acima descritos, ocasionando uma baixa qualidade na interface.

Tanto a usabilidade como seus indicadores devem ser estudados de forma mais sistemática, permitindo que seja feito um projeto adequado com todas as suas etapas bem estabelecidas. Partindo dessa necessidade, criou-se a disciplina Engenharia de Usabilidade que tem como objetivo conduzir de forma sistematizada as atividades necessárias para a elaboração da interface durante todo o ciclo de vida de desenvolvimento do sistema (NIELSEN, 1993).

CAPÍTULO IV

4. ENGENHARIA DE USABILIDADE

Neste capítulo serão apresentados a engenharia de usabilidade e seu modelo de ciclo de vida com os principais estágios para a elaboração de um produto.

O principal objetivo da engenharia de usabilidade é melhorar a qualidade de interação do usuário na utilização do produto, diminuindo o esforço no desempenho geral de sua função. Para se obter resultados satisfatórios de usabilidade é necessário que se estabeleça quais são os critérios e procedimentos a serem seguidos, a partir da observação e analise da interação de um grupo explícito e/ou implícito de usuários.

O processo que determina quais etapas devem ser utilizadas e como elas devem ser aplicadas é conhecido como ciclo de vida de usabilidade, que busca identificar as necessidades de um determinado sistema para que o mesmo apresente as principais características de usabilidade: inteligibilidade, apreensibilidade, operacionalidade, atratividade e conformidade.

A engenharia de usabilidade é um conjunto de processos que são executados durante todo o ciclo de vida do produto e ocorre em diversos estágios, antes mesmo da interface do usuário ter sido projetada. Segundo GOULD e LEWIS *apud* NIELSEN (1993), "*...a necessidade da utilização de múltiplos estágios na engenharia de usabilidade foi reconhecida desde cedo, em função de sua complexidade na busca da qualidade do software".*

O desenvolvimento em etapas possibilita a detecção de erros mais rapidamente, o aumento do nível de eficiência e eficácia do produto final e do ganho de tempo, o que diminui o gasto final do produto. A usabilidade não pode ser vista de forma isolada em relação ao contexto geral de desenvolvimento de um produto, do contrário, essa atitude pode provocar alterações indesejáveis em vários aspectos, desde o ciclo de vida da engenharia de software, até um produto com falhas de interação ao usuário final.

Entretanto, essa situação ocorre com uma certa frequência em função do desconhecimento da própria usabilidade ou pelo alto grau de importância dispensado ao produto no que se refere a sua funcionalidade básica, em detrimento da qualidade do processo de interação com o usuário.

4.1. Ciclo de Vida da Engenharia de Usabilidade

Embora existam propostas mais recentes como a de MAYHEW (1999), esta pesquisa se baseia na proposta apresentada por NIELSEN (1993), por entender que a primeira apresenta um nível de detalhamento elevado para compor um capítulo de ambientação.

O ciclo de vida da engenharia de usabilidade pode ser desdobrado em estágios como apresentado no quadro abaixo, os quais serão descritos a seguir.

Estágios do Ciclo de Vida da Engenharia de Usabilidade			
01	Conhecimento do usuário	1.1	Características individuais do usuário
		1.2	Análise da tarefa
		1.3	Análise funcional
		1.4	Evolução do usuário
02	Análise competitiva		
03	Metas de usabilidade	3.1	Análise do impacto financeiro
04	Projeto paralelo		
05	Projeto participativo		
06	Coordenação da interface total		
07	Diretrizes		
08	Prototipagem		
09	Projeto interativo		

Quadro I – Estágios do Ciclo de vida.

4.1.1. Conhecimento do usuário.

O primeiro passo no processo de análise da usabilidade está em estudar as características dos possíveis usuários e dos produtos que estes utilizam. As características individuais de cada usuário e a variabilidade das tarefas são os dois fatores de maior impacto na usabilidade, necessitando de um cuidadoso estudo. Para a engenharia de usabilidade, a definição do conceito de usuário inclui todos aqueles cujo trabalho pode ser afetado pelo produto, em algum momento, no decorrer do processo de utilização.

Mesmo sabendo que o conhecimento do usuário e de suas características se constituem informações básicas para a maioria das diretrizes da usabilidade, as mesmas são, com freqüência, difíceis de serem ser acessadas pelos desenvolvedores. GRUDIN *apud* NIELSEN (1993) destaca alguns dos principais obstáculos a esse acesso:

- A necessidade da empresa de desenvolvimento em proteger seus desenvolvedores em relação a clientes com solicitações de tarefas técnicas, desviando-os de seu objetivo principal;
- A relutância de alguns setores da empresa analisada em permitir que desenvolvedores ou profissionais de usabilidade entrevistem seus clientes, temendo que os mesmos se sintam ofendidos ou insatisfeitos com o procedimento;
- O pouco tempo disponível de alguns funcionários da empresa analisada, por ocuparem funções estratégicas.

Não há nenhuma solução padrão disponível. O que se recomenda é um esforço explícito para se obter acesso direto aos usuários representativos, não se satisfazendo com dados e informações imprecisas.

Uma considerável parcela de tempo no desenvolvimento de um projeto é absorvida para se discutir como são os usuários ou o que eles desejam fazer. Ao invés de discutir essas questões sem a disponibilidade necessária de informações, é muito melhor e menos tempo se consome se os dados forem obtidos diretamente com os usuários.

4.1.1.1. Características individuais do usuário.

É necessário conhecer as características do usuário que utilizará o sistema. Esta caracterização se torna fácil e possível de ser identificada quando os usuários são indivíduos concretos, isto ocorre quando o produto que se vai desenvolver pertence a um departamento específico de uma companhia. Para outros produtos direcionados ao público em geral, cujos usuários são amplamente dispersos isto é possível somente através da entrevista de alguns clientes representativos; em compensação, seu alcance pode atingir um contingente bem maior de usuários.

Através do conhecimento do nível de experiência do trabalho, nível de escolaridade, idade, experiência prévia com computadores, é possível antecipar as dificuldades desses usuários em relação ao sistema proposto e determinar qual o limite apropriado de complexidade de sua interface. Outra característica importante é conhecer a capacidade de leitura e de comunicação de cada usuário, como no caso de crianças com pouca capacidade de leitura, onde se faz necessária a utilização de uma interface não textual.

O conhecimento do tempo disponível de cada usuário para a aprendizagem e se eles terão oportunidade de participar de cursos de treinamento, são fatores que podem determinar o nível de simplicidade da interface e o tempo máximo de treinamento.

A análise do ambiente de trabalho e do contexto social dos usuários são informações que podem ser obtidas através de questionários ou entrevistas, contudo não se deve confiar totalmente nas informações escritas, novos pontos de vista são alcançados a partir da observação e da conversa informal com os usuários reais em seu próprio ambiente de trabalho.

4.1.1.2. Análise da tarefa.

A análise da tarefa é essencial para o início do projeto de um sistema. Devem ser analisadas as metas globais dos usuários, como são executadas, quais informações são necessárias, como são trabalhadas as circunstâncias excepcionais ou emergenciais. Segundo GARBER & GRUNES *apud* NIELSEN (1993),

> *...em alguns casos, ao entrevistar ou observar clientes dos usuários ou outros que tenham interagido com estes podem ser obtidas visões*

> *de análise de tarefa adicionais. Além disso, devem ser identificados os pontos fracos de uma situação corrente.*

Determinar em que ponto o usuário falha em alcançar suas metas, o que causa os gastos de tempo excessivo, são informações que podem acrescentar melhorias no novo produto.

Basicamente, o resultado de uma análise de tarefa deve gerar uma relação contendo as metas desejadas no sistema com as informações e os passos necessários para obtê-las, além de suas interdependências, os resultados e relatórios que são necessário produzir, os critérios utilizados para determinar a qualidade e a aceitabilidade desses resultados e a comunicação necessária entre usuários a interação com o sistema. De acordo com GREIF *apud* NIELSEN (1993), uma análise de tarefa pode ser decomposta em um modelo hierárquico, começando com as metas e tarefas maiores da organização e quebrando cada uma delas dentro em subdivisões menores. Este tipo de análise hierárquica, além de permitir relacionar uma determinada atividade a metas importantes (por que fazer isto?), permite decompor tal atividade em outras tarefas menores (como você faz isto?). NIELSEN *et al. apud* NIELSEN(1993) sugere algumas questões alternativas, "*por que você não faz isto de tal forma?*", "*ocorrem erros sempre que se faz isto?*" ou "*como você faz para descobrir e corrigir estes erros?*".

Os usuários devem descrever exceções de seus fluxos de trabalho normal; contudo, como grande parte delas não serão recordadas e nem previstas as futuras, deve haver um número considerável de observações a serem feitas.

Por último, deve-se solicitar aos usuários a indicação de falhas, problemas, sugestões mínimas de melhorias ou fatores incômodos.

4.1.1.3. Análise funcional.

Segundo SCHMIDT *apud* NIELSEN (1993), no desenvolvimento de um novo sistema computacional não devem ser analisados apenas os procedimentos na realização de uma tarefa mas também sua razão funcional, sua real necessidade e quais procedimentos podem e, talvez, devam ser alterados. Como exemplo dado por EGAN *et al*. *apud* NIELSEN (1993), observações iniciais de usuários de extensos manuais impressos onde , frequentemente, se percebe a manipulação dos textos, pode-se elaborar

um projeto ingênuo de documentação *on-line* que apresente uma boa e rápida paginação e mecanismo de rolagem de texto. Uma análise funcional criteriosa pode mostrar que, mesmo com esses recursos, a tarefa de localizar um determinado trecho do texto demanda um longo tempo. Com base nessa análise poderia se propor um projeto de interface acrescentando um recurso de pesquisa e localização, selecionando os resultados obtidos, diminuindo consideravelmente o tempo de realização da tarefa.O exemplo citado anteriormente demonstra que além da análise da tarefa se faz necessária uma análise da funcionalidade do processo.

4.1.1.4. Evolução do usuário.

Todo usuário tende a evoluir no decorrer da interação com um sistema, utilizando-o ou adaptando-o para novas tarefas. De acordo com CARROL & ROSSON *apud* NIELSEN (1993), este fenômeno dialético é conhecido como a coevolução das tarefas e artefatos.

É impossível se planejar completamente as mudanças que podem ocorrer nos usuários após um determinado período de utilização do sistema, entretanto um projeto flexível é capaz de suportar melhorias para esse novo perfil de usuário. Uma mudança típica observada após os usuários tornarem-se experientes é o desejo de atalhos de interação (teclas aceleradoras).

Deve-se considerar, durante o projeto, a possibilidade da qualificação do usuário iniciante, permitindo que o sistema apresente configurações avançadas, alternativas e meios que possibilitem alterações para atualizações ou futuras versões.

4.1.2. Análise competitiva.

Quando vários produtos competitivos são disponibilizados para análise comparativa, suas semelhanças e diferenças na interface podem gerar importantes questões quanto à usabilidade do protótipo analisado. Essas observações serão usadas como diretrizes na elaboração do projeto, desenvolvendo um produto mais competitivo. Em alguns casos, a análise competitiva envolve o estudo de interfaces não computadorizadas tal como o projeto de um livro eletrônico, onde primeiramente são analisadas as pessoas que têm uma boa interação com enciclopédias impressas.

A análise competitiva não significa a apropriação indevida de projetos de interface já existentes e sim a capacidade de produzir interfaces melhores após a observação, definição e análise dos pontos fracos e fortes de protótipos em comparação com produtos semelhantes disponíveis no mercado.

4.1.3. Metas de usabilidade.

A usabilidade não possui apenas um único atributo, ela é composta por vários componentes que podem, em alguns casos, entrar em conflito. Normalmente esses componentes possuem pesos diferentes, variando de acordo com o projeto desejado, necessitando quase sempre de um estudo de prioridades para determinar seus graus de importância.

Os diferentes parâmetros da usabilidade podem ser operacionalizados e expressos em modelos mensuráveis. De acordo com CHAPANIS e BUDURKA *apud* NIELSEN (1993), antes de iniciar o projeto de uma nova interface é importante discutir quais são os parâmetros da usabilidade de interesse para o projeto e a especificação das metas para sua interface em relação à usabilidade.

Pode ser que nem sempre haja recursos disponíveis para se obter dados estatísticos confiáveis a respeito das métricas de usabilidade que permitam estabelecer a especificação exata das metas; contudo, o importante é que se obtenha uma idéia básica do nível de usabilidade desejado para o projeto.

Metas de usabilidade são razoavelmente fáceis de se atribuir para novas versões de sistemas já existentes ou para sistemas que possuem concorrentes claramente definidos no mercado a partir de uma análise competitiva. Geralmente o nível mínimo de usabilidade aceitável se iguala entre os aplicativos similares e, em alguns casos, a principal meta da usabilidade, após uma considerável melhora na interface, é fazer com que o usuário prefira mudar de sistema, adotando para si o que apresenta melhor interface entre os concorrentes similares.

Para sistemas novos, sem qualquer parâmetro de comparação com produtos similares, as metas de usabilidade são mais difíceis de serem definidas. Neste caso, pode-se estabelecer um conjunto de tarefas específicas do produto, para que diversos especialistas em usabilidade determinem as metas necessárias para realizá-las, a fim de possibilitar ao usuário um desempenho satisfatório.

Outra alternativa é usar o mesmo grupo de tarefas com usuários, ao invés de especialistas em usabilidade. Entretanto, diversos projetos desenvolvidos a partir das observações dos usuários têm apresentado um maior número de falhas em relação à satisfação durante a utilização real do produto.

4.1.3.1. Análise do impacto financeiro.

Esta análise deve ser especificada basicamente durante o mesmo tempo em que são estabelecidas as metas de usabilidade, tendo como objetivo a obtenção de uma idéia aproximada do impacto financeiro da usabilidade do sistema. Para isso, faz-se necessária uma comparação entre o investimento realizado no projeto de usabilidade e o retorno financeiro proporcionado por este. Sua estimativa envolve o número de usuários do sistema, o número projetado de usuários e a estimativa do nível dos salários envolvidos, custos diversos e o tempo aproximado de utilização do sistema em relação à produtividade. O custo do tempo disponibilizado para a utilização do sistema agrega outros valores tais como pensões, benefícios, encargo trabalhistas e, em alguns casos, o valor do aluguel do espaço utilizado. Todo e qualquer fator que possa interferir direta ou indiretamente no custo final do produto deve ser considerado.

Nos casos em que o produto é desenvolvido por empresa terceirizada, a análise do impacto financeiro apresenta basicamente dois componentes:

1. A estimativa do impacto na organização que desenvolve o projeto de usabilidade, para se determinar o tamanho do orçamento a ser consumido;
2. A estimativa do impacto nas organizações do usuário para determinar os recursos necessários a serem disponibilizados para a implantação do projeto.

Outras informações necessárias a esse tipo de análise podem estar disponíveis tanto no departamento financeiro apresentando, como base de comparação, propostas ou projetos já realizados anteriormente, como no departamento pessoal em relação a custos de contratação de diversos profissionais alocados em outros projetos.

4.1.4. Projeto paralelo

De acordo com NIELSEN *et al apud* NIELSEN (1993), um bom recurso usado freqüentemente ao se iniciar um projeto é utilizar o "projeto paralelo", em que vários projetistas trabalham em diferentes projetos preliminares. O objetivo principal é

explorar alternativas de projetos diferenciados para, posteriormente, efetuar comparações assimilando para o projeto final as principais características de cada um.

Normalmente é muito mais vantajoso ter um grupo de projetistas trabalhando individualmente ao invés de um grande número envolvido no mesmo projeto, desde que os projetos individuais apontem para o mesmo grupo de idéias básicas do projeto final.

Recomenda-se que os projetistas não devem discutir seus projetos entre si antes de terminarem o esboço de sua interface.

Uma variação do projeto paralelo é conhecida como "projeto paralelo diversificado", no qual diferentes projetistas se dedicam a aspectos diferentes do mesmo projeto. Como exemplo, pode-se propor a um projetista o desenvolvimento de uma interface direcionada a usuários novatos e, ao mesmo tempo, a um outro, o desenvolvimento de uma interface voltada a usuários peritos, ambos com o foco no mesmo projeto. Como resultado final pode-se obter um projeto apresentando interfaces configuráveis tanto para usuários novatos como para experientes.

A vantagem financeira desta técnica, desde que se tenha disponível um número considerável de projetistas, é a possibilidade de executar vários projetos preliminares ao mesmo tempo, diminuindo o tempo de desenvolvimento e, conseqüentemente, o de comercialização do produto final.

4.1.5. Projeto participativo.

Após o início da fase de execução do projeto, um conjunto representativo de usuários que utilizarão o sistema deve ter acesso a este, afim de que interajam com o mesmo para que possam formular perguntas e questionamentos. Os usuários devem ser envolvidos nessa fase do projeto através de encontros regulares com os projetistas.

Quanto maior e mais heterogêneo for o grupo de usuários consultados, maior será o nível de confiabilidade das informações; em contrapartida, um número muito grande de usuários representativos pode acarretar perda de tempo no processo, implicando em aumento de gastos. Esses dados devem ser considerados ao se definir um número de usuários pesquisados.

É importante se destacar que usuários representativos são todos que interagem diretamente com o sistema.

4.1.6. Coordenação da interface total.

Uma das características mais importantes da usabilidade é a consistência que deve ser aplicada em todas as diferentes mídias que formam a interface total do usuário, incluindo-se neste grupo a documentação, o sistema de ajuda *on line* ou qualquer outro tipo de tutorial informativo. Mesmo as empresas que produzem famílias de produto devem primar pela consistência de seus projetos, de forma a obter um padrão de interface do usuário.

Para alcançar a consistência total da interface é necessário que haja uma coordenação para o desenvolvimento do projeto nos vários aspectos da interface. Basicamente, essa coordenação pode ser feita por um profissional capacitado, contudo, nos grandes projetos, pode ser mais apropriada a utilização de um grupo maior de profissionais.

O estabelecimento de padrões de interface, discutidos no tópico abaixo, contribui significativamente para a obtenção da consistência total da usabilidade. Além disso, esse processo permite a sincronia de toda a equipe de desenvolvimento em relação à interface do usuário.

A consistência pode ser aumentada através do compartilhamento de códigos, onde os vários produtos de uma mesma família utilizem o mesmo código em diversos pontos semelhantes, gerando a consistência das interfaces.

4.1.7. Diretrizes

As diretrizes apresentam os princípios que devem ser seguidos no projeto de desenvolvimento para a interface do usuário. Os projetos devem possuir vários níveis diferentes de diretrizes:

- Diretrizes gerais, aplicadas para toda a interface do usuário;
- Diretrizes específicas de categoria, aplicáveis às várias espécies de sistemas existentes;
- Diretrizes específicas de produto, aplicadas ao produto individualmente.

Um exemplo de diretriz geral é estabelecer a necessidade de fornecer ao usuário um retorno sobre o estado do sistema e suas ações. Para a diretriz específica de categoria pode-se, por exemplo, assegurar que os objetos principais de interesse do

usuário sejam visíveis na tela e que seus atributos mais importantes sejam mostrados. No caso das diretrizes específicas de produto, um exemplo pode ser a representação gráfica de cada arquivo e subdiretório que tem sua representação indicada por um ícone, onde cada classe diferente é representada por um ícone diferente.

O Guia de Estilo e o Guia de Recomendação são exemplos de diretrizes utilizadas por projetistas e aplicadas na concepção de interfaces.

4.1.8. Prototipagem

Na maioria dos modelos tradicionais de engenharia de software o tempo de desenvolvimento é dedicado aos vários refinamentos intermediários do produto, onde a versão final do programa é produzida no último momento. Tal procedimento impede que os usuários testem a interface do produto durante os trabalhos intermediários até a conclusão final do projeto; como solução, separa-se a interface do usuário em um protótipo no qual o mesmo poderá interagir. Dessa forma a prototipagem possibilita ganho de tempo e de custo, permitindo que usuários reais possam avaliar a interface do produto em paralelo.

Existem basicamente dois tipos de prototipagem: a prototipagem vertical e a horizontal (Fig. 10).

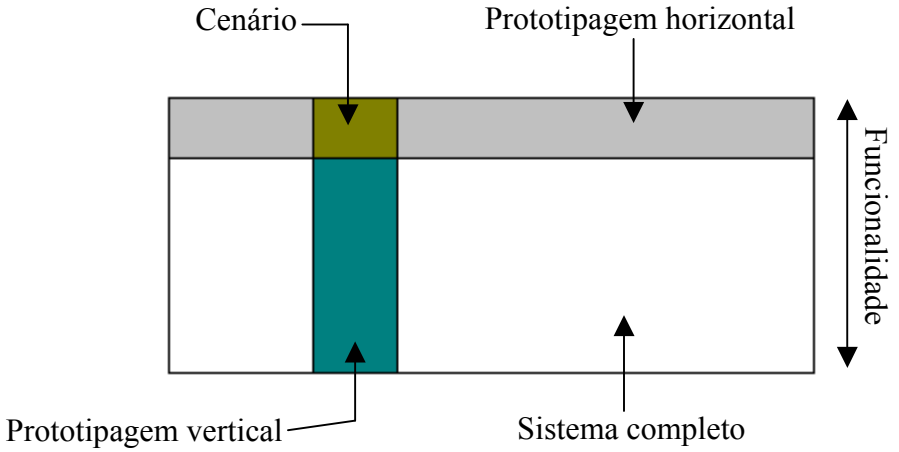


FIGURA 09 – As duas dimensões da prototipagem.

A prototipagem vertical apresenta sua funcionalidade em relação à profundidade, ou seja, os recursos apresentados são poucos, contudo, são simuladas as circunstâncias como tarefas reais para os usuários – como exemplo pode-se citar o

acesso a um banco de dados com informações reais possibilitando as diversas interações.

No caso da prototipagem horizontal a interface do usuário é apresentada completa, não permitindo um aprofundamento da sua funcionalidade. O protótipo horizontal é uma simulação onde nenhum trabalho real pode ser desempenhado, apenas sua interface é avaliada.

A principal vantagem de projetos que utilizam protótipos está na redução do número de recursos e do nível de funcionalidade que estes exigem para se obter um esboço funcional do projeto, colocando-se menos ênfase na eficiência de sua implementação. Tal característica pode fazer com que o próprio protótipo apresente uma velocidade de processamento aparentemente rápida em relação ao produto final acabado, entretanto, deve-se ter em mente que a função do protótipo é mais aplicada a conceitos de interface propriamente ditos.

Outras vantagens que devem ser destacadas:

- A possibilidade de estudos de medição de tempo de interação;
- A utilização de códigos mais simples;
- A qualidade do nível de confiança reduzido;
- Utilizar algoritmos simplificados, sem a necessidade de grande programação para obtê-lo.

4.1.9. Projeto interativo.

Dados obtidos a partir da visão da natureza do problema e do registro das seqüências de interação de usuários experientes, que demonstram situações de perda de tempo ou erros, podem ajudar a compreender as causas principais dos problemas de usabilidade, além de permitir a classificação formal das diferentes categorias de problemas (BOOTH *apud* NIELSEN, 1993).

Com base nos problemas de usabilidade e nas descobertas oportunas através de testes empíricos com usuários representativos, pode-se produzir uma nova versão de interfaces com qualidade.

A familiaridade com as opções do projeto, a observação dos usuários, criatividade e sorte são necessárias neste momento.

HOUDE *apud* NIELSEN (1993) apresenta um caso interessante de interatividade num projeto de interface gráfica para a manipulação de objetos tridimensionais na tela do computador. A questão trata do projeto de cursores em relação à movimentação e rotação de objetos.

Na primeira parte da interação o cursor apresentava um desenho na forma de uma mão agarrando; entretanto, para os usuários, era como se o cursor estivesse agarrando o vazio, uma vez que em qualquer local da tela o cursor apresentava a mesma caracterização.

A segunda interação repôs a imagem padrão do cursor e em cada objeto móvel foi inserida uma área ativa, de tal forma que, ao posicionar e pressionar o cursor na área ativa, a imagem personalizada do cursor na forma de mão agarrando surgiria. Novamente os usuários tiveram dificuldades na interação, uma vez que os mesmos não conseguiam localizar, de prontidão, a área ativa em determinados objetos, necessitando clicar em vários pontos da mesma.

A terceira interação propôs a introdução de múltiplas figuras de mão em diversas posições, a dificuldade encontrada nessa fase está no desejo dos usuários em visualizar as mais variadas posições para a imagem da mão, além das disponíveis – o formato da mão ao levantar uma lâmpada pode ser diferente ao levantar uma cadeira.

Finalmente, a quarta interação solucionou o problema, posicionando-se em volta de cada objeto uma moldura, cuja forma regular permitia a perfeita identificação desta como área ativa para efetuar a movimentação do objeto.

O exemplo demonstra que as mudanças feitas com o objetivo de solucionar problemas de usabilidade podem não ser as mais adequadas. Quando se refaz parte de um projeto com o objetivo de melhorar algum parâmetro de usabilidade é muito comum a ocorrência de impactos adversos sobre algum outro parâmetro. Em alguns casos, a resolução prévia de um problema pode fazer com que a interface piore para os usuários que não experimentaram o referido problema. Neste caso recomenda-se que a alteração só seja efetuada a partir da análise da interação do usuário diante do problema. A interface do usuário deve ser mudada e testada tão logo um problema de usabilidade seja detectado e compreendido, de modo a evitar que a permanência do referido problema seja esquecido.

Nem sempre é possível aplicar, de forma satisfatória, todos os procedimentos recomendados no ciclo de vida de usabilidade em um determinado projeto. Vários são os fatores que podem interferir na aplicação das recomendações, desde restrições no orçamento do projeto priorizando outras etapas dentro da engenharia de software, até problemas com usuários que temem perdas financeiras ou de *status* após a implantação do sistema.

Caso seja detectada a permanência ou o surgimento de algum problema de usabilidade após a aplicação completa do ciclo de vida, recomenda-se que o processo seja reiniciado, objetivando a detecção do referido problema, suas causas e prováveis soluções.

É importante destacar que tanto na prototipagem quanto no projeto interativo cabe a utilização de ferramentas como forma de apoiar a implementação. Guias de Estilo ou de Recomendações podem auxiliar no desenvolvimento desses estágios no sentido de direcionar o processo. Outra ferramenta que pode ser utilizada é o checklist, funcionando como recurso de verificação e validação dos protótipos obtidos.

Com base nos estágios descritos acima, o ciclo de vida da usabilidade apresenta três importantes atividades:

1. Coleta de dados e registros;
2. Elaboração e desenvolvimento do projeto;
3. Implementação e teste.

As figuras 11, 12 e 13 representam graficamente essas atividades.

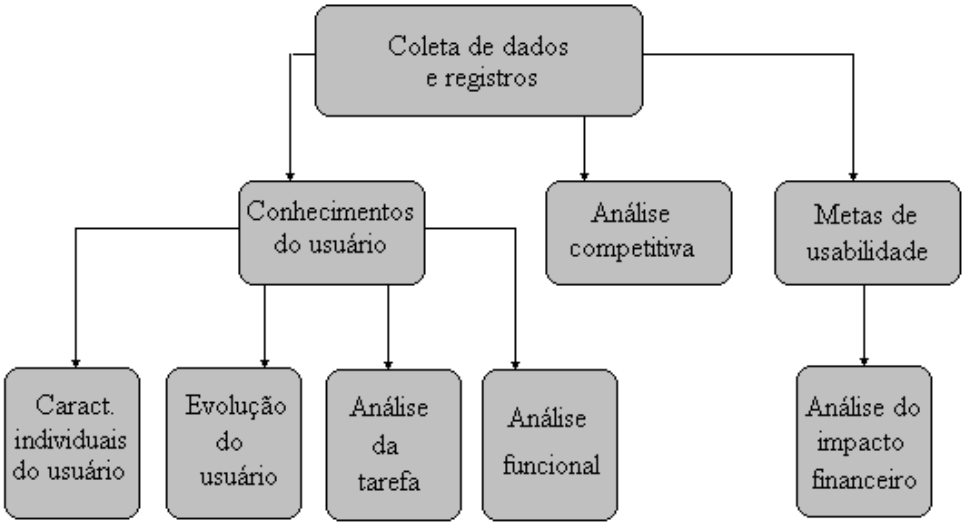


FIGURA 10 – Coleta de dados e registros.

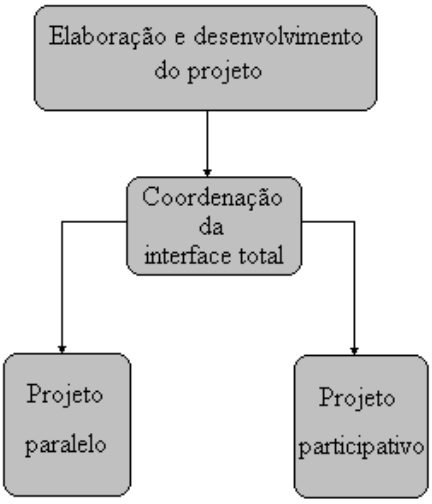


FIGURA 11 – Elaboração e desenvolvimento do projeto.

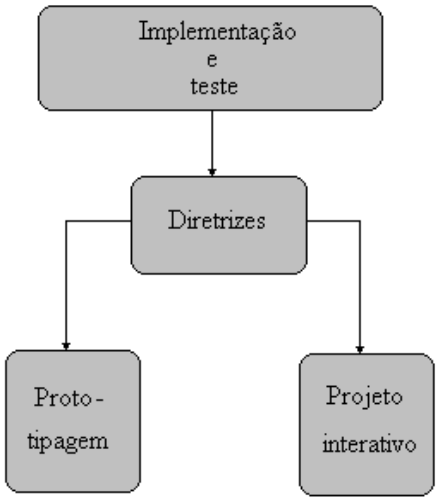


FIGURA 12 – Implementação e teste.

4.2. Guias de Estilos, Guias de Recomendações e Checklists

Entre as diversas ferramentas que apóiam a engenharia de software, duas terão destaque nesta pesquisa por apresentarem características voltadas para o projeto e verificação de software, além de serem aplicadas pela engenharia de usabilidade no desenvolvimento e verificação do ciclo de vida de interfaces ergonômicas.

4.2.1. Guia de Estilo

Seu objetivo é apresentar um framework com características comuns e que possam ser utilizadas tanto para a elaboração do projeto como para a implementação do mesmo. Este framework deve estabelecer as definições das metas e requisitos necessários para o sistema, devendo, inclusive documentá-los.

Suas recomendações são padronizadas com o intuito de manter a consistência no desenvolvimento de projetos similares. São desenvolvidos, em sua maioria, por grandes corporações e apresentam um conjunto de características aplicadas em seus sistemas proprietários, tais características são utilizadas na implementação de vários produtos desenvolvidos por outras empresas ou organizações. A reprodução dessas recomendações tem como objetivo manter a consistência no ambiente de trabalho.

Outros objetivos pretendidos na utilização do guia de estilo são:

- Redução do tempo de desenvolvimento – O guia de estilo apresenta as diretrizes básicas de desenvolvimento de um projeto, variações podem ser acrescentadas ou retiradas sem comprometer sua consistência e sem que seja necessário o desenvolvimento de um novo projeto do ponto inicial.
- Aumento na produtividade – Usuários terão maior facilidade em utilizar aplicativos que apresentarem um elevado grau de consistência, permitindo uma maior adaptação tanto para usuários avançados como para novatos.
- Diminuição do tempo gasto em treinamento - Menos tempo de assimilação em virtude da similaridade entre as aplicações, possibilitando que um único treinamento seja capaz de abranger todas elas.

4.2.2. Guia de Recomendações

Sua característica principal é apresentar um conjunto de recomendações mais genéricas em relação ao guia de estilo. São desenvolvidos com base em conhecimentos assimilados de forma empírica, através de observações, interações, entrevistas, entre outros, e posteriormente validados com testes feitos por especialistas ou aplicados por estes em grupos de usuários.

Diferentemente do guia de estilo, não são desenvolvidos por corporações e sim por pesquisas que buscam a padronização de critérios em diversas áreas do conhecimento.

4.2.3. Checklist

Também denominados de listas de verificação ou conferência, os checklists têm o objetivo de verificar e/ou validar a qualidade de um determinado produto.

São desenvolvidos a partir de uma lista de critérios, apresentados geralmente na forma de questionário, onde o avaliador deverá conduzir a verificação de acordo com os parâmetros estabelecidos no checklist.

Podem apresentar tanto aspectos gerais de uma avaliação como podem avaliar características particulares pertinentes a um projeto específico. Em alguns casos os checklists específicos podem derivar de uma avaliação mais abrangente.

Alguns checklists elaborados são acompanhados de informações adicionais que facilitam sua aplicabilidade como:

- Comentários – complementando definições que possam apresentar alguma forma de ambigüidade ou confusão;

- Notas explicativas – com o objetivo de esclarecer particularidades e um projeto específico;
- Exemplos positivos e negativos – como forma de demonstrar, mais claramente, inconsistências sutis;
- Figuras – representação gráfica, possibilitando a visualização das questões;
- Glossário – definições de palavras e termos técnicos específicos de um projeto.

As principais vantagens apresentadas na utilização de checklists são:

1. Avaliação sistematizada – Assim como no guia de estilo, todas as metas estão definidas, todas as questões estão padronizadas para serem efetivamente verificadas e avaliadas independentemente de quem seja o avaliador;
2. Aplicação realizada pela equipe de projetistas – Não há necessidade de se contratar um especialista, uma vez que todo o conhecimento necessário está disponibilizado no próprio checklist;
3. Fácil detecção de erro – Em função de sua padronização, o checklist permite que o avaliador detecte as falhas na execução do projeto, especificando sua ocorrência;
4. Redução de custo – O processo de avaliação por checklist permite uma redução no custo final do projeto, por apresentar questões objetivas, diminuindo o número de aplicações e o tempo consumido ao realizá-las;
5. Processo eficaz de avaliação – Ocorre uma diminuição da subjetividade no processo avaliativo, em função do checklist apresentar questões diretivas, elaboradas a partir de critérios claramente estabelecidos.

No capítulo que trata da Engenharia de Usabilidade, as ferramentas acima serão citadas como diretrizes de seu ciclo de vida.

CAPÍTULO V

5. DESENVOLVIMENTO DO GUIA DE RECOMENDAÇÕES ERGONÔMICAS PARA INTERFACES DE SOFTWARE EDUCATIVO

O objetivo desta pesquisa tem como foco o desenvolvimento ergonômico de interfaces em software educativo. Analisando, mais especificamente, ferramentas de concepção/condução ergonômica e ferramentas de avaliação ergonômica.

O grande número de aplicações educacionais existentes e o aumento do grau de exigência dos recursos humanos envolvidos, apontam para a necessidade de se conceber software de qualidade, devendo a mesma ser buscada nos diversos níveis do seu desenvolvimento, desde o projeto inicial até sua implantação e avaliação com o objetivo de satisfazer necessidades explícitas e implícitas.

Dentre esses aspectos analisados, encontra-se a interface, responsável pela interação direta entre sistema e o usuário e que deve apresentar características que facilitem sua manipulação de forma confiável e eficiente.

5.1. Detecção do Problema

Entre as diversas formas de se elaborar e avaliar interfaces ergonômicas para software pode-se destacar a utilização de guias de estilos ou de recomendações combinados com a aplicação de checklists.

A utilização de guias de estilos ou de recomendações e checklists que apresentam bases de conhecimentos diferenciados pode gerar diversos conflitos, uma vez que conduzirão os projetistas a aplicarem, no início do projeto, determinadas soluções e na fase de avaliação soluções diferenciadas.

O quadro acima descrito pode apresentar três situações que provavelmente afetaram o ciclo de vida da engenharia de usabilidade:

1. Se uma interface, projetada com base em um guia de recomendações especialmente desenvolvido para o software em questão, for implantada sem que

uma avaliação de usabilidade tenha sido realizada, há grandes possibilidades da interface apresentar falhas. De fato, a aplicação de um guia é uma iniciativa sujeita a falhas do projetista, e a falta de uma avaliação fará com que estas falhas cheguem até o usuário.

2. Se a referida interface for desenvolvida sem o apoio prévio de um guia de recomendações e somente aplicada a esta, uma técnica de inspeção de conformidades ergonômicas ao final de sua elaboração, os problemas detectados serão de difícil correção e poderão se reproduzir em outros projetos. Isto porque a inspeção não permitirá uma definição precisa da origem do problema, podendo corresponder a falhas estruturais do projetista ou inconsistência na definição dos critérios ergonômicos relevantes.
3. As duas situações acima descritas se reproduzirão em menor escala, caso a mesma interface seja desenvolvida com base em um guia de recomendações e avaliada por uma técnica de inspeção de conformidades ergonômicas que apresente critérios e quesitos parciais ou totalmente diferentes. Será como se parte o projeto não pudesse ser avaliada, e parte dos erros não pudesse ser controlada.

Em relação ao ciclo de vida da engenharia de usabilidade, responsável pelo projeto de elaboração de interfaces, este desacordo entre instrumentos pode representar uma série de problemas como:

- *Aumento de número de ciclos a serem refeitos para a detecção e correção do erro*.

  O ciclo de vida de usabilidade da interface precisará ser refeito para que se possa detectar os problemas, causas, conseqüências e possíveis soluções.

- *Gasto de tempo na localização de um guia de recomendações e/ou de uma ferramenta de inspeção mais adequada ao produto*.

O aumento do tempo de execução do projeto pode ser considerável em função da dificuldade de se localizar um guia de recomendações e de uma técnica de inspeção que estejam em conformidade.

- *Aumento no custo do projeto.*

Os gastos com o projeto serão ainda maiores em função da contratação de um especialista em ergonomia.

- *Aumento no tempo de conclusão do produto*.

Qualquer que seja a estratégia adotada será necessária uma redefinição do cronograma, no sentido de dilatar o tempo de conclusão do projeto.

- *Aumento no custo final do produto*.

Todas as hipóteses acima descritas concorrerão para um aumento de gastos com horas de trabalho, alocação de dependências e equipamentos, contratação de profissionais especializados, entre outros, podendo interferir no valor final do produto.

5.2. Solução Proposta

Com o objetivo de solucionar os possíveis problemas gerados a partir da utilização de guia de recomendações ou de estilo e/ou checklists que apresentem conhecimentos diferenciados.

Recomenda-se que sejam utilizados guias de estilos ou de recomendações em combinação com checklists, desde que ambos apresentem o mesmo conteúdo e a mesma estrutura organizacional, ou seja, que o projeto tenha, nas fases de concepção, elaboração, verificação e avaliação, a mesma base de conhecimento.

Para produzir esta situação, se faz necessária uma análise detalhada das diversas ferramentas e procedimentos que envolvem a condução e avaliação de projetos de interfaces, tendo seu direcionamento voltado para ambientes educacionais informatizados.

Em função desta proposta as seguintes etapas foram estabelecidas:

- Análise do estado da arte das ferramentas de concepção/condução e de avaliação ergonômicas;
- Estratégia a ser adotada;
- Concepção da ferramenta.

5.3. O Estado da Arte das Ferramentas de concepção/condução e de avaliação ergonômica.

Diversas ferramentas desenvolvidas com o intuito de conduzir e avaliar a concepção de interfaces ergonomicamente corretas têm sido apresentadas ao longo, principalmente, das três últimas décadas. Algumas dessas ferramentas desenvolvidas

para conduzir a concepção das interfaces são direcionadas a um tipo específico de software que apresenta um nível de particularidade determinante e um forte direcionamento para a padronização de linhas específicas de produtos, resultando na elaboração de um *Guia de Estilo*. Outras apresentam características mais voltadas ao projeto de ambientes interativos, com o objetivo focado apenas na qualidade de usabilidade da interface, sem se preocupar com produto ou plataforma específicos, gerando os *Guias de Recomendações* Ergonômicas.

HIX & HARTSON *apud* NIELSEN (1993) afirmam que:

> "*....as normas, os guias de estilo e de recomendações são formados por regras e heurísticas que dão suporte ao desenvolvimento de software centrados no usuário e que fornecem aos projetistas uma síntese acumulada de conhecimento, resultante de pesquisas e experimentos que, juntamente com outras práticas de IHC, contribuem para o projeto de interfaces com usabilidade*."

A maioria dos guias, sejam de estilo ou de recomendações, não são desenvolvidos para projetos específicos de interfaces, apresentando, no máximo, um direcionamento voltado para software ou sistemas operacionais.

Dentre os guias de recomendações e de estilos, os mais conhecidos mundialmente estão:

- Guias de recomendações:

a) *Guidelines for designing user interface software,* proposto por SMITH e MOSIER *apud* NIELSEN (1993).

b) *Principlies and Guidelines in Software User Interface Design,* proposto por MAYHEW (1992).

c) *Guide Ergonomique de la présentation des applications hautement interactives,* proposto por VANDERDONCKT e BODART *apud* CYBIS (1997).

- Entre os guias de Recomendações nacionais constam:

a) *Guia de Estilo para serviços de informação em Ciência e Tecnologia via Web*, proposto por PARIZOTTO (1997).

b) *Guia de Estilo para Seleção de Objetos de interação*, proposto por SCHUHMANHER (1988).

a) *Norma ISO 9241 – Requisitos Ergonômicos para trabalho em escritório*, Parte 10 (princípios de diálogos) (1993);

Obs.: A norma acima referida, até o momento, não obteve sua publicação como norma brasileira.

c) *GEPESE – Guia de Recomendações para software educacional,* proposto por VALIATI, *et al*. (2000), que é uma produção nacional voltada para o domínio de software educacional.

- Guias de estilo:

a) *The Windows Interface Guidelines - A Guide for Designing Software* - Microsoft® Windows®.

b) *OSF/Motif Style Guide* - Motif™.

c) *ISO 9241 (International Organization for Standardization)*

d) *Guia de Estilo para Seleção de Objetos de interação,* proposto por SCHUHMANHER (1988).

A pesquisa em questão baseia-se nas Técnicas analíticas, as quais são compostas por:

- Análise hierárquica da tarefa;
- Avaliação heurística;
- Inspeção cognitiva;
- Inspeções ergonômicas via Checklist.

A inspeção ergonômica via Checklist é baseada em uma lista de verificação a ser utilizada por profissionais que não sejam, necessariamente, especialistas em ergonomia e que tenha como objetivo detectar problemas na construção de interfaces. O mais importante nesse tipo de inspeção é a qualidade do checklist em função da falta de qualificação técnica em ergonomia por parte dos avaliadores.

Dentre os Checklists para a avaliação de software educacional, os mais conhecidos internacionalmente são:

a) *Educational Products Information Exchange, desenvolvido pelo instituto EPIE* - Educational Products Information Exchange, em Nova York;

b) *The School Microware User Software Review Program (SMUSRP),* desenvolvido pela empresa Dresdem Associados, nos EUA;

c) *Courseware Report Card - Evaluation of Microcomputer Programs for Educational*, também dos EUA.;

d) *Minnesota Educational Computing Consortium*, desenvolvido pelo governo do estado de Minnesota, EUA;

e) *CONDUIT Package Evaluating Form for Microcomputer-Based Instructional Materials*, desenvolvido pela Universidade de Iowa, na IA, em conjunto com o National Science Foudation (NSF) e o Fund for the Improvement of Post Secundary Education, nos EUA.;

f) *Software Evaluation Form*, do National Council of Teatchers of Mathematics, EUA;

g) *Scholastics Software Evaluation Form*, desenvolvido por POIROT & BILLINGS (1982), nosEUA;

h) *Checklist for Microcomputer Program Revision* do Programa SOFTSWAP da Microcomputer Center na cidade de Redwood, na Califórnia;

i) *American MicroSift Evaluators Guide* – método para avaliação de sistemas instrucionais baseados em computador, (1980-1981);

j) *Evaluation and Selection of courseware development software*, desenvolvido por DAVIDOVE (1987), EUA.

- Entre os checklists nacionais destacam-se:

a) *Ergolist*, desenvolvido em colaboração entre o SoftPólis, núcleo Softex-2000 e o LabUtil, UFSC;

b) *TICESE - Técnica e Inspeção de Conformidade Ergonômica de Software Educacional*, proposto por GAMEZ (1998).

5.4. Análise de alguns Guias de Recomendações Existentes

5.4.1. *Guia de Estilo para serviços de informação em Ciência e Tecnologia via Web*, proposto por PARIZOTTO (1997).

Este guia de recomendações tem como objetivo auxiliar projetistas de páginas Web na elaboração de sites que apresentem informações acadêmicas sobre ciência e tecnologia. Está disponível na forma de página Web.

É formado por seis atributos gráficos: Cores, Fontes, Fundos, Ícones, Layouts e Textos e um glossário de termos técnicos. Cada atributo apresenta breves considerações sobre sua função, importância ou aplicação e um conjunto de recomendações e observações na forma de itens.

Algumas recomendações e observações apresentam, quando necessário, referências bibliográficas e/ou *links* com a função de acionar o glossário para termos técnicos.

5.4.2. *GEPESE – Guia de Recomendações para software educacional, proposto por* VALIATI, *et al*. (2000).

O guia tem como objetivo auxiliar projetistas de software educacionais sem formação e conhecimentos na área de IHC (Interação Humano-Computador). O tipo de interface para o qual foram projetadas suas recomendações são utilizadas em softwares educacionais do tipo hipertexto-hipermídia informativo em CD-ROM.

Segundo VALIATI *et al*. (2000):

> "*...o guia é formado por recomendações sugestivas, relacionando aspectos de usabilidade e questões educacionais para a seleção e configuração de elementos de interface tais como: layout, sons, animações e vídeo, envolvendo outros objetos de interação tipicamente usados*".

O guia abrange basicamente os seguintes atributos: ícones, imagens, layout, menus, sons e texto.

Apresenta uma introdução sobre o guia, além de um glossário para termos técnicos e índices alternativos para determinados assuntos. Suas recomendações são numeradas, apresentando figuras para representar graficamente os elementos da interface juntamente com textos explicativos e regras, além de exemplos e justificativas em algumas recomendações.

Cada elemento da interface apresentado possui uma justificativa da importância da seleção e configuração corretas e regras sugestivas apresentadas de forma numerada.

5.4.2.1. *Contribuição do Guia – GEPESE*

Dentre os guias de recomendações anteriormente citados, o Guia – GEPESE contribuiu para esta pesquisa como sustentação bibliográfica, servindo como parâmetro na busca de embasamento teórico, onde VALIATI *et al*. (2000) descreve a seguinte situação:

Após o desenvolvimento da interface de um protótipo de software educacional do tipo hipertexto/hipermídia informativo, com a utilização de um guia de recomendações (Guia-GEPESE), foi iniciado seu processo de avaliação utilizando as seguintes técnicas:

- Análise heurística
- *Checklist* TICESE – Técnica de Inspeção de Conformidades Ergonômicas em Software Educacional.

Devido à dificuldade de aplicação, tanto da análise heurística como da TICESE, não foi possível verificar o grau de aplicabilidade e conformidade de todas as recomendações constantes no Guia-GEPESE. A solução encontrada foi utilizar o próprio guia para a verificação das recomendações.

A pesquisa aponta para alguns fatos relevantes detectados:

- Há uma resistência natural, por parte dos projetistas, em atender totalmente as recomendações referentes à interface contidas no guia.
- A utilização isolada do guia de recomendações, ou de qualquer outra ferramenta de concepção e avaliação, não garante a usabilidade da interface.

5.5. Análise de alguns checklists existentes

5.5.1. *Ergolist, desenvolvido em colaboração entre o SoftPólis, núcleo Softex-2000 e o LabUtil, UFSC*.

Este checklist é composto por três módulos:

- Módulo Checklist – Permite a inspeção sistemática da qualidade ergonômica da interface com o usuário.
- Módulo de Questões – Possibilita conhecer de modo informal as questões que compõem o módulo Checklist.

- Módulo de Recomendações – Composto de recomendações ergonômicas que podem auxiliar nas decisões de projeto de interfaces com o usuário.

Sua estrutura apresenta um menu de critérios, um glossário para termos técnicos e uma base de informações (denominada "Mais sobre...") que apresenta justificativas, exemplos de recomendações e comentários de cada critério, além de referências bibliográficas.

Cada critério apresenta um número variado de questões e cada questão apresenta alternativas quanto sua aplicabilidade e sua conformidade, sendo que nesta última é observada a conformidade parcial ou total.

A aplicação do Ergolist pode ser feita on line em conexão via internet com seu site (http://www.labiutil.inf.ufsc.br/ergolist/index.html). Ao final da aplicação de cada critério ou da avaliação total, o Ergolist permite a emissão de um relatório informando os resultados alcançados e eventuais observações.

O Ergolist possui um total de dezoito critérios:

Presteza, Agrupamento por localização, Agrupamento por formato, Feedback, Legibilidade, Concisão, Ações Mínimas, Densidade Informacional, Ações Explícitas, Controle do Usuário, Flexibilidade, Experiência do Usuário, Proteção contra erros, Mensagens de erro, Correção de erros, Consistência, Significados e Compatibilidade.

Por ser um checklist composto por uma técnica de avaliação rápida, é recomendado a apoiar na inspeção de interfaces, buscando seus defeitos ergonômicos mais flagrantes.

5.5.1.1. *Contribuição do checklist – Ergolist*

Durante a análise do Ergolist, verificou-se que sua estrutura organizacional apresenta uma formatação clara e simplificada, tanto nos critérios como nas questões utilizadas. A concepção da ferramenta proposta pela pesquisa adotou a organização do Ergolist nos seguintes aspectos:

- Critérios – Organizados de forma seqüencial, numerada em ordem crescente, apresentando definição e justificativa, além de um número variável de questões aplicáveis.
- Questões – Apresentação textual, comentários e exemplos.

Esses aspectos, quando formatados para a concepção da ferramenta proposta, sofreram algumas modificações que serão relatadas posteriormente.

5.5.2. *TICESE - Técnica e Inspeção de Conformidade Ergonômica de Software Educacional*, proposto por GAMEZ (1998).

Foi elaborada a partir da revisão bibliográfica de SCAPIN e BASTIAN *apud* GAMEZ (1998) e de sues critérios, os quais foram estendidos para serem aplicados a software educacional, tendo como proposta fornecer diretrizes para o avaliador, afim de que o mesmo possa realizar a inspeção de conformidade ergonômica de software educacional, integrando tanto os aspectos pedagógicos como os de usabilidade.

O processo é todo conduzido através de um manual composto por três sessões:

- Instruções para a aplicação da técnica;
- Descrição detalhada de critérios e sub-critérios de avaliação com justificativas, além de apresentar a taxonomia de software educativo;
- Formulário de inspeção na forma de checklist a ser utilizado pelo verificador.

a) Instruções para a aplicação da técnica

Esta sessão é composta pelos seguintes itens, que devem ser executados pelo avaliador:

- Reconhecimento do software para compreender seu funcionamento;
- Reconhecimento da técnica de inspeção através da leitura dos seguintes tópicos: definição de critérios, taxonomia do software, formulário de inspeção e tratamento quantitativo;
- Aplicação efetiva da técnica.

b) Descrição detalhada de critérios e sub-critérios de avaliação e taxonomia de software educativo

Esta sessão apresenta a definição e justificativa dos módulos e critérios de avaliação.

Dentro desta encontra-se o módulo de classificação que tem caráter introdutório, objetivando classificar o software a partir de seus atributos e proposta

pedagógica, e o módulo de avaliação, com o objetivo de avaliar a conformidade ergonômica do software educacional, tanto em relação aos recursos pedagógicos e de apoio, como sobre aspectos ergonômicos da interface.

O módulo de avaliação é composto de três partes:

b.1. Avaliação da documentação, dividida em:

- Dados de identificação do produto – com a identificação dos pré-requisitos técnicos e pedagógicos, tanto da documentação impressa como da *on-line* do software;
- Qualidade da informação impressa – em relação a manuais, embalagem, ficha de descrição e toda documentação impressa pertencente ao produto. Os critérios utilizados nesta fase são: Presteza, Legibilidade, Agrupamento de Itens, Densidade Informacional, Consistência e Significado dos Códigos e Denominações.

Cada critério apresenta uma descrição de sua definição e justificativa.

b.2. Avaliação do produto – utiliza os critérios representados no quadro abaixo:

Critérios	**Sub-critérios**
1 – Condução	1.1 – Presteza
	1.2 – Qualidade das opções de Ajuda
	1.3 – Legibilidade
	1.4 – Feedback imediato
	1.5 – Agrupamento e distinção de itens 1.5.1 – por localização 1.5.2 – por formato
2 – Adaptabilidade	2.1 – Flexibilidade
	2.2 – Consideração da Experiência do Utilizador
3 – Controle Explícito	3.1 – Ações Explícitas do Utilizador
	3.2 – Controle do Utilizador
4 – Recursos de Apoio à Compreensão dos Conteúdos	

5 – Gestão de Erros	5.1 – Correção de Erros
	5.2 – Qualidade das Mensagens de Erro
	5.3 – Proteção Contra Erros
6 – Avaliação da Aprendizagem	
7 – Carga de Trabalho	7.1 – Carga Informacional
	7.2 – Brevidade 7.2.1 – Concisão 7.2.2 – Ações Mínimas
	7.3 – Densidade informacional
8 – Significado dos Códigos e Denominações	
9 – Homogeneidade	
10 – Compatibilidade	

Quadro II – Critérios que compõem o checklist TICESE.

b.3. Avaliação contextual – tem como objetivo complementar o anterior, auxiliando no processo de tomada de decisão em relação à adoção e implementação de um software de acordo com o contexto específico da instituição de ensino. Apresenta como critério a adequabilidade.

Ainda nessa sessão encontra-se a taxonomia de software educacional, apresentando a classificação de acordo com as seguintes categorias:

- Exercício e Prática;
- Tutorial;
- Sistemas Tutoriais Inteligentes (STI);
- Simulação e Modelagem;
- Jogos Educativos;
- Informativos;
- Hipertexto/Hipermídia.

c) Formulário de inspeção na forma de checklist

O formulário de inspeção divide-se em:

c.1. Módulo de Classificação, composto por:

- Classificação da modalidade – de acordo com a taxonomia de software educacional;
- Identificação da Abordagem Pedagógica;
- Complexidade Cognitiva.

c.2. Módulo de avaliação – contendo os critérios anteriormente citados.

5.5.2.1. *Contribuição do checklist – TICESE*

Dentre os checklists analisados, a TICESE foi selecionada como ferramenta de avaliação ergonômica para compor o par concepção/avaliação proposto pela pesquisa.

5.6. Estratégia Adotada

De acordo com o objetivo da pesquisa e com base nas análises realizadas em relação às ferramentas de concepção e avaliação de interface, além do enfoque estar direcionado para software educacional, detectou-se três possibilidades:

5.6.1. Utilização de ferramentas já concebidas – Selecionar um par de ferramentas de concepção e avaliação existentes, como base nos estudos realizados nesta pesquisa.

Tal solução apresentou um elevado grau de dificuldade para sua realização, em função de alguns aspectos:

- Localização de duas ferramentas que compartilhassem da mesma base de conhecimentos.
- Necessidade de que ambas apresentassem sua aplicabilidade voltada para software educativo.
- Critérios, recomendações e questões, quantitativamente semelhantes.

5.6.2. Utilização de uma ferramenta de concepção existente – Seria necessária a realização de duas etapas distintas:

- A escolha de uma ferramenta de concepção que tivesse sua aplicabilidade voltada para o projeto de interface de software educativo.

- O desenvolvimento de uma ferramenta de avaliação, contendo a mesma base de conhecimento utilizada pela ferramenta de concepção adotada, além da sincronização de seus critérios, recomendações e questões.

5.6.3. Utilização de uma ferramenta de avaliação existente - Seria necessária a realização de duas etapas no sentido oposto à situação anterior, ou seja:

- A escolha de uma ferramenta de avaliação que tivesse sua aplicabilidade voltada para o projeto de interface de software educativo.
- O desenvolvimento de uma ferramenta de concepção, contendo a mesma base de conhecimento utilizada pela ferramenta de avaliação adotada, além da sincronização de seus critérios, recomendações e questões.

A análise das três hipóteses e do referencial bibliográfico disponibilizado pela pesquisa, deram o embasamento necessário para se optar pela terceira hipótese, ou seja, a utilização de uma ferramenta de avaliação já existente, propondo, posteriormente, uma ferramenta de concepção. No caso da pesquisa em questão, elaborou-se um Guia de Recomendações.

5.7. Justificativa da escolha da TICESE

Após a definição da estratégia a ser adotada, o passo seguinte foi definir qual ferramenta de avaliação seria a mais adequada para utilizada na pesquisa.

Dentre as técnicas de avaliação encontradas, o checklist TICESE apresentou condições favoráveis de utilização, em função do seu alto grau de abrangência, que pode ser observado a partir dos seguintes pontos:

- Ter seu desenvolvimento voltado para a avaliação de ambientes educacionais informatizados;
- Considerar tanto os aspectos pedagógicos, como os de usabilidade;
- Disponibilizar, na íntegra, material bibliográfico tanto do processo de elaboração como de seu manual de avaliação.
- Possuir um vasto material bibliográfico desenvolvido com base em sua análise.

Critérios	**Sub-critérios**
1 – Condução	1.1 – Presteza
	1.2 – Feedback imediato
	1.3 – Legibilidade
	1.4 – Agrupamento e distinção de itens *1.4.1 – por localização* *1.4.2 – por formato*
	1.5 – Qualidade das opções de Ajuda
2 – Carga de Trabalho	*2.1 – Brevidade* *2.1.1 – Concisão* *2.1.2 – Ações Mínimas*
	2.3 – Densidade informacional
	2.4 – Carga Mental ou Informacional
3 – Controle Explícito	3.1 – Ações Explícitas do Usuário
	3.2 – Controle do Usuário
4 – Recursos de Apoio à Compreensão dos Conteúdos	
5 – Adaptabilidade	5.1 – Flexibilidade
	5.2 – Consideração da Experiência do Usuário
6 – Gestão de Erros	*6.1 – Proteção Contra Erros*
	6.2 – Qualidade das Mensagens de Erro
	6.3 – Correção de Erros
7 – Avaliação da Aprendizagem	
8 – Homogeneidade	
9 – Significado dos Códigos e Denominações	
10 – Compatibilidade	

Quadro III – Critérios que compõem o guia de recomendações (modificados em relação a TICESE).
Obs.: Os itens destacados em *itálico* sofrem mudança de ordenação e/ou de nomenclatura.

5.7.1. Critérios que sofreram alteração.

Alguns dos critérios utilizados na elaboração do guia de recomendações, sofreram modificação em relação à quantidade de questões apresentadas.

A análise da relação entre as definições dos critérios e suas respectivas questões, constatou que o checklist TICESE apresenta algumas inconsistências no que diz respeito à correta localização de suas questões. Tal constatação resultou na redefinição de alocação destas, em relação a um grupo de critérios apresentados no quadro a seguir.

Critérios	Sub-critérios
1 – Condução	1.1 – Presteza
	1.2 – Feedback imediato
	1.3 – Legibilidade
2 – Carga de Trabalho	2.1 – Brevidade 2.1.1 – Concisão 2.1.2 – Ações Mínimas
3 – Controle Explícito	3.1 – Controle do Usuário
4 – Recursos de Apoio à Compreensão dos Conteúdos	
5 – Adaptabilidade	5.1 – Flexibilidade
6 – Gestão de Erros	6.1 – Proteção Contra Erros
	6.2 – Qualidade das Mensagens de Erro
	6.3 – Correção de Erros
7 – Significado dos Códigos e Denominações	
8 – Compatibilidade	

Quadro IV – Critérios que sofreram alterações no guia de recomendações.

5.7.2. Composição do Guia de Recomendações

O guia é composto por:

- Critérios – Classificados numericamente em ordem crescente e que apresentam:
  - Definição – Especifica a base de conhecimento em que atua.
  - Justificativa – Apresenta a necessidade e a funcionalidade de cada critério.
- Recomendações – Podem apresenta os seguintes itens:
  - Comentários – Define melhor a recomendação, contribuindo para sua compreensão.
  - Exemplos textuais – Reforça o entendimento através de situações reais ou hipotéticas
  - Exceções – Apresentar possíveis situações adversas, não aplicáveis à recomendação.

- Figuras – Demonstra, graficamente, a recomendação em questão
- Referências Bibliográficas – Indicação do embasamento teórico apresentado pela recomendação.
- Glossário – Apresenta a relação de termos técnicos encontrados no Guia de Recomendação e seus significados.

5.7.3. Formatação do Guia de Recomendações

A formatação do guia se divide em duas partes:

- Os critérios – Apresentam tanto sua definição como sua justificativa, formatadas em duas colunas com conteúdos somente textuais.
- As recomendações – Apresentam uma formatação em duas colunas definidas da seguinte forma:
- Coluna esquerda – Contém comentários, exemplos textuais, exceções e referências bibliográficas.
- Coluna direita – Contém figuras, devidamente identificadas por rótulos.

5.8. Resultados pretendidos após a aplicação do Guia de Recomendações

Como contribuição para o desenvolvimento de interfaces ergonômicas de software educativo, essa pesquisa pretendeu obter os seguintes resultados durante a fase de avaliação do projeto:

5.8.1. *Uma interface que apresente o maior número possível de questões aplicáveis em sua avaliação* – São consideradas questões aplicáveis aquelas que são pertinentes à interface analisada. Quanto maior for a identidade entre os critérios de elaboração da interface e as questões estabelecidas para avaliá-la, menor será o número de questões não aplicáveis durante sua avaliação.

5.8.2. *Uma interface que apresente o maior número possível de questões em total conformidade* – Questões consideradas em conformidade total são aquelas que atendem a todos os aspectos da avaliação. A interface terá um número maior de questões em total conformidade, quanto maior for a identidade entre os critérios de elaboração da interface e os aspectos envolvidos na avaliação ergonômica. O que possibilitará uma avaliação mais consistente.

5.8.3. *Uma interface com redução do número de características que obedecem, em parte, aos quesitos recomendados* – A interface deve apresentar, em sua avaliação, uma diminuição de quesitos com conformidade parcial, em função do sincronismo existente entre as fases de desenvolvimento e avaliação do projeto, desde que sejam aplicadas corretamente as recomendações durante o desenvolvimento.

5.8.4. *Maior rapidez na detecção de falhas e suas prováveis causas* – Teoricamente, pode-se supor que haverá uma diminuição das possibilidades de erros causados pela utilização de ferramentas que contenham regras baseadas em critérios diferenciados. Se as ferramentas forem desenvolvidas com a mesma base de conhecimento, pode-se supor que o erro tenha ocorrido na fase de desenvolvimento. Entretanto, se um item da interface não for contemplado em função da utilização de ferramentas que utilizem base de conhecimento diferenciadas, não será possível avaliar se o mesmo apresenta conformidade ou não, impedindo, conseqüentemente, a detecção de possíveis falhas na fase de desenvolvimento.

5.8.5. *Redução do tempo de conclusão do projeto da interface* – A aplicação de ferramentas que utilizem a mesma base de conhecimento, se corretamente aplicadas, podem reduzir o número de voltas dadas no ciclo de vida de usabilidade em função da diminuição considerável de erros.

5.8.6. Redução do custo final do projeto – Havendo redução do tempo de conclusão do projeto, conseqüentemente haverá redução de gastos com horas trabalhadas, locação de equipamentos, etc. Refletindo de forma positiva no custo final do projeto.

CAPÍTULO VI

6. CONCLUSÃO E TRABALHOS FUTUROS

6.1. Considerações Finais

A pesquisa aqui apresentada teve como objetivo geral introduzir o desenvolvimento de interfaces ergonômicas para software educativo em uma única base de conhecimento. Tal objetivo visa contemplar tanto sua concepção como sua avaliação, a partir da análise dos conhecimentos inseridos nos critérios e questões do checklist TICESE.

A análise dos critérios e questões do referido checklist deram sustentação para a elaboração de um Guia de Recomendações, cuja aplicabilidade está voltada para critérios ergonômicos de interface de software educativo. A partir dessa análise foi possível, também, estabelecer a sincronia entre o guia de recomendações e o checklist TICESE, tendo em comum a base de conhecimentos.

A pesquisa não tem o intuito de comprovar que basta a utilização do par Guia de Recomendação/Checklist, desenvolvidos a partir da mesma base de conhecimento, para se obter uma interface ergonomicamente correta, pois considera fundamental a aplicação, em conjunto, de outras técnicas analíticas. Sua proposta é o um aumento de eficácia e eficiência na condução do ciclo de vida do projeto de interfaces para ambientes educacionais informatizados.

Limitações foram encontradas durante a concepção do guia de recomendações, em função de algumas inconsistências detectadas nas questões do checklist, no que diz respeito à sua classificação em relação a um determinado critério ergonômico. Tal situação requereu uma redefinição da questão, quando convertida em recomendação, para outro critério mais apropriado.

Por ser essencialmente teórica, a pesquisa não produziu uma aplicação do guia de recomendações proposto, o que poderia produzir dados reais estatísticos capazes de validá-la experimentalmente.

O guia de Recomendações aqui desenvolvido, apresenta um conjunto de informações que, reunidas, podem melhor conduzir o projetista na elaboração de interfaces ergonômicas, pois, além de disponibilizar informações relativas a critérios e recomendações, trata as exceções, apresenta exemplos textuais e disponibiliza uma grande variedade de telas explicativas como exemplos visuais.

Conclui-se, finalmente, que os objetivos propostos foram alcançados e que se faz necessário um tratamento, cuidadosamente elaborado, para se conceber interfaces ergonômicas em software educativo que apresentem eficácia, eficiência e satisfação no seu uso.

6.2. Indicações para trabalhos futuros.

Dando continuidade a esta pesquisa pretende-se desenvolver os seguintes trabalhos:

- Reavaliação da estrutura do checklist TICESE, no que diz respeito à classificação de suas questões em relação aos critérios definidos, objetivando classificá-las adequadamente.
- Aplicação da atual proposta como forma de validar, experimentalmente, suas proposições teóricas, criando condições necessárias, tais como: tempo disponível e ambiente propício para sua aplicação, composição de uma equipe de avaliadores qualificados, composição de uma equipe de usuários representativos para simular as interações necessárias com o ambiente, entre outras.
- Análise dos dados obtidos na experimentação, apresentando estatisticamente os resultados.

Com o objetivo de dar continuidade à linha de pesquisa, pretende-se analisar questões relativas à concepção e avaliação de interfaces direcionadas para ambientes virtuais interativos educacionais.

ANEXO 1 - GUIA DE RECOMENDAÇÕES ERGONÔMICAS PARA SOFTWARE EDUCATIVO

ÍNDICE

1. Condução

Definição: A *condução* refere-se aos meios disponíveis para aconselhar, orientar, informar e conduzir o usuário na interação com o computador (mensagens, alarmes, rótulos, etc.).
Quatro sub-critérios participam da condução: a *presteza*, o *agrupamento/distinção entre itens*, o *feedback imediato* e a *legibilidade*.

Justificativa: Uma boa *condução* facilita o aprendizado e a utilização do sistema permitindo que o usuário:

- Saiba, em qualquer momento, o ponto em que se encontra numa sequência de interações ou na execução de uma tarefa;
- Conheça as ações permitidas bem como suas consequências;
- Obtenha informações suplementares (eventualmente a seu pedido);
- Tenha melhora no desempenho e na diminuição do número de erros;
- Atinja com maior rapidez e eficácia a aquisição do conhecimento proposto.

Sub-critérios – Condução

1.1. Presteza

Definição: Este critério engloba os meios utilizados para levar o usuário a realizar determinadas ações como, por exemplo, a entrada de dados, os mecanismos ou meios que permitam ao usuário conhecer as alternativas, em termos de ações, do estado ou contexto nos quais ele se encontra bem como a apresentação das ferramentas de ajuda e seu modo de acesso.
Do ponto de vista educacional, a presteza refere-se à capacidade do *software* em orientar o usuário na obtenção de um determinado objetivo pedagógico, fornecendo-lhe ferramentas e meios para o atingir.

Justificativa(s): Uma boa *presteza* guia o usuário e poupa-lhe, por exemplo, o aprendizado de uma série de comandos desnecessários para uma referida ação. Ela permite, também, que se saiba exatamente em que modo ou em que estado ele está, em que ponto se encontra no diálogo e o que fez para se encontrar nessa situação. Uma boa *presteza* facilita a navegação no aplicativo e diminui a ocorrência de erros, consequentemente facilitando a situação de ensino/aprendizagem.

Recomendações

1. O software deve disponibilizar um glossário para auxiliar o usuário na compreensão de termos técnicos (Figura 01);

2. O software deve permitir a impressão das informações desejadas.

3. O software deve utilizar recursos do tipo hipertexto com links apropriados que facilitem a compreensão do conteúdo (Figura 02);

4. A apresentação do tipo hipertexto deve ter boa condução, de forma que o usuário possa se localizar bem enquanto navega no programa (Figuras 03, 04 e 05);

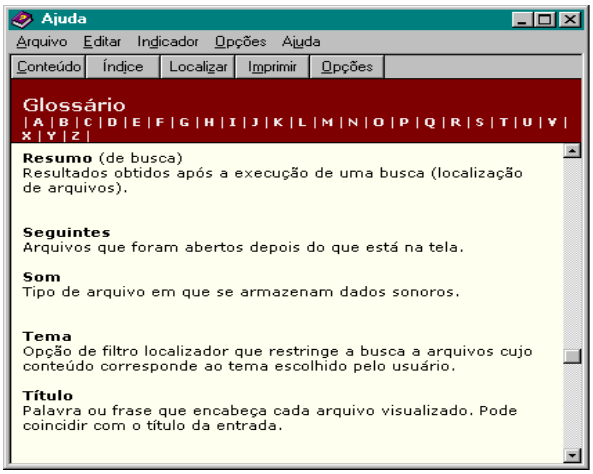


Fig. 01 – Glossário.

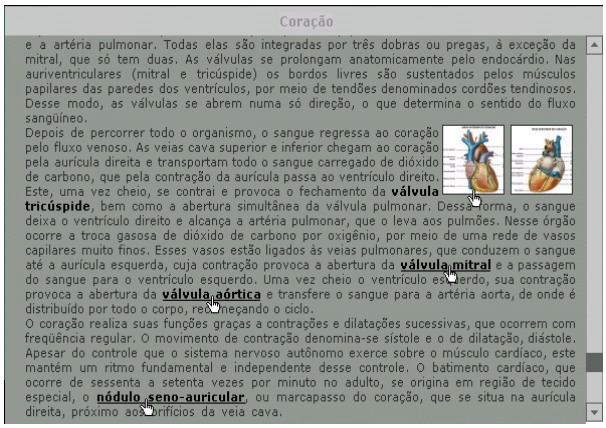


Fig. 02 – Links para navegação.

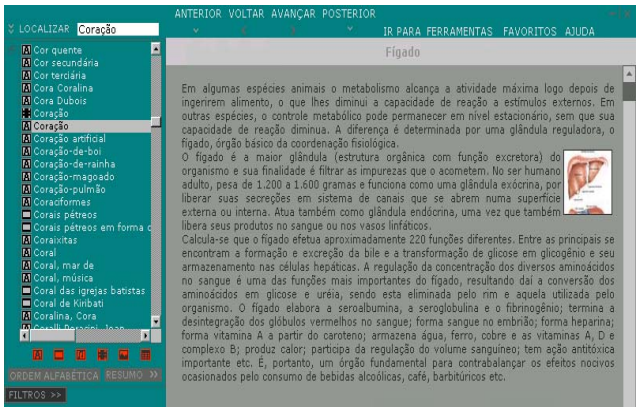


Fig. 03 – Ambiente de localização e navegação.

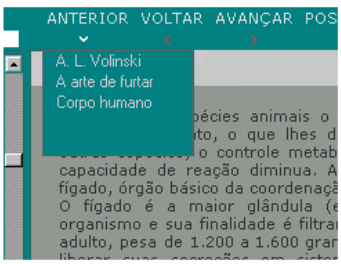


Fig. 04 – Links anteriores.

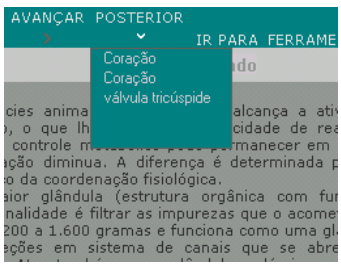


Fig. 05 – Links posteriores.

5. O software deve informar ao usuário o estado da ação, de forma que possa acompanhar a evolução do processamento da informação, utilizando recursos como: mensagens, ampulhetas, relógios e/ou barra de progressão (Figuras 06 e 07);

Fig. 06 – Ampulheta e Mensagem.

6. O usuário deve encontrar disponível na tela as informações necessárias para executar suas ações e efetuar as operações requeridas pelo software.

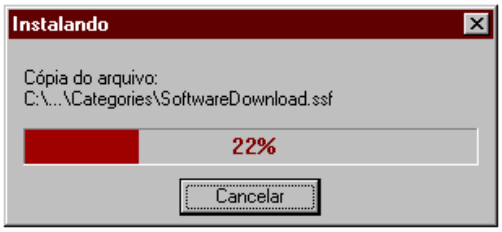


Fig. 07 – Barra de Progressão.

- COMENTÁRIO 1: Assegurar que quaisquer dados de que o usuário necessite para qualquer transação estejam disponíveis para apresentação. A apresentação dos dados se refere às saídas de dados do computador para os usuários e a assimilação da informação de tais saídas.

- COMENTÁRIO 2: O projetista da interface deve aplicar, com o usuário, alguns métodos de análise da tarefa (ex., diagrama de seqüência operacional) para determinar as necessidades detalhadas de informação dos usuários para qualquer transação.

- COMENTÁRIO 3: Se as necessidades de dados excederem as habilidades dos usuários de assimilar informações da apresentação, será necessário desdobrar as tarefas em passos menores. Podem ser necessários testes de protótipos, para determinar a apresentação ótima de dados, em tarefas críticas.

- COMENTÁRIO 4: O usuário não deve ter que lembrar de dados de uma apresentação para outra.

REFERÊNCIA: Smith & Mosier [1986]

7. O software deve apresentar títulos nas caixas de diálogo, formulários, campos de entrada de dados, janelas, etc., e estes devem estar no alto, centralizados ou alinhados à esquerda (Figuras 08 e 09).

Fig. 08 – Título alinhado à esquerda.

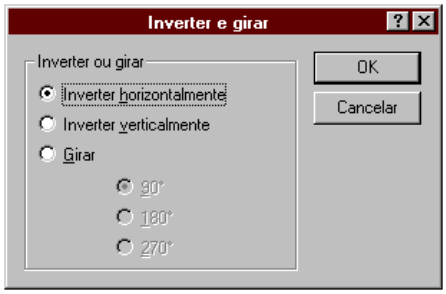


Fig. 09 – Título centralizado.

8. As opções que comandam a apresentação da abertura de outras opções de diálogo devem apresentar em seus rótulos sinais indicadores da continuidade do diálogo. (tais como "..." ou o sinal ">").

- COMENTÁRIO 1: As reticências indicam, para o usuário, a necessidade de mais informações para que o sistema execute alguma ação efetiva, podendo, assim, explorar as funcionalidades com total segurança (Figura 10).

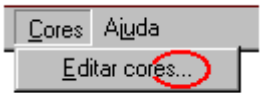


Fig. 10 – Reticências.

- COMENTÁRIO 2: O triângulo (seta) indica, para o usuário, quais das opções que acionam outros painéis de menu (Figura 11).

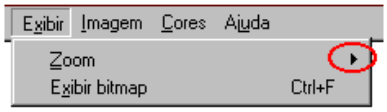


Fig. 11 – Triângulo (seta).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

9. No caso em que são apresentadas tabelas ao longo do software, devem possuir cabeçalho para linhas e colunas, devendo estes apresentar-se de maneira distinguíveis dos restantes dados (quanto a cor, fonte ou tipo de letra).

- COMENTÁRIO 1: Há muitos modos de distinguir diferentes tipos de material rotulado, incluindo diferenças consistentes no formato/localização da apresentação assim como fontes e marcadores (Figura 12).

REFERÊNCIA: Smith & Mosier [1986]

	Jan	Fev	Mar	Total
Leste	7	7	5	19
Oeste	6	4	7	17
Sul	8	7	9	24
Total	21	18	21	60

Fig. 12 – Cabeçalhos com formatação distinta.

1.2. Feedback imediato

Definição: Feedback Imediato diz respeito às respostas do sistema às ações do usuário. Estas ações podem ir do simples pressionar de uma tecla até a uma seleção dentro de uma lista de comandos. Em qualquer caso, as respostas do computador devem ser fornecidas, de forma rápida, no momento apropriado e de forma consistente com cada tipo de transação. Devem ser fornecidas respostas rápidas com informação sobre a transação solicitada e o seu resultado.
A emissão de *feedback* mediante interações inadequadas em *software* educacional é fundamental para informar adequadamente o usuário quando este executa um erro ou encontra uma dificuldade específica, conduzindo-o à sua resolução. O *feedback* deve ser positivo e capaz de reforçar as respostas corretas dos usuários. A qualidade das mensagens de *feedback* imediato será tratada no critério qualidade das mensagens de erro.

Justificativa(s): A qualidade e rapidez do *feedback* são dois fatores importantes para o estabelecimento da satisfação e confiança do usuário, assim como para o entendimento do diálogo. Estes fatores possibilitam um melhor entendimento do funcionamento do sistema.
A ausência de *feedback* ou sua demora pode ser incômoda para o usuário. Este pode, por exemplo, suspeitar de uma falha no sistema e realizar ações prejudiciais aos processos em andamento.

Os *feedbacks* sonoros devem ser utilizados com o cuidado para não provocarem sensações desagradáveis ou de embaraço aos usuários. Por exemplo, se cada vez que o aluno erra a resolução de um exercício o sistema emitir o som de uma campainha, este som poderá provocar a esse aluno uma carga emocional negativa, irritabilidade e dificultar a situação de ensino/aprendizagem.

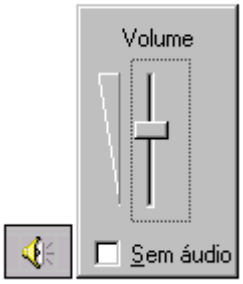
Recomendações

1. O sistema deve emitir algum *feedback* sonoro mediante respostas inadequadas do usuário na resolução de exercícios.

   - COMENTÁRIO 1: Na ocorrência de possíveis erros, durante a entrada de dados, deve ser emitido um sinal sonoro com o objetivo de chamar a atenção do usuário para a tela.

   - COMENTÁRIO 2: O *feedback* sonoro deve ser amplamente utilizado em software do tipo simulação, podendo representar sons de fenômenos físicos como colisão, quebra, explosão, etc. Músicas e narrativas também podem ser usadas em simulações.

   - COMENTÁRIO 3: Observar que em um ambiente de trabalho em grupo, os sinais sonoros podem distrair outras pessoas ou embaraçar o usuário, cujo erro está sendo assinalado. Nesse caso, permitir ao usuário desabilitar o sinal sonoro (Figuras 13 e 14).



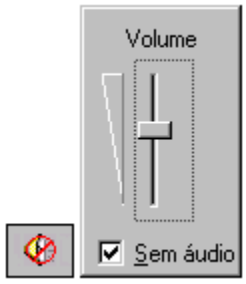


Fig. 13 – Som habilitado. Fig. 14 – Som desabilitado.

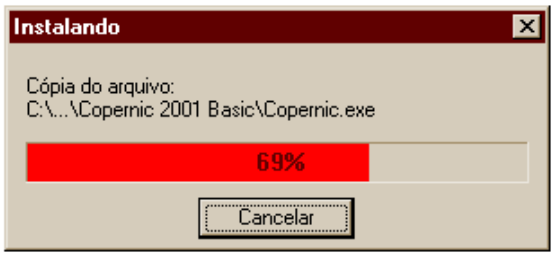


Fig. 15 – Progressão em ambiente gráfico.

2. O sistema deve fornecer informações sobre o progresso do processamento da informação.

   - EXEMPLO POSITIVO 1: Em um ambiente gráfico, um retângulo com indicador de 0 a 100% pode representar a progressão de uma tarefa; em um ambiente não gráfico, apenas a apresentação da porcentagem pode ser suficiente (Figuras 15 e 16).

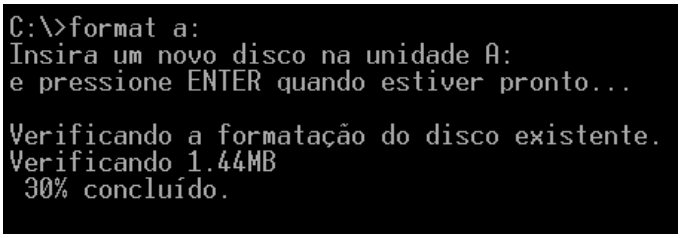


Fig. 16 – Progressão em ambiente não gráfico.

- COMENTÁRIO 1: Na maior parte dos casos, um contador regressivo é preferível a um contador progressivo. Por exemplo, em uma tarefa de cópia de arquivos, o sistema pode apresentar uma mensagem indicando a quantidade de arquivos restantes a serem copiados (Figura 17).

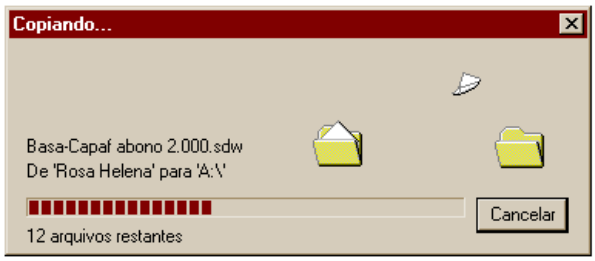


Fig. 17 – Contador regressivo para cópia de arquivos.

- EXEMPLO POSITIVO 2: Um contador pode indicar a quantidade atual de operações repetitivas em uma dada ação. Por exemplo, uma simulação de algoritmo genético apresentando o número de divisões de uma bactéria (Figura 18).

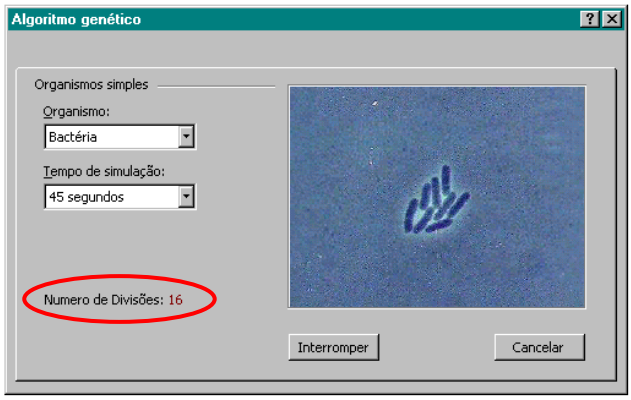


Fig. 18 – Indicador de número de divisões de uma bactéria.

- EXCEÇÃO 1: Quando o número de operações não for elevado, o sistema poderá listar cada operação separadamente no momento em que é executada. Por exemplo (Figura 19):

"Excluindo o arquivo "Sinoptic.doc"
"Excluindo o arquivo "Finanças.doc"
"Excluindo o arquivo "Medicina.doc"

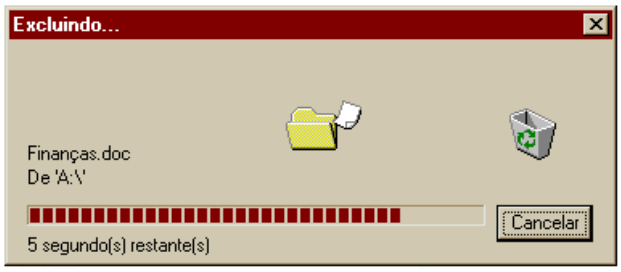


Fig. 19 – Excluindo o arquivo “Finaças.doc”.

- COMENTÁRIO 2: Os indicadores dinâmicos fornecem uma resposta imediata (dando sinais de que ação foi aceita) e contínua durante o tempo de espera do usuário. Eles informam sobre o avanço regular do trabalho, mantendo a atenção do usuário durante o tempo de espera.

REFERÊNCIAS:Bodart & Vanderdonckt [1993]

3. O sistema deve fornecer informações sobre o tempo total requerido ao processamento da informação quando este é demorado (Figura 20).

Fig. 20 – Indicativo de tempo.

4. O sistema deve fornecer "*feedback*" imediato de todas as entradas de dados dos usuários. (incluindo dados sigilosos, que neste caso devem produzir um *feedback* perceptível, como, por exemplo, o símbolo *).

    - COMENTÁRIO 1: Forneça feedback visual para todas as ações do usuário durante a entrada de dados. Apresente feedback para entradas pelo teclado, toque por toque, exceto para senhas e outras entradas sigilosas. Mesmo nesses casos, cada toque deve produzir um feedback perceptível (i.e., símbolos como *) (Figura 21).

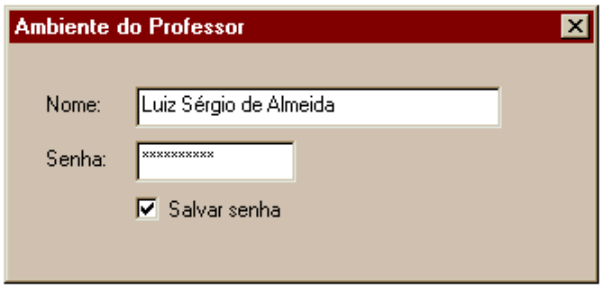


Fig. 21 – "*Feedback*" de entradas digitadas.

REFERÊNCIA: Smith & Mosier [1986]

5. O tempo de resposta do produto deve ser adequado à operação levando em consideração a complexidade, a abrangência e o volume dos dados manipulados.

6. Nas operações interativas o tempo de resposta deve ser adequado e homogêneo em todas as operações. (carregamento de telas, imagens, dados, arrasto do mouse, etc.).

    - COMENTÁRIO 1: Nos casos em que o sistema se mostra lento ou a operação em andamento será muito demorada, é importante fornecer um feedback imediato que informe o inicio da operação e o tempo de duração da execução.

- EXEMPLO 1: Quando o usuário necessitar copiar ou mover um conteúdo relativamente pequeno, o sistema deve permitir que a mensagem de processamento permaneça na tela tempo suficiente para que a mesma possa ser cancelada caso seja necessário (Figura 22).

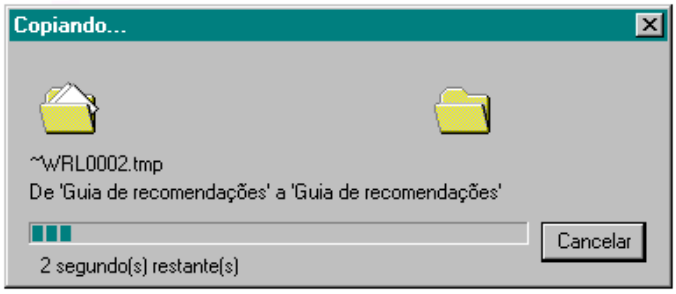


Fig. 22 – Copiando arquivo pequeno – 2 segundos restantes.

- EXEMPLO 2: O velocidade de rolagem, proporcionado pelo arraste do mouse sobre um texto ou imagem, deve ser proporcional e homogêneo, permitindo que o usuário visualize o local desejado.

7. Caso o usuário interrompa um processamento de dados, o sistema deve mostrar uma mensagem garantindo-lhe que o sistema voltou ao seu estado prévio (Figura 23).

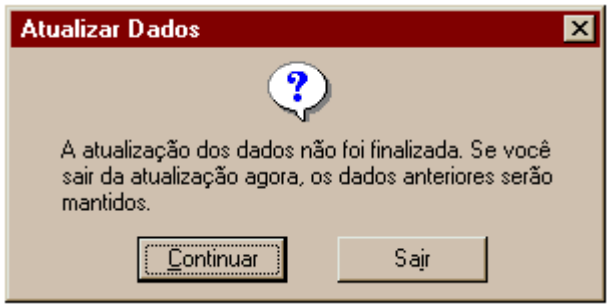


Fig. 23 – Interrupção com recuperação dos dados prévios.

8. Quando o processamento da informação for concluído, o sistema deve apresentar uma mensagem que informe sobre o sucesso ou fracasso da operação.

- COMENTÁRIO 1: O computador deve confirmar o final de uma transação de entrada de dados através de uma mensagem de confirmação, se a entrada de dados for bem sucedida ou, no caso contrário, com uma mensagem de erro (Figuras 24 e 25).

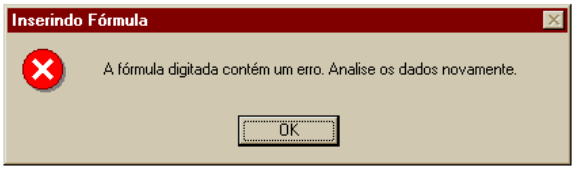


Fig. 24 – Fracasso da operação.

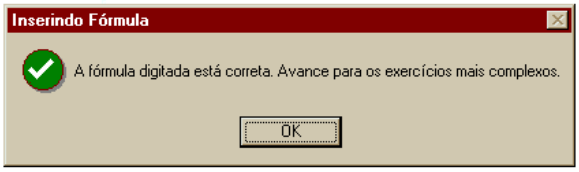


Fig. 25 – Sucesso da operação.

- EXCEÇÃO: De modo a acelerar o ritmo das transações em ações repetitivas de entrada de dados, a finalização bem sucedida de uma entrada pode gerar a apresentação do formulário inicial de entrada de dados, porém vazio.

- COMENTÁRIO 2: Para cada transação de entrada de dados é melhor deixar os dados que entraram na tela até o usuário explicitamente decidir limpá-los. Uma entrada de dados bem sucedida não deve ser confirmada através da retirada do dado da tela, exceto no caso de entrada de dados repetitivos.

9. Durante a tarefa de impressão, o sistema deve fornecer informações sobre o estado desta, através da seleção e utilização de bons gerenciadores (Figura 26).

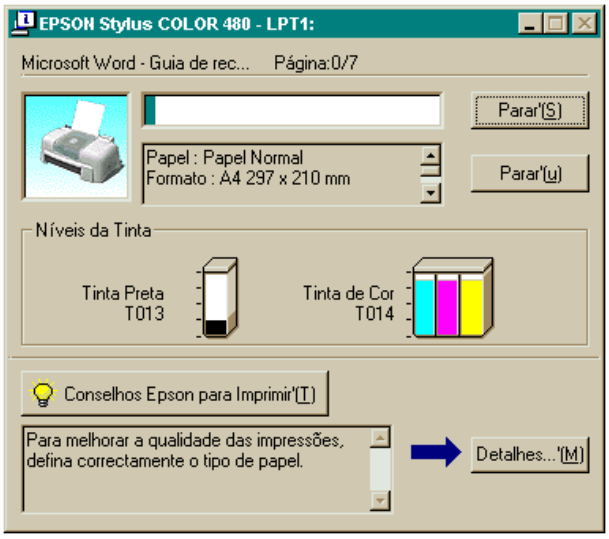


Fig. 26 – Estado de impressão.

- COMENTÁRIO 1: O computador deve reconhecer as solicitações de impressão imediatamente, oferecendo mensagens subseqüentes indicando quando a impressão foi completada, se a impressora estiver longe da estação de trabalho do usuário.

- COMENTÁRIO 2: Se existirem documentos aguardando impressão, o usuário deve ter uma estimativa de quando um documento particular deverá ser impresso.

- COMENTÁRIO 3: Se o usuário é responsável pela operação de uma impressora local, o sistema deve apresentar mensagens para alertá-lo sobre as disfunções potenciais, por exemplo, se o suprimento de papel terminou, ou se o papel não está corretamente colocado, etc.

REFERÊNCIA: Smith & Mosier [1986]

10. O sistema deve fornecer "*feedback*" sobre as mudanças de atributos dos objetos de interação.

    - COMENTÁRIO 1: Ao selecionar um botão, o ícone correspondente a esta opção deve apresentar mudança de estado, entre acionado e não acionado (Figura 27 e 28).

Fig. 27 – Botão não acionado.

Fig. 28 – Botão acionado.

1.3. Legibilidade

Definição: Legibilidade diz respeito às características lexicais das informações apresentadas na tela que possam dificultar ou facilitar a leitura desta informação (brilho do caráter, contraste letra/fundo, tamanho da fonte, espaçamento entre palavras, espaçamento entre linhas, espaçamento entre parágrafos, comprimento da linha, etc.). Por definição, o critério *Legibilidade* não abrange mensagens de erro ou de *feedbac*k.
No que se refere aos aspectos pedagógicos, o critério legibilidade avalia também se a qualidade da informação apresentada favorece a compreensão e assimilação dos conteúdos educacionais.

Justificativa(s): O desempenho melhora quando a apresentação da informação leva em conta as características cognitivas e perceptivas dos usuários. Uma boa legibilidade facilita a leitura da informação apresentada e contribui para a compreensão dos conteúdos e para alcançar os objetivos pedagógicos propostos.
O critério legibilidade avalia, por exemplo, situações referentes à apresentação gráfica da informação. Letras escuras em um fundo claro são mais fáceis de ler que letras claras contra fundo escuro; texto escrito com letras maiúsculas e minúsculas é lido mais rapidamente que texto escrito somente com maiúsculas.
Um *software* com boa legibilidade apresenta informações claras, é bem redigido e livre de equívocos conceituais. Utiliza uma linguagem apropriada e orientada para o seu público alvo específico, facilitando a compreensão e assimilação dos conteúdos pelas suas estruturas cognitivas.
Legibilidade não se aplica a mensagens de *feedback* ou de erro. Todos os aspectos relacionados com as dificuldades de leitura das mensagens, ou mais genericamente com a qualidade destas mensagens, quando se tratar de mensagens de *feedback* ou de erro, deveriam estar relacionados respectivamente com os critérios *Feedback Imediato* ou *Qualidade das Mensagens de Err*o.

Recomendações

1. Os conteúdos apresentados devem ser livres de equívocos conceituais (Figuras 29 e 30).

2. A redação das informações textuais deve estar correta, livre de erros gramaticais e de pontuação (Figuras 31 e 32).

3. O estilo literário do texto deve favorecer a compreensão dos conteúdos (Figuras 33 e 34).

   - COMENTÁRIO 1: Para se determinar o estilo literário adequado deve-se levar em conta as características do usuário.

   - COMENTÁRIO 2: Fatores como nível de instrução e área de atuação do aluno são fundamentais na concepção do estilo literário.

   - EXEMPLO 1: Alunos do último ano do curso de Direito apresentam nível literário diferenciado de alunos do primeiro ano do curso de Nutrição.

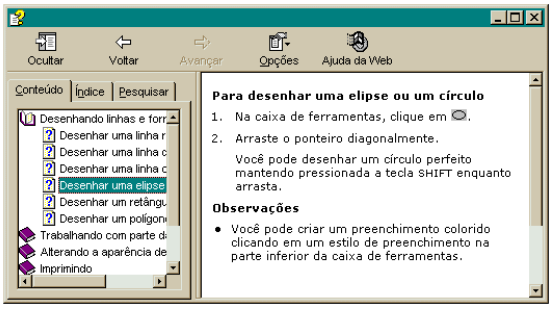


Fig. 29 – Procedimento equivocado.

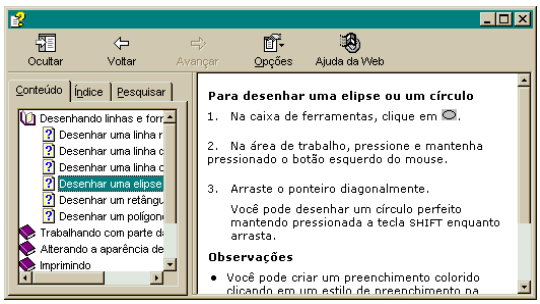


Fig. 30 – Procedimento correto.

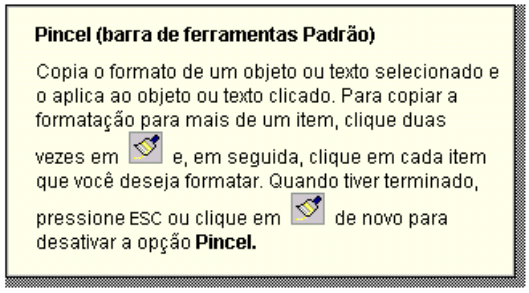


Fig. 31 – Texto e pontuação corretos.

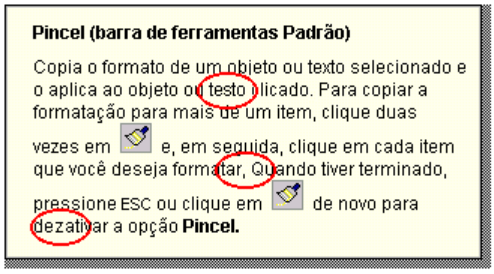


Fig. 32 – Texto e pontuação incorretos.

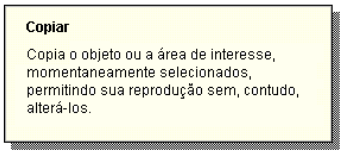


Fig. 33 – Estilo A.

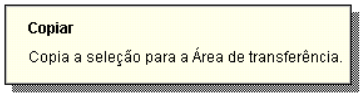


Fig. 34 – Estilo B.

4. A apresentação do texto, o tipo e tamanho das letras devem ser de fácil legibilidade (Figuras 35 e 36).

   - COMENTÁRIO 1: Os aspectos lexicais que podem influenciar a facilidade da leitura (isto é, tamanho da fonte, tipo da fonte, etc.) dos elementos ligados à Presteza, dizem respeito à Legibilidade.

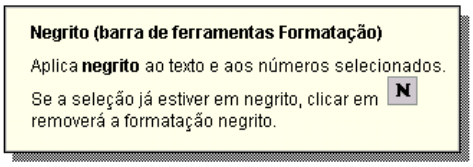


Fig. 35 – Texto de fácil legibilidade.

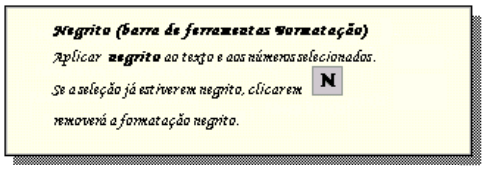


Fig. 36 – Texto de difícil legibilidade.

5. Deve-se evitar o uso exclusivo de maiúsculas nos textos apresentados.

   - COMENTÁRIO 1: O uso exclusivo de maiúsculas deve ser evitado, deve ser apresentado o texto convencionalmente num misto de caixa-alta e caixa-baixa. As maiúsculas devem ser utilizadas com precaução, sobretudo no corpo de um texto.

   - COMENTÁRIO 2: A leitura e a decifração de texto em maiúsculas é 30% mais lenta do que um texto em minúsculas, em virtude das formas distintas presentes nos traços verticais (por exemplo, o traço vertical do h) e as pernas das letras. Além disso, escrever tudo em maiúsculas pode produzir, no ato de leitura, a sobreposição parcial de duas linhas, tornando a leitura mais difícil.

   - EXCEÇÃO 1: A letra maiúscula deve ser usada quando a minúscula perder legibilidade, por exemplo, numa apresentação em terminal de vídeo onde não se disponha de fontes originais (true types) em minúsculas (Figura 37).

Fig. 37 – Perda de legibilidade nas fontes minúsculas.

- EXCEÇÃO 2: Um rótulo ou título pode ser mostrado em letras maiúsculas para atrair a atenção dos usuários (Figura 38).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

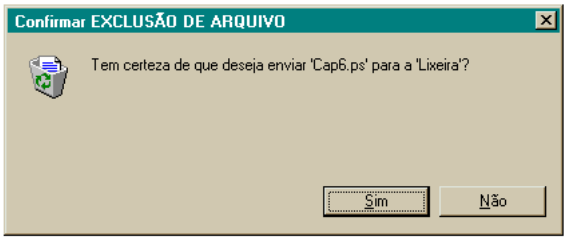


Fig. 38 – Título em letra maiúscula.

6. O uso de recursos de estilo como sublinhado, negrito, itálico, deve ser feito de maneira ponderada e não atrapalhar a legibilidade do texto.

- EXEMPLO POSITIVO 1: Utilizar o negrito para decompor uma tela em regiões funcionais com título para guiar o olhar sobre a tela (Figura 39).

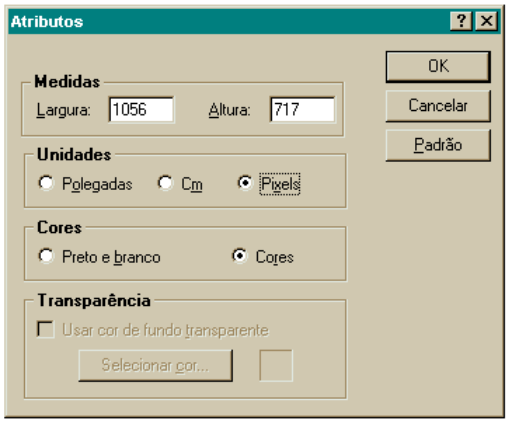


Fig. 39 – Título de regiões funcionais.

- COMENTÁRIO 1: Muito texto sublinhado pode atrapalhar a legibilidade. Usar o sublinhado para destacar palavras ou frases relevantes (Figuras 40 e 41).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

Fig. 40 – Destaque de palavras relevantes.

Fig. 41 – Excesso de sublinhado.

7. Os parágrafos de texto devem ser separados por uma linha em branco, pelo menos, e possuir margens bem definidas (Figura 42).

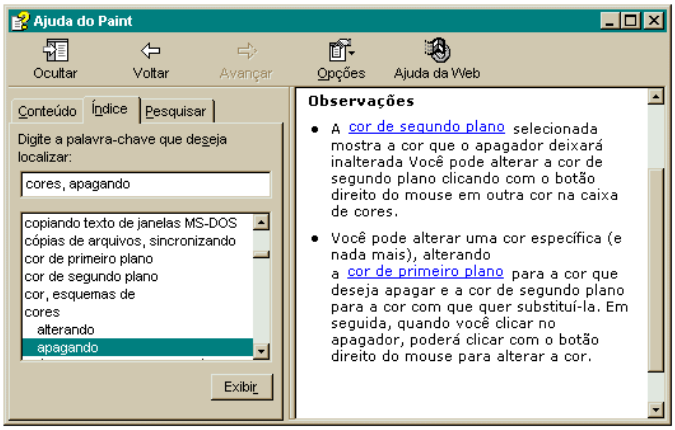


Fig. 42 – Linha em branco entre os parágrafos.

8. Deve ser evitado o uso de abreviaturas nos menus, opções de menu, título das caixas de diálogo, e mostradores de dados.

   - COMENTÁRIO: Não use abreviaturas, a não ser que ela seja significantemente menor do que a palavra inteira e tão significativa para o usuário como a palavra completa. Se com a abreviação você economizar somente um ou dois caracteres use a palavra inteira (Figura 43).

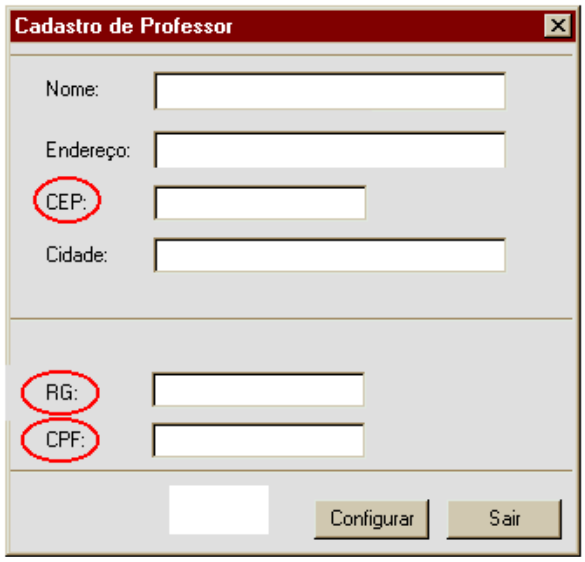


Fig. 43 - abreviatura significantemente menor do que a palavra inteira

9. O uso de cores deve favorecer a legibilidade do programa (figuras 44 e 45).

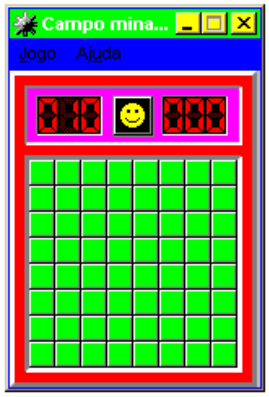


Fig. 44 – Legibilidade prejudicada.

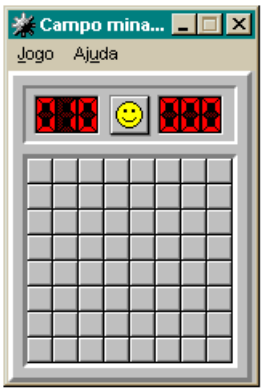


Fig. 45 – Legibilidade favorecida.

10. A cor do fundo em relação à cor da letra deve permitir uma boa leitura (Figuras 46 e 47).

11. O texto apresentado nas caixas de opções de menu deve apresentar boa legibilidade, ou seja, devem estar adequadamente posicionados e separados das bordas neste tipo de caixa (Figuras 48 e 49).

12. Os ícones devem ser legíveis e representar funções (Figuras 50 e 51).

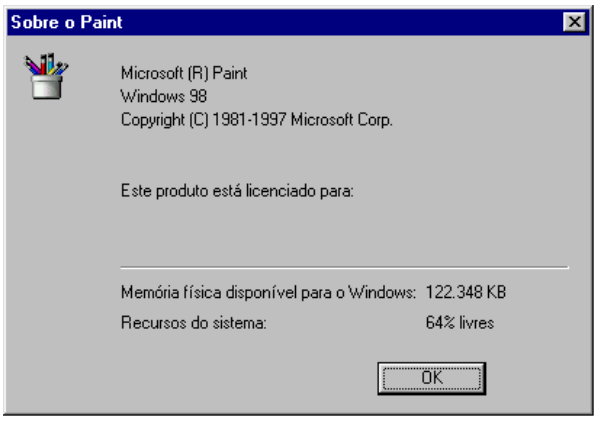


Fig. 46 – Boa leitura.

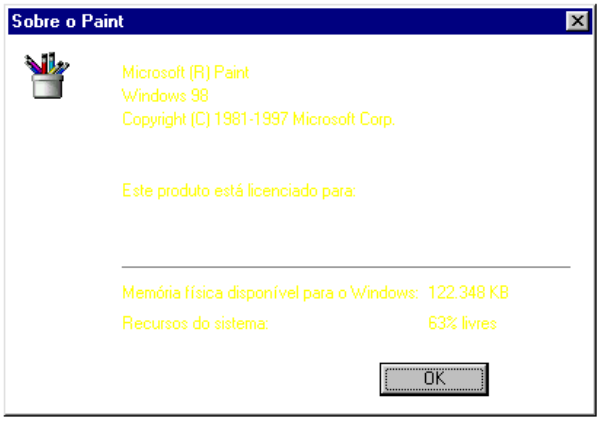


Fig. 47 – Má leitura.

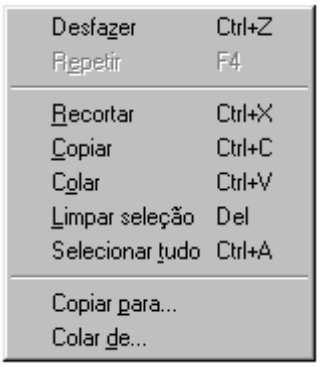


Fig. 48 – Forma correta.

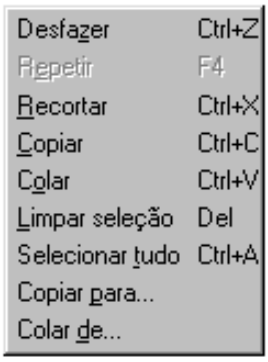


Fig. 49 – Forma incorreta.

Fig. 50 – Exemplo positivo.

Fig. 51 – Exemplo negativo.

- COMENTÁRIO 1: Ícones bem projetados são auto-explicativos, no mínimo, no contexto no qual são usados. Existem diversas dicas para o projeto de ícones claros:
-

1. Usar representações concretas de objetos e ações atualmente em uso;

2. Enfatizar os elementos gráficos que distinguem esse objeto de outros;

3. Simplificar a representação, eliminando ou dissimulando os elementos que não contribuem para a identificação do objeto;

4. Seguir as convenções de projeto na construção de todos os ícones do sistema, no que se refere ao uso, aparência e localização de rótulos textuais, bordas e outros elementos definidos no projeto.

REFERÊNCIA:Brown [1988]

13. Deve-se evitar a utilização de somente letras maiúsculas nos títulos de caixas de diálogo.

- COMENTÁRIO 1: Para facilitar a leitura, os títulos das caixas de diálogos devem sempre iniciar com letra maiúscula. O restante do rótulo pode conter somente letras minúsculas (Figuras 52, 53, 54 e 55).

Fig. 52 – Exemplo positivo.

Fig. 53 – Exemplo negativo.

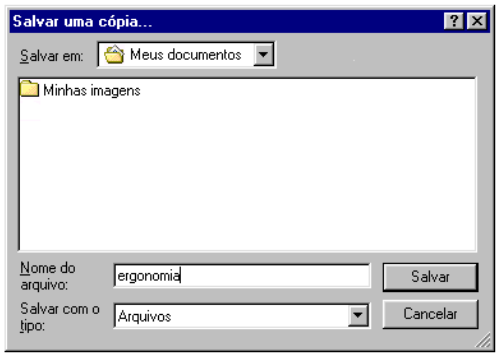


Fig. 54 – Exemplo positivo.

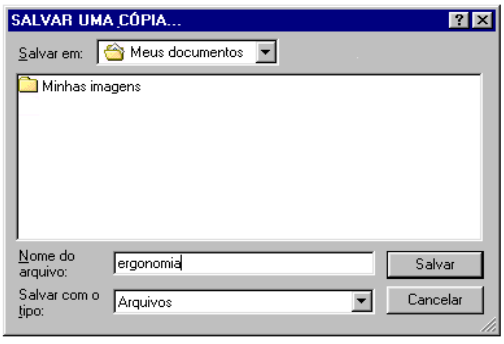


Fig. 55 – Exemplo negativo.

14. Os objetos de interação (botões, campos de edição, etc.) disponíveis nas caixas de diálogo devem estar alinhados vertical e horizontalmente (Figuras 56, 57 e 58).

- COMENTÁRIO 1: Sempre que possível os grupos de controles, apresentações e outros objetos compostos que formam as caixas de diálogo devem estar alinhados vertical e/ou horizontalmente, à esquerda e/ou direta.
-
- COMENTÁRIO 2: Os blocos visuais ficam mais destacados quando têm a mesma dimensão, mesmo se contêm informações diferentes ou objetos de tamanho diferente.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

15. As áreas livres devem ser usadas para separar grupos lógicos, não devendo estar todos de um só lado da tela, caixa ou janela.

- COMENTÁRIO 1: Deve-se usar áreas livres para estruturar uma apresentação. Dados apresentados muito próximos são difíceis de localizar e de ler. Deve-se usar espaços brancos para separar grupos de dados. Já os itens de dados relacionados dentro de um grupo devem estar dispostos suficientemente próximos, minimizando o tempo do movimento dos olhos para encontrar esses dados (Figura 59).

Fig. 56 – Exemplo positivo.

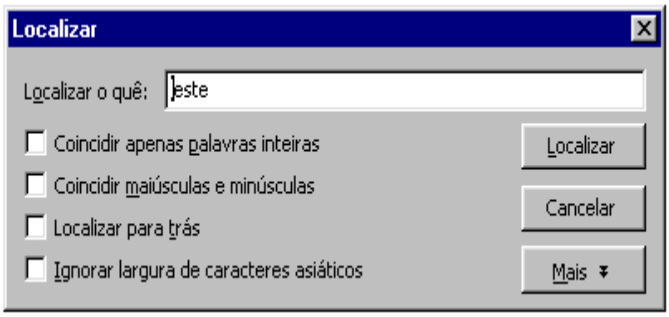


Fig. 57 – Exemplo

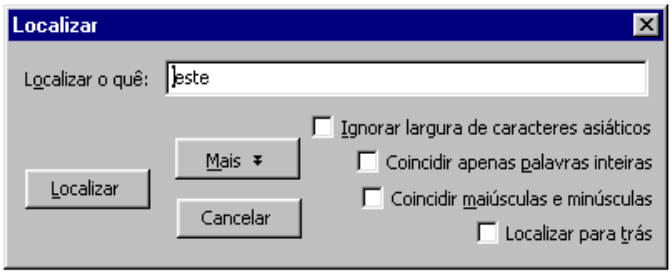


Fig. 58 – Exemplo negativo.

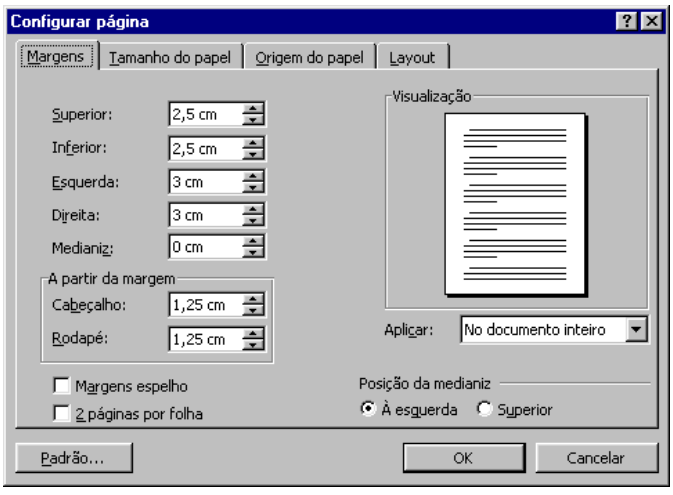


Fig. 59 – Exemplo positivo.

- COMENTÁRIO 2: Deve-se evitar grande quantidade de informação e confusão nas apresentações. A confusão é causada pelo pouco espaçamento, pela falta de ordem e apresentação desnecessária de dados. Deve-se apresentar os dados usando espaçamento ou agrupamento, ou colunas para produzir uma apresentação legível e ordenada. Não apresentar muitos dados numa única tela. Distribuir as áreas não usadas para separar grupos lógicos, em vez de ter todas as áreas não usadas em um só lado (Figura 60).

- COMENTÁRIO 3: Existe uma densidade ótima para cada apresentação, e densidades altas ou baixas podem degradar o desempenho do usuário. Apresentações usando 15% de posições disponíveis de caracteres podem ser ótimas. Aumentar a densidade da apresentação de 15% para 25% pode ter um efeito colateral mínimo no desempenho do usuário; mas densidades acima de 25% podem degradar a performance.

REFERÊNCIA: Smith & Mosier [1986], Brown [1988]

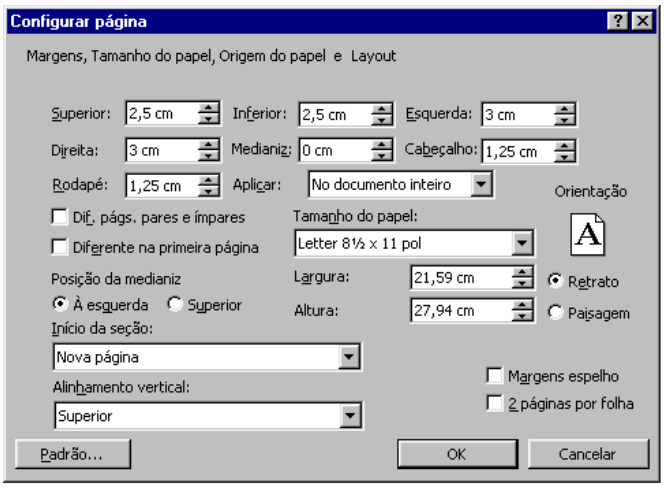


Fig. 60 – Exemplo negativo.

16. As informações codificadas através das cores devem apresentar uma codificação adicional redundante.

    - COMENTÁRIO 1: Deve-se compor a codificação de cores de forma redundante com outras características de apresentação tais como a simbologia. Deve-se evitar a codificação apenas pela cor (Figuras 61 e 62).

    - COMENTÁRIO 2: Os dados apresentados devem oferecer informação necessária mesmo quando visualizados em terminal monocromático, ou cópia impressa, ou quando visualizado por usuários com problemas visuais em relação a cores.

REFERÊNCIA: Smith & Mosier [1986]

17. Dados numéricos que se alterem rapidamente devem ser apresentados analogicamente.

    - COMENTÁRIO 1: Em uma exibição digital deve-se tomar cuidado para que a apresentação dos números não ocorra rápido demais, impedindo sua leitura por parte dos usuários.

    - COMENTÁRIO 2: Deve-se usar uma exibição analógica para mostrar um dado numérico que sofrem alteração de forma dinâmica. Este procedimento é mais conveniente que uma exibição digital.

Fig. 61 – Exemplo positivo - Links visitados sublinhados.

Fig. 62 – Exemplo negativo – Todos os Links sublinhados.

- COMENTÁRIO 3: Podem ser utilizados mostradores analógicos para apresentar valores numéricos. Sua oscilação deve manter-se dentro de um intervalo regular e bem definido.

- EXEMPLOS POSITIVOS: Um contagiros ou um velocímetro de um carro, um altímetro ou um horizonte artificial para um avião (Figuras 63 e 64).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

Fig. 63 – Velocímetro.

Fig. 64 – Altímetro.

1.4. Agrupamento e Distinção de Itens

*Definiçã*o: O critério *Agrupamento/Distinção de Itens* diz respeito à organização visual dos itens de informação de alguma maneira relacionados entre si. Este critério leva em conta a topologia (localização) e algumas características gráficas (formato) para indicar as relações entre os vários itens mostrados, para indicar se eles pertencem ou não a uma dada classe, ou ainda para indicar diferenças entre classes. Este critério também diz respeito à organização dos itens de uma classe. O critério *agrupamento/distinção de itens* está subdividido em dois critérios: *agrupamento/distinção por localização* e *agrupamento/distinção por formato*.

Justificativa(s): A compreensão de uma tela pelo usuário depende, entre outras coisas, da ordenação, do posicionamento e da distinção dos objetos (imagens, textos, comandos, etc.) que são apresentados.
Os usuários poderão detectar os diferentes itens ou grupos de itens e aprender suas relações mais facilmente se, por um lado, eles forem apresentados de uma maneira organizada (i.e., por ordem alfabética, segundo a frequência de uso, etc.); por outro lado, se os itens ou grupos de itens são apresentados em formatos ou codificados de maneira a indicar as suas semelhanças ou diferenças. Além disso, a aprendizagem e o estabelecimento de itens ou de grupos de itens será melhorado. O *Agrupamento/distinção de itens* leva a uma melhor *Condução*.

1.4.1. Agrupamento e Distinção de Itens por Localização

Definição: O critério de *Agrupamento/Distinção por Localização* diz respeito ao posicionamento relativo dos itens, estabelecido para indicar se eles pertencem ou não a uma dada classe ou, ainda, para indicar diferenças entre classes. Este critério também diz respeito ao posicionamento relativo dos itens dentro de uma classe.
Alguns *softwares educacionais apresentam* uma distinção clara entre os módulos de informação teórica e os módulos práticos e de resolução de exercícios. Já outros tipos de *software* não utilizam esta divisão de modo tão explícito e testam os conhecimentos adquiridos à medida que o aluno vai evoluindo na sequência de apresentação do programa. Este critério permite avaliar se a integração entre a teoria e a prática no *software* educacional é feita de maneira eficaz.

Justificativa(s): A compreensão de uma tela pelo usuário depende, entre outras coisas, da ordenação dos objetos (imagens, textos, comandos, etc.) que são apresentados. Os usuários irão detectar os diferentes itens mais facilmente se eles forem apresentados de uma forma organizada (i.e., por ordem alfabética, segundo a frequência de uso, etc.). Além disso, a aprendizagem e o restabelecimento de itens será melhorado. O *Agrupamento/distinção por localização* leva a uma melhor *Condução*.

Recomendações

1. Deve existir uma distinção clara na localização da informação dos módulos teórico e prático, e esta divisão deve ser balanceada (Figura 65).

2. No caso em que os conceitos teóricos são separados dos exercícios práticos, deve existir facilidade para o usuário navegar entre a parte prática e a teórica do software (Figuras 66 e 67).

Fig. 65 – As elipses representam os módulos teórico e prático, respectivamente.

Fig. 66 – Módulo prático.

Fig. 67 – Módulo teórico.

3. As informações devem possuir boa organização entre os itens e devem ser divididas em capítulos, unidades ou seções (Figuras 68 e 69).

   - COMENTÁRIO 1: Caso o usuário necessite analisar conjuntos de dados para discernir similaridades, diferenças e relações, a estrutura da apresentação deve ser formatada de modo a permitir que os dados sejam consistentemente agrupados.

   - COMENTÁRIO 2: Caso a meta seja ordenar dados em grupos lógicos, os critérios devem seguir a seguinte ordenação:
     1. Seqüencial;
     2. Funcional;
     3. Importância dos dados;
     4. Freqüência de uso.

   - OBSERVAÇÃO: Quando não existir lógica apropriada para o agrupamento de dados por seqüência, função ou importância, deve-se adotar algum outro princípio, tal como agrupamento cronológico ou alfabético.

REFERÊNCIA: Smith & Mosier [1986]

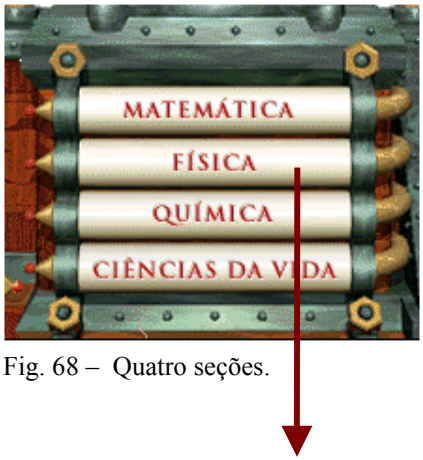


Fig. 68 – Quatro seções.

Fig. 69 – Seção de Física.

4. A informação deve ser apresentada em tópicos organizados por funções.

    - COMENTÁRIO 1: Para agrupar dados por seções, nas quais, o conjunto de dados está associado a funções específicas, deve-se considerar cada grupo de dados, de modo a ilustrar essas relações funcionais (Figuras 70 e 71).

REFERÊNCIA: Smith & Mosier [1986]

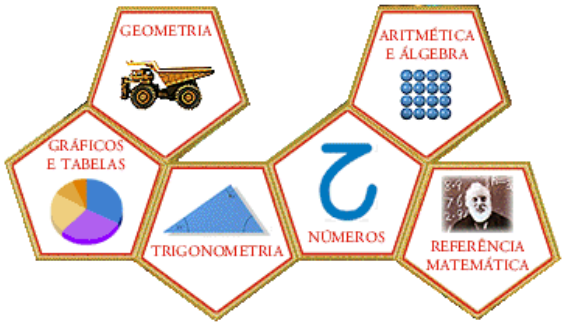


Fig. 70 – Seção de Matemática.

5. A informação deve ser apresentada em tópicos organizados por objetivos do usuário.

    - COMENTÁRIO 1: Quando houver a necessidade de apresentar dados agrupados por seqüência de uso de acordo com uma ordenação espacial ou temporal, deve-se considerar o agrupamento desses dados pela seqüência de uso, para preservar essa ordem (Figura 72).

Fig. 71 – Seção de Química.

Fig. 72 – Módulo prático - Questionário.

- OBSERVAÇÃO: Quando dados apresentados forem particularmente importantes, ou seja, oferecerem informação significativa e/ou necessitarem de resposta imediata do usuário, os mesmos devem ser agrupados por importância, definindo sua localização no topo da apresentação (Figuras 73a e 73b).

REFERÊNCIA: Smith & Mosier [1986]

6. Os itens de menus devem estar organizados (agrupados) hierarquicamente segundo uma ordem lógica.

- COMENTÁRIO 1: As opções devem ser agrupadas dentro de um menu para que reflitam as expectativas do usuário e facilitem a pesquisa das opções (Figura 74).

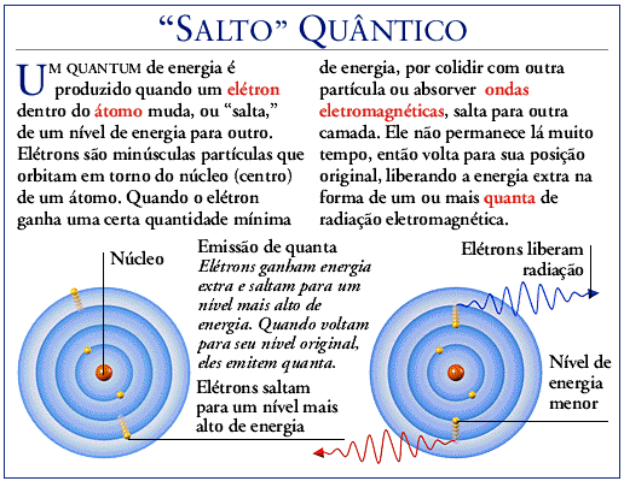


Fig. 73a – Módulo teórico – "Salto Quântico".

Fig. 73b – Módulo teórico – "Gelo e Vapor".

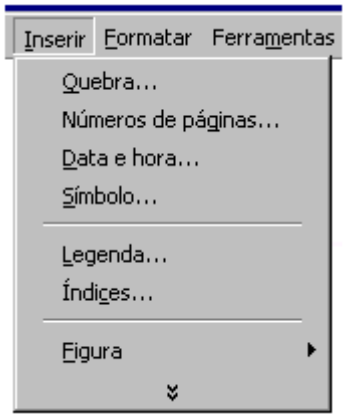


Fig. 74 – Menu Inserir e suas opções.

- COMENTÁRIO 2: Se o menu possui um grande número de opções (mais do que 8) e essas opções podem ser agrupadas de maneira lógica, as opções devem ser agrupadas por função, ou por outra categoria lógica significativa ao usuário (Figuras 75a e 75b).

- COMENTÁRIO 3: Ordem da seqüência de utilização: Se a seqüência de utilização das opções é conhecida, elasu devem ser classificadas segundo essa ordem (Figura 76).

- OBS. 1: Se 8 ou mais opções são dispostas arbitrariamente em um painel de menu, elas devem estar dispostas em grupos igualmente repartidos, utilizando a equação: g = raiz quadrada de n (onde g = número de grupos e n = número de opções no painel).

- OBS. 2: Ordem de Importância: Se as opções particulares são muito importantes, devem ser dispostas no início do grupo. No caso de ser importante impedir a execução acidental de uma opção, a recomendação acima pode ser abandonada.

- OBS. 3: Ordem convencional ou existente: Se existe uma ordem convencional e/ou outra que seja muito utilizada pelos usuários habituais, as opções devem estar dispostas segundo essas ordens.

REFERÊNCIA: ISO 9241 Parte 14 [1995]

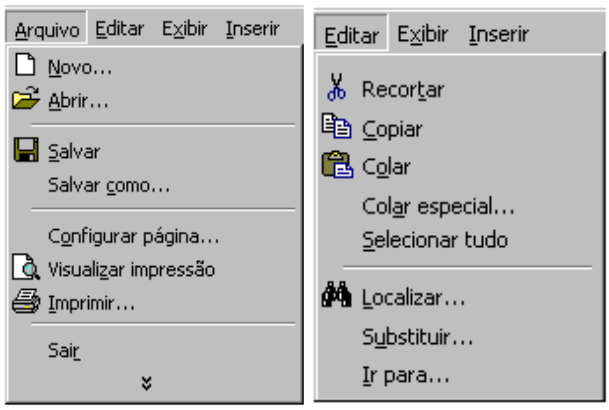


Fig. 75a – Menus Arquivo e Editar – Agrupados por função.

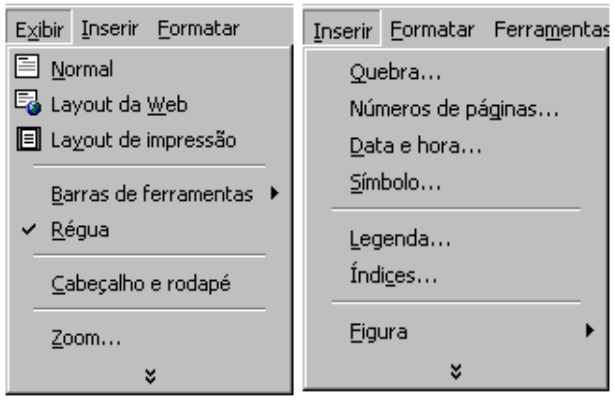


Fig. 75b – Menus Exibir e Inserir – Agrupados por função.

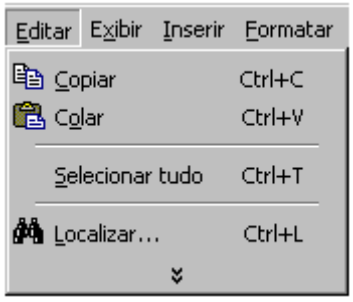


Fig. 76 – Seqüência de utilização conhecida – Copiar antes de Colar.

7. As opções de menu devem estar agrupadas de forma lógica, de forma a facilitar a interação com o usuário.

   - COMENTÁRIO 1: É conveniente que as estruturas de menus reflitam a expectativa do usuário, permitindo a localização e seleção imediata das opções de menu correspondentes à tarefa ou função a ser executada, auxiliando a progressão de seu trabalho (Figura 77).

   - COMENTÁRIO 2: Se as opções podem ser dispostas em grupos convencionais ou naturais, conhecidos pelos usuários, essas opções devem ser organizadas em níveis e menus coerentes com essa disposição (Figura 78).

   - OBS. 1: Categorias lógicas: As opções que não apresentam agrupamentos ou estruturas padronizadas devem ser agrupadas ou classificadas de modo não ambíguo e facilmente memorizável pelos usuários.

   - OBS. 2: Agrupamento arbitrário: Se as opções não podem ser agrupadas em categorias sem ambigüidades ou evidentes para os usuários (geralmente porque os usuários não sabem ao certo sobre a maneira como a opção desejada será descrita), é conveniente que as opções sejam dispostas de forma coerente (por exemplo, em ordem alfabética, em ordem numérica), em grupos de quatro a oito opções por nível. A divisão das opções

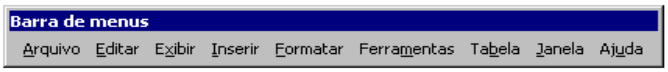


Fig. 77 – Opções de menu correspondentes à tarefa ou funcão a ser executada.

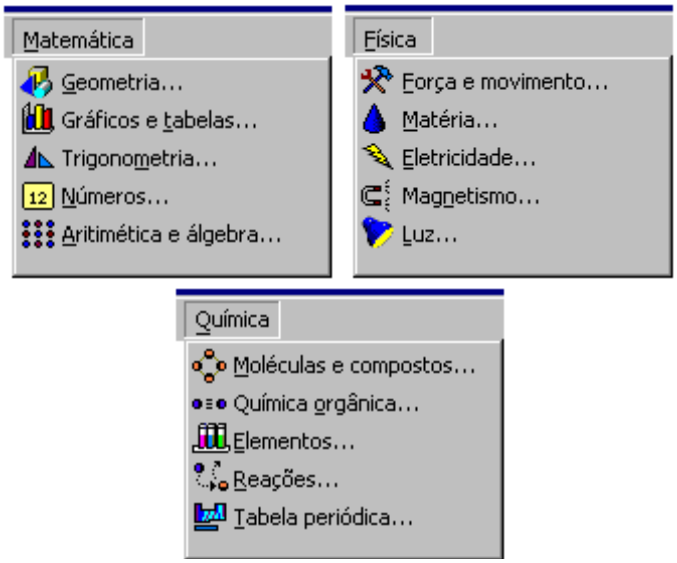


Fig. 78 – Níveis e menus coerentes conhecidos pelos usuários.

em pequenos grupos pode facilitar as estratégias de busca, quando a comparação das diferentes opções toma muito tempo (por exemplo, quando as opções são muito longas ou quando o usuário não está seguro da maneira como a opção desejada será descrita) (Figura 79).

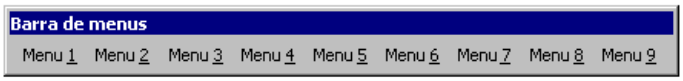


Fig. 79 – Opções de menu arbitrárias sem nenhuma indicação de tarefa ou função.

- OBS. 3: Considerações a respeito do tempo de busca: Para minimizar o tempo de busca, deve-se colocar a maior quantidade possível de opções e de níveis em um único quadro, mostrado na tela. A quantidade de opções em um menu depende, ao mesmo tempo, do espaço disponível na tela e da distinção que pode ser feita entre as diferentes opções.

REFERÊNCIA: ISO 9241 parte 14 [1995]

8. Nos agrupamentos de dados, os itens devem estar organizados espacialmente segundo um critério lógico e facilitador.

   - COMENTÁRIO 1: Se a meta é ordenar dados em grupos lógicos, o primeiro critério de agrupamento é o seqüencial; o segundo critério deve ser o funcional; o terceiro deve ser a importância dos dados e o quarto critério deve ser a freqüência.

   - COMENTÁRIO 2: Quando os dados apresentados forem usados numa ordenação espacial ou temporal, deve-se considerar o agrupamento desses dados pela seqüência de uso, para preservar essa ordem.

Exemplo 1: A ordem dos dados em uma tela deve corresponder à ordem dos itens no formulário de papel associado (Figuras 80, 81, 82 e 83).

- COMENTÁRIO 3: Quando o conjunto de dados for associado a funções específicas, deve-se considerar o conjunto de cada grupo de dados, de modo a ilustrar essas relações funcionais.

Exemplo 2: As telas apresentadas anteriormente, indicam que cada grupo de dados está associado a uma função específica – Dados residenciais, dados comerciais e dados de identificação.

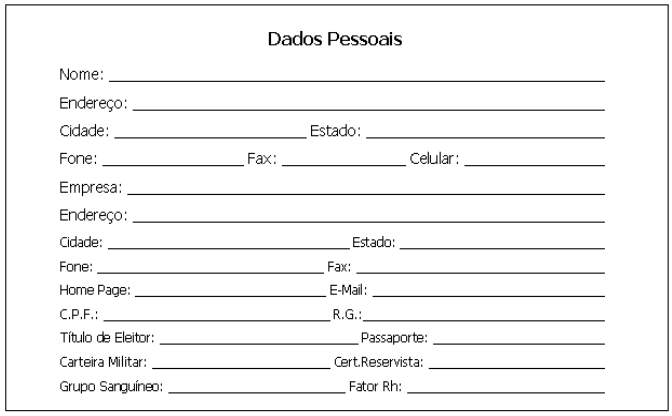


Fig. 80 – Formulário – Dados Pessoais.

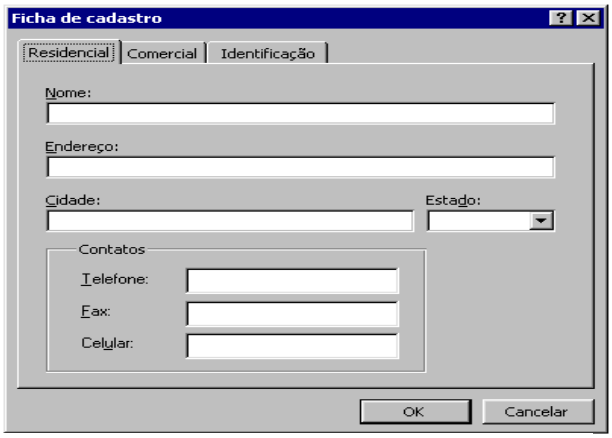


Fig. 81 – Ficha de Cadastro – Dados residenciais.

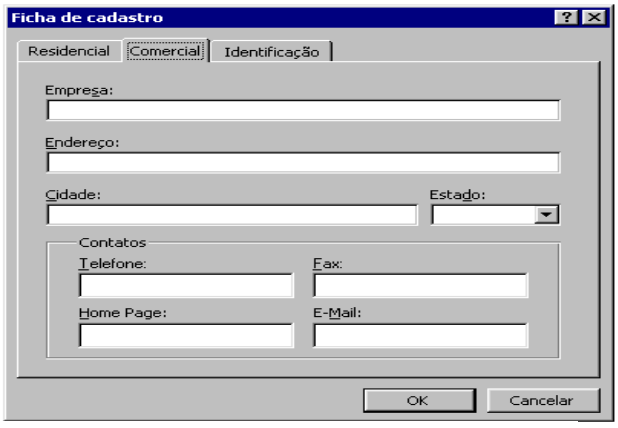


Fig. 82 – Ficha de Cadastro – Dados comerciais.

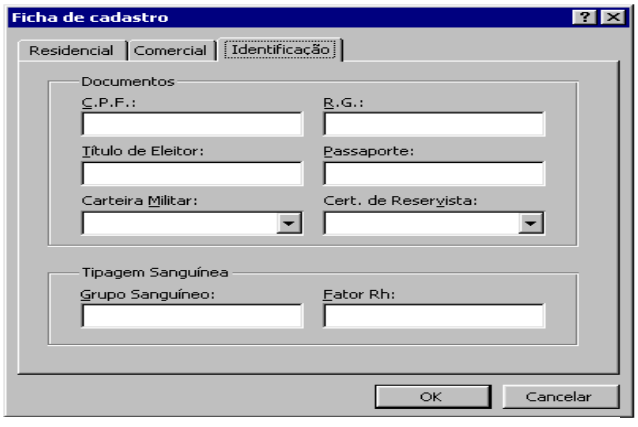
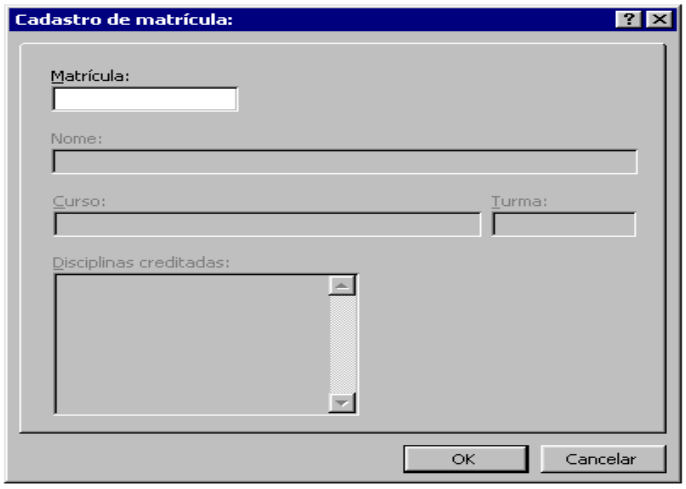


Fig. 83 – Ficha de Cadastro – Dados de identificação.

- COMENTÁRIO 4: Quando alguns itens de dados apresentados forem particularmente importantes, representando informação significativa e/ou necessitem resposta imediata do usuário, deve-se considerar a localização desses itens no topo da apresentação (Figuras 84a e 84b).

- COMENTÁRIO 5: Quando não existe lógica apropriada para o agrupamento de dados por seqüência, função ou importância, deve-se adotar algum outro princípio, tal como agrupamento cronológico ou alfabético (Figura 85).

REFERÊNCIA: Smith & Mosier [1986]



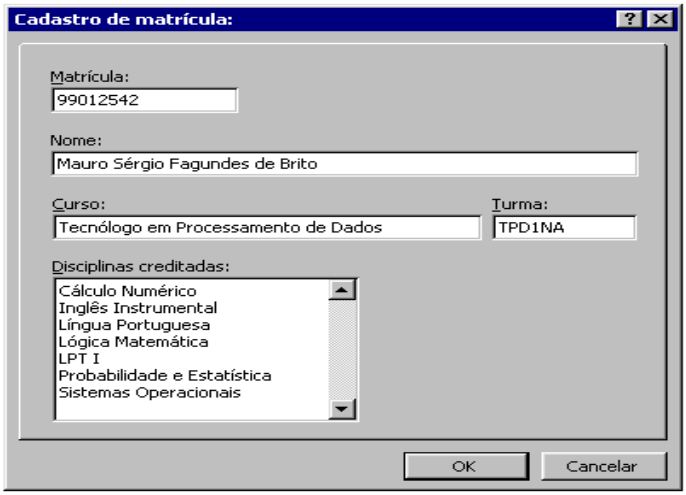


Fig. 84a e 84b – Informação significativa no topo da apresentação – Matrícula.

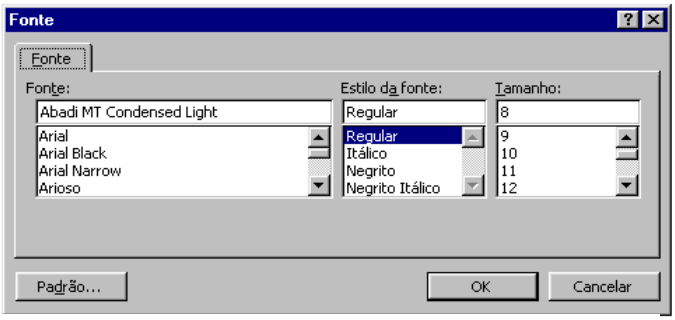


Fig. 85 – Agrupamento alfabético e/ou numérico – Fonte e Tamanho.

9. Nas listas de seleção, suas opções devem estar organizadas segundo alguma ordem lógica e coerente (Figura 86).

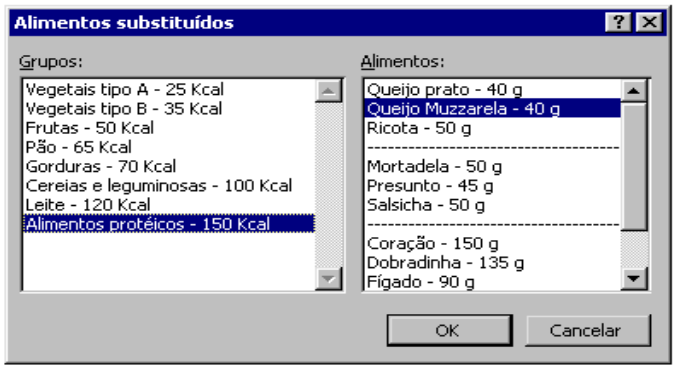


Fig. 86 – Lista de seleção.

   - COMENTÁRIO 1: Os critérios lógicos devem incluir freqüência de uso, ordem cronológica da tarefa, relações funcionais e ordem alfabética.

   - COMENTÁRIO 2: O projetista deve especificar a ativação de mecanismos de navegação internos (barras de rolamento) quando o número de escolhas possíveis se torne elevado.

   - COMENTÁRIO 3: Separadores devem ser empregados para marcar a organização dos itens segundo grupos lógicos.

   - COMENTÁRIO 4: Se qualquer organização não for possível, os separadores (um traço simples) devem ser dispostos a cada 6 linhas de alternativas.

REFERÊNCIAS: Brown [1988], ISO 9241 parte 12 [1994]

1.4.2. Agrupamento e Distinção de Itens por Formato

Definição: O critério de *Agrupamento/Distinção por Formato* diz respeito mais especificamente às características gráficas (formato, cor, etc.) que indicam se os itens pertencem ou não a uma dada classe, ou que indicam distinções entre classes diferentes, ou ainda distinções entre itens de uma dada classe.

Justificativa(s): Será mais fácil para o usuário perceber relacionamento(s) entre itens ou classes de itens, se diferentes formatos ou diferentes

códigos ilustrarem as suas similaridades ou diferenças. Deste modo, tais relacionamentos serão mais fáceis de aprender e de recordar. Um bom agrupamento/distinção por formato leva a uma boa condução.

Recomendações

1. O software deve apresentar uma distinção visual clara de áreas que possuem diferentes funções. (área de comandos, área de mensagens, etc).

   - COMENTÁRIO 1: Os separadores podem ser empregados para delimitar visualmente as zonas lógicas, de mensagens, funcionais ou de escolha de uma aplicação (Figura 87).

REFERÊNCIA: Smith & Mosier [1986]

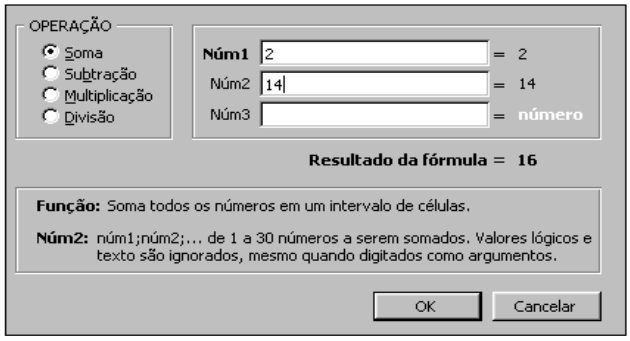


Fig. 87 – Áreas funcionais e de mensagens delimitadas por separadores.

2. Os dados críticos e que requeiram atenção imediata devem ser destacados através do emprego de sinais sonoros ou através do uso de cores eminentes para alertar os usuários, realçando as situações anormais (Figura 88).

   - COMENTÁRIO 1: "Destaque" é usado aqui no seu sentido geral, significando enfatizar ou fazer evidente, não sendo restrito a qualquer método particular de codificação de

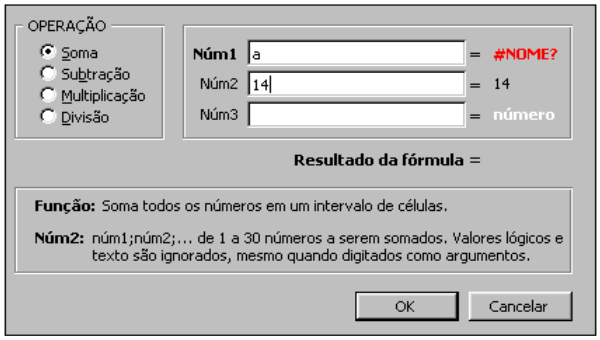


Fig. 88 – Tela de aparência uniforme com destaque em localização particular.

apresentação tal como brilho ou vídeo reverso.

- COMENTÁRIO 2: O destaque é mais efetivo quando usado esparsamente, funcionando melhor para telas com aparência relativamente uniforme.

- COMENTÁRIO 3: Para alguns propósitos, uma maneira de salientar é a codificação por posição, como na apresentação de itens numa localização consistentemente particular.

- COMENTÁRIO 4: Outros códigos auxiliares podem ainda ser necessários para salientar itens importantes, mesmo se forem posicionados consistentemente (Figura 89).

- COMENTÁRIO 5: A codificação colorida deve ser usada para dados que exijam atenção imediata. Os dados associados a estados indesejáveis e a alarmes devem ser apresentados em vermelho. Usa-se verde para indicar condições normais (Figuras 90 e 91).

- OBS. 1: Brilho e saturação para chamar a atenção

- COMENTÁRIO 6: Em alguns dispositivos os projetistas podem variar a intensidade, bem como a saturação da apresentação. Nessas apresentações, tanto a intensidade quanto a saturação podem ser usadas para chamar a atenção do

Fig. 89 – Palavra destacada em vermelho (link) – posicionamento consistente.

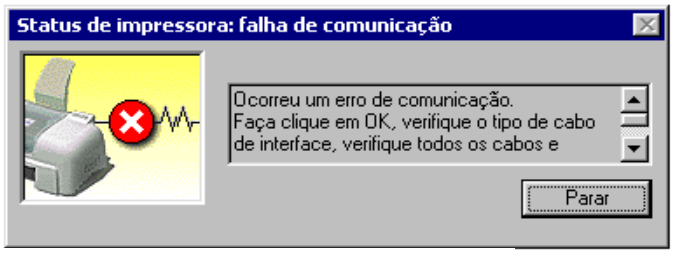


Fig. 90 – Alarme – codificação em vermelho.

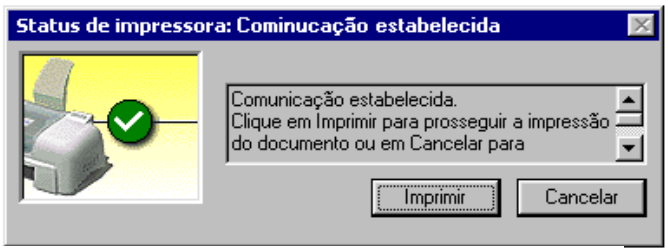


Fig. 91 – Condição normal – codificação em verde.

usuário para dados críticos (Figura 92a).

- COMENTÁRIO 7: Embora a saturação ou intensidade da cor sejam úteis para chamar a atenção dos usuários, seu uso exagerado pode resultar em uma apresentação extravagante, difícil de visualizar por longos períodos (Figura 92b).

- COMENTÁRIO 8: Quando decidir pela saturação de qualquer dispositivo colorido, considere a iluminação do ambiente no qual o dispositivo será visualizado. Cores que aparecem altamente saturadas em um ambiente escuro aparecerão menos saturadas em ambientes bem iluminados.

REFERÊNCIA: Smith & Mosier [1986]

3. Em situações em que é exigida atenção especial do usuário, as mensagens de alerta e de aviso devem ser apresentadas de maneira distinta das demais (Figura 93).

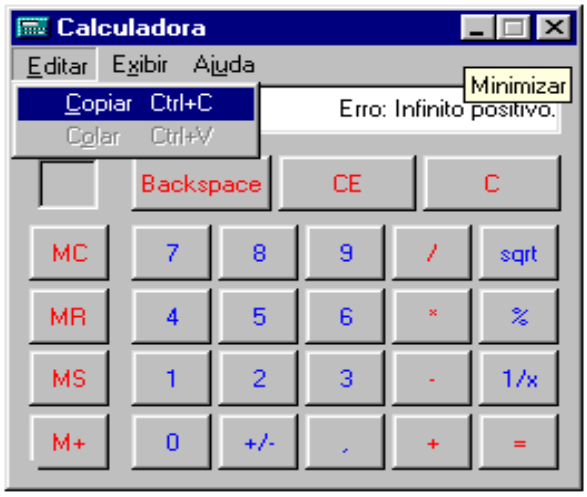


Fig. 92a – Brilho e saturação – exemplo positivo.

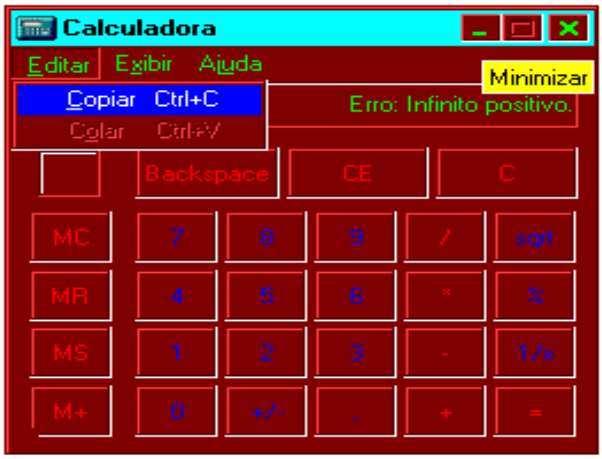


Fig. 92b – Brilho e saturação – exemplo negativo.

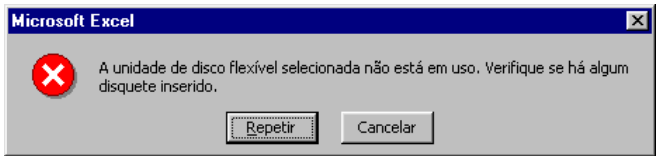


Fig. 93 – Janela individual de mensagem contendo símbolo visual em destaque.

4. Se houver necessidade de salientar palavras ou noções importantes na apresentação de textos, devem ser empregados recursos de estilo, como itálico, negrito, sublinhado ou diferentes fontes (Figuras 94, 95, 96 e 97).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

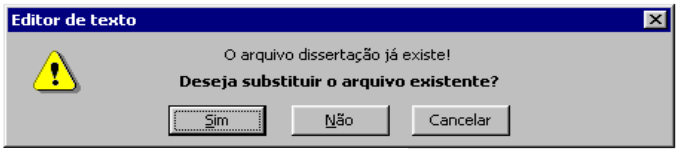


Fig. 94 – Codificação por negrito.

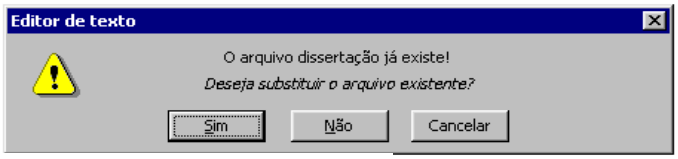


Fig. 95 – Codificação por itálico.

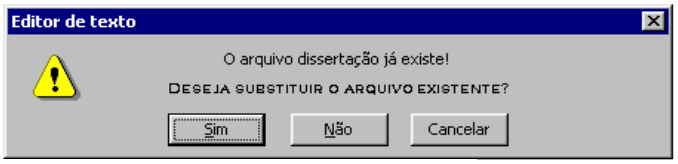


Fig. 96 – Codificação por fonte diferente.

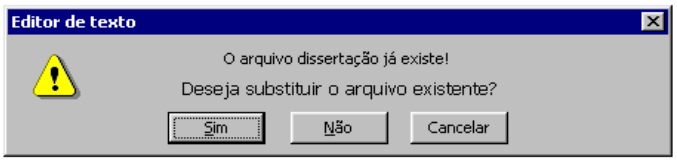


Fig. 97 – Codificação por tamanho de fonte.

5. Nas tabelas, os cabeçalhos devem estar diferenciados através do emprego de cores diferentes, letras maiores ou sublinhadas (Figura 98).

   - COMENTÁRIO 1: Existem muitos modos de distinguir diferentes tipos rótulos, incluindo diferenças consistentes no formato/localização nas telas, bem como fontes e marcadores especiais.

   - COMENTÁRIO 2: Se a tabela for muito complexa, os cabeçalhos podem ser codificados através de cores. O tamanho da letra e o sublinhado podem ser usados para distinguir e enfatizar cabeçalhos. O vídeo reverso pode ser usado para assinalar cabeçalhos mais importantes.

REFERÊNCIA: Smith & Mosier [1986]

Armarinho Sorriso					
Produto	Pç. de Compra		Pç. de Venda		Lucro/Prejuízo
Arroz Agulinha	R$	2,25	R$	3,16	R$ 0,91
Feijão Jalo	R$	3,15	R$	3,10	-R$ 0,05
Trigo Ponto Branco	R$	0,85	R$	1,23	R$ 0,38
Sal Du Mar	R$	0,56	R$	0,82	R$ 0,26
Óleo Esperança	R$	1,22	R$	1,20	-R$ 0,02

Fig. 98 – Codificação de cabeçalhos – Exemplo positivo.

6. As opções de menu que não estiverem disponíveis no momento, devem ser apresentadas de forma diferenciada visualmente.

   - COMENTÁRIO 1: Se opções geralmente não disponíveis podem, em um outro nível do

diálogo, tornar-se disponíveis e se a coerência da apresentação é importante, tais opções podem ser mostradas além das opções disponíveis. Todavia, uma codificação visual deve ser utilizada para diferenciar os dois tipos de opção (Figuras 99a e 99b).

- COMENTÁRIO 2: Os itens inativos de um menu não devem ser selecionáveis.

REFERÊNCIA: ISO 9241 Parte 14 [1995]

7. Os rótulos dos mostradores de dados devem ser visualmente diferentes dos dados aos quais estão associados.

   - COMENTÁRIO 1: Os rótulos devem apresentar diferenças quanto à forma e ao posicionamento, para ajudar os usuários a distingui-los de dados e outros elementos (Figura 100a e 100b).

REFERÊNCIAS: Bodart & Vanderdonckt [1993]

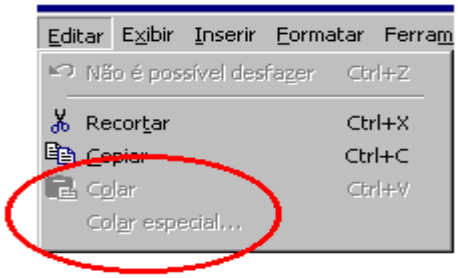


Fig. 99a – Opções de menu – não disponível.

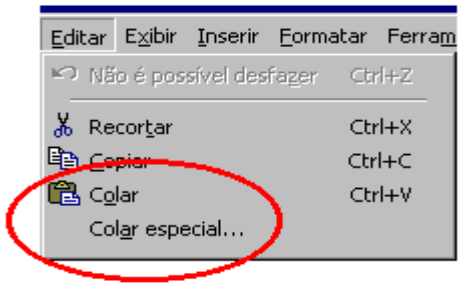


Fig. 99b – Opções de menu – disponível.

Fig. 100a e 100b – Rótulos - Diferenciados na forma e no posicionamento, respectivamente.

8. Para o software que apresentar telas de consulta, seus diferentes elementos (dados, comandos e instruções) devem ser visualmente distintos uns dos outros (Figura 101).

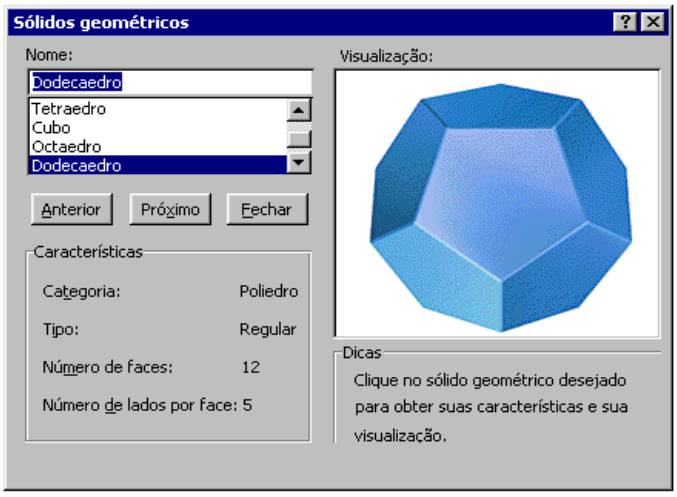


Fig. 101 – Sólidos geométricos – Tela de consulta.

9. Na ocorrência de várias opções ou ações possíveis, a mais provável ou mais lógica deve ser apresentada num formato que a distingue das outras, tal como uma borda circundando um botão (a opção *default*).

   - COMENTÁRIO 1: Nas caixas de mensagens, a situação por default de todo botão de comando deve ser suficientemente distinta no que se refere a sua apresentação (Figura 102).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

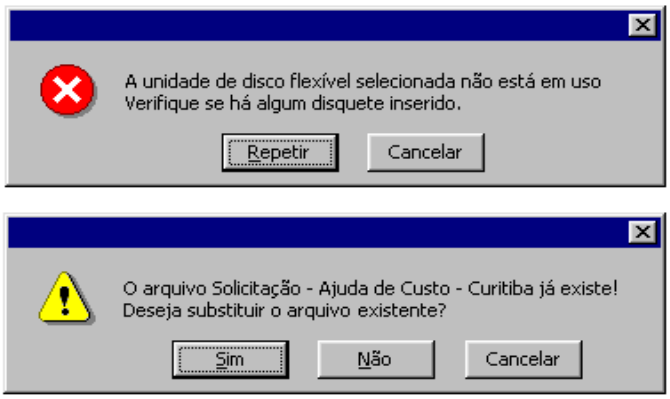


Fig. 102 – Botões Repetir e Sim – Opção default.

1.5. Qualidade das opções de Ajuda

Definição: Este critério avalia a conformidade da opção de ajuda oferecida pelo *software* e sua qualidade em orientar os usuário na busca de informações específicas ou na resolução de problemas.

Justificativa: O programa deve possibilitar recursos através de explicações fornecidas em telas opcionais de menus de ajuda. Estas opções visam contribuir para a superação das dificuldades que os usuários enfrentam na interação com o sistema ou permitir que encontrem instruções de utilização desejadas. A opção de menu de ajuda, quando bem orientada, conduz o usuário e facilita-lhe a aprendizagem tanto do sistema como dos conteúdos teóricos do *software* educacional.

Recomendações

- O software deve disponibilizar ao usuário a opção, *on line*, de menu "Ajuda" (Figura 103).

  - COMENTÁRIO 1: O sistema deve permitir a consulta, a qualquer momento, de toda informação e/ou recurso necessário para conduzir o usuário na execução de qualquer tarefa possível ao ambiente, independentemente da ajuda impressa.

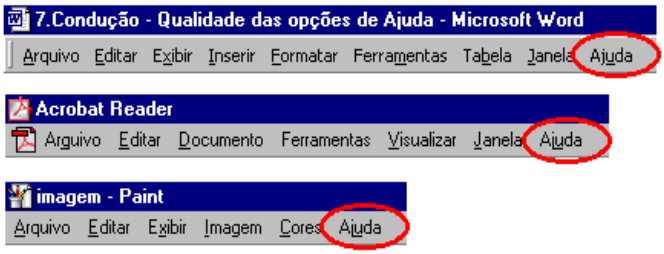


Fig. 103 – Exemplos positivos – Menu "Ajuda" *on line*.

1. A ajuda *on line* deve ser consistente com a documentação em papel no que se refere ao conteúdo.

   - COMENTÁRIO 1: Toda informação disponibilizada na ajuda on line deve estar de acordo com a ajuda impressa, permitindo que o usuário possa recorrer a ambas sem que haja risco de má interpretação ou inconsistência nos conteúdos.

- A ajuda *on line* deve ser consistente com todas as outras.

  - COMENTÁRIO 1: Quaisquer que sejam os meios disponibilizados de ajuda (*on line*, impressa, via internet, etc) é fundamental que apresentem total consistência de informação entre si.

- Os dispositivos de ajuda devem abranger a totalidade do sistema.

  - COMENTÁRIO 1: Todas as formas de ajuda devem proporcionar ao usuário, sendo ele iniciante, intermediário ou avançado, condições para realizar processos, tais como: instalação,

operação, manutenção e desinstalação do sistema.

- O acionamento da opção de ajuda deve estar estruturado no contexto da tarefa e da transação corrente (Figuras 104 e 105).

  - COMENTÁRIO 1: Deve haver uma estreita ligação entre a ajuda acionada e a tarefa em andamento, sem que seja necessário efetuar buscas para a obtenção da ajuda adequada.

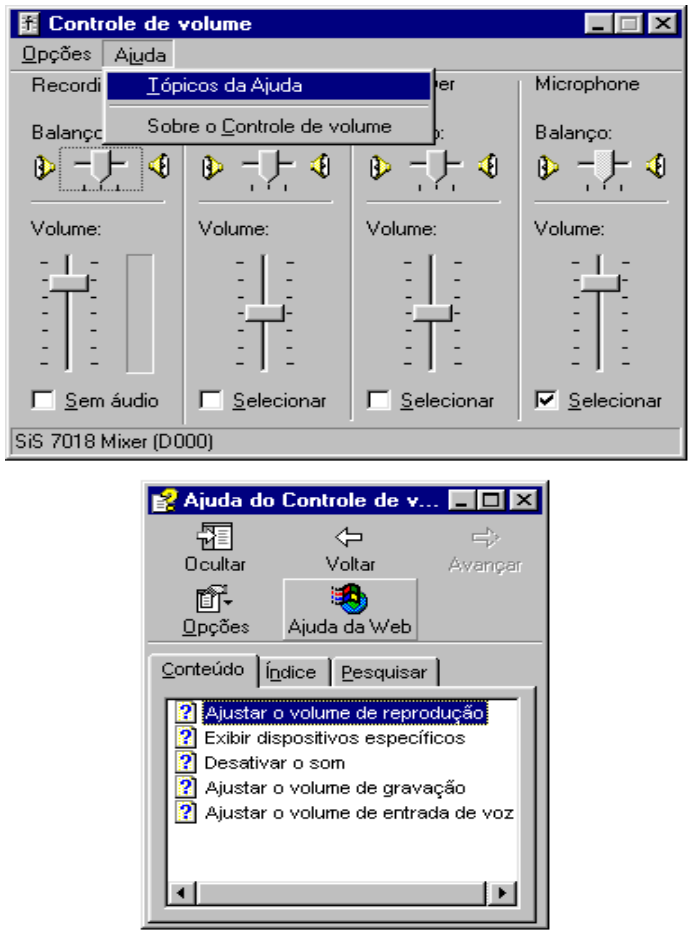


Fig. 104 – Exemplo positivo – Controle de volume.

Fig. 105 – Exemplo positivo – Gravador de som.

- COMENTÁRIO 2: A opção de ajuda estruturada no contexto da tarefa transmite ao usuário iniciante maior confiabilidade na utilização do sistema.

- COMENTÁRIO 3: A opção de ajuda estruturada no contexto da tarefa reduz o tempo de busca para solução de possíveis problemas ou dúvidas operacionais.

- O sistema de ajuda deve funcionar de forma exclusiva, não permitindo a interrupção da continuação na execução do sistema.

  - COMENTÁRIO 1: O sistema de ajuda deve ser acessível de forma independente, permitindo sua consulta sem que ocorra qualquer tipo de interrupção na execução de uma tarefa em andamento.

  - EXEMPLO POSITIVO: Durante a gravação de um som ou uma narração é possível acessar a ajuda do Gravador de som.

- O sistema deve apresentar diferentes formas de acesso aos conteúdos de ajuda.

  - COMENTÁRIO 1: Deve haver a possibilidade de ativação da ajuda por mais de um meio, permitindo seu acesso através de menu, barra de ferramenta, teclado, botões, etc (Figuras 106, 107, 108 e 109).

- O sistema de ajuda deve utilizar princípios de hipertexto, para permitir

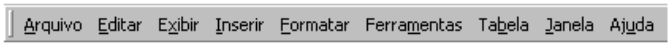


Fig. 106 – Barra de menu.

Fig. 107 – Barra de ferramenta.

Fig. 108 – Teclado – tecla F1.

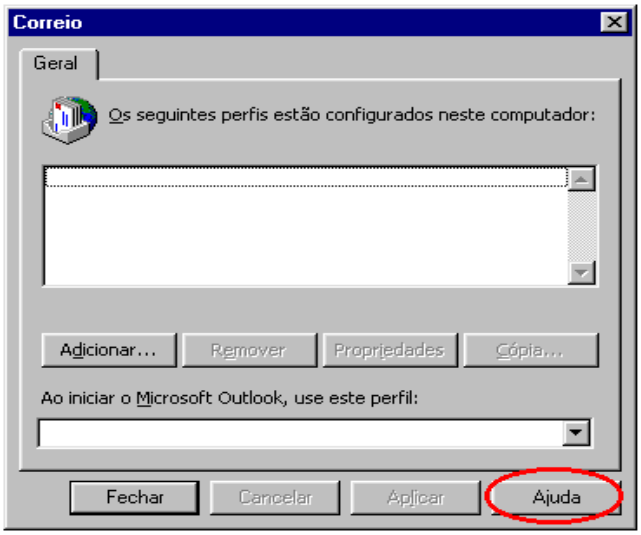


Fig. 109 – Botões de ajuda.

- ao usuário expandir tópicos por palavras chave (Figura 110).

2. O software deve disponibilizar ao usuário balões de ajuda para informar sobre a função de um botão, menu ou caixa de diálogo (Figura 111).

3. Nas caixas de mensagens de erro, o botão de comando “ajuda” deve estar sempre presente (Figura 112).

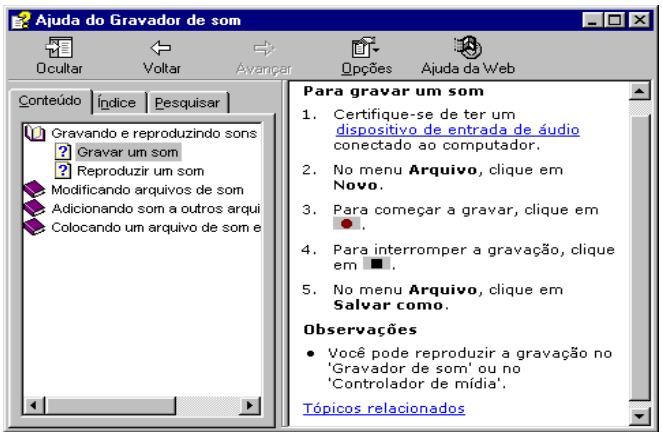


Fig. 110 – Hipertexto – dispositivo de entrada de áudio, Tópicos relacionados.

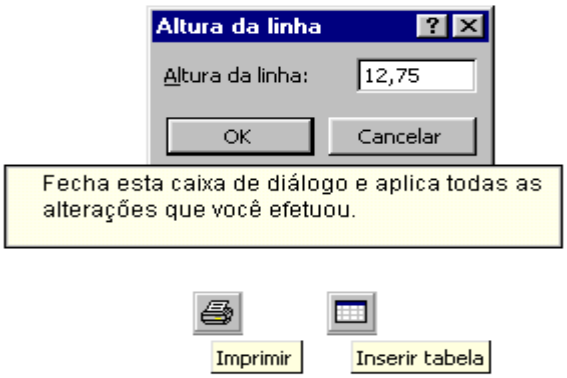


Fig. 111 – Balões de ajuda – informação sobre função de botões.

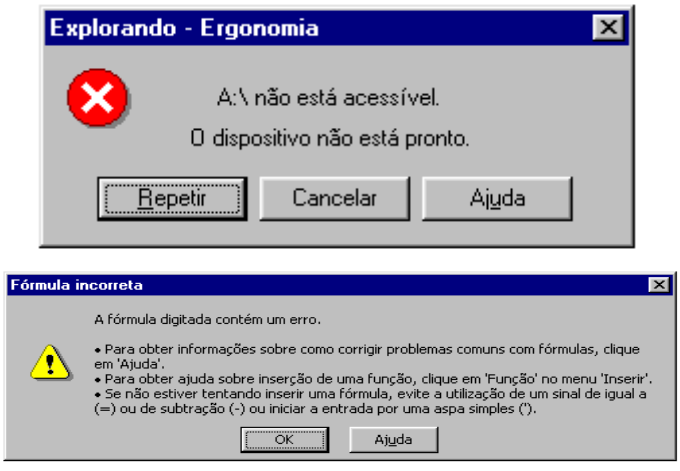


Fig. 112 – Mensagens de erro - Botão de ajuda presente.

2. Carga de Trabalho

Definição: O critério *Carga de Trabalho* diz respeito a todos elementos da interface que têm um papel importante na redução da carga cognitiva e perceptiva do usuário e no aumento da eficiência do diálogo.
O critério *Carga de Trabalho* está subdividido em três sub-critérios: *Brevidade* (o qual inclui *Concisão* e *Ações Mínima*s), *Carga mental* e *Densidade Informacional*.

Justificativa(s): Quanto maior for a carga de trabalho, maior será a probabilidade de cometer erros e mais longo se torna o aprendizado. Quanto menos o usuário for distraído por informação desnecessária, mais ele será capaz de desempenhar as suas tarefas eficientemente e atingir os objetivos educacionais propostos. Quanto menos ações forem necessárias, mais rápidas serão as interações e mais rapidamente o aluno consolidará os seus conhecimentos.

Sub-critérios - Carga de Trabalho

2.1. Brevidade

Definição: O critério *Brevidade* diz respeito à carga de trabalho perceptiva e cognitiva, tanto para entradas e saídas individuais, quanto para conjuntos de entradas (i.e., conjuntos e ações necessárias para se alcançar uma meta). *Brevidade* corresponde ao objetivo de limitar a carga de trabalho de leitura e entradas e o número de passos. O critério de *Brevidade* está subdividido em dois sub-critérios: *Concisão* e *Ações Mínima*s.

2.1.1. Concisão

Definição: O critério *Concisão* diz respeito à carga perceptiva e cognitiva de saídas e entradas individuais. Por definição, *Concisão* não diz respeito às mensagens de erro e de *feedbac*k.

Justificativa(s): A capacidade da memória de curto termo é limitada. Consequentemente, quanto menos entradas, menor a probabilidade de cometer erros. Além disso, quanto mais sucintos forem os itens, menor será o tempo de leitura.

Recomendações

1. A interface do software deve apresentar nomes concisos nas opções de menu, nas janelas, caixas de diálogo para facilitar sua memorização.

   - COMENTÁRIO 1: É conveniente que as opções sejam formuladas de maneira coerente e expressas de forma concisa.

   - COMENTÁRIO 2: Deve-se escolher identificadores de menus, telas, janelas e caixas que sejam pequenos e significativos o suficiente para serem aprendidos e relembrados facilmente (Figuras 113, 114 e 115).

REFERÊNCIA: Brown [1988]

2. Quando for necessária a utilização de senhas por parte do usuário, estas não devem ser maiores do que 4 ou 5 caracteres.

   - COMENTÁRIO 1: No momento da digitação, o usuário sempre encontra mais dificuldade para memorizar e empregar códigos mais longos.

3. O sistema deve oferecer valores "*default*" para acelerar a entrada de dados.

   - COMENTÁRIO 1: No início da transação de entrada de dados, deve-se apresentar os valores atualmente definidos como *default* em seus respectivos campos de dados (Figura 116).

Fig. 113 – Janela.

Fig. 114 – Menus.

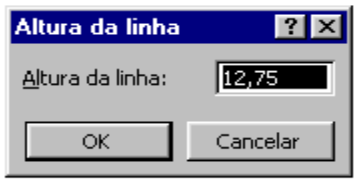


Fig. 115 – Caixa.

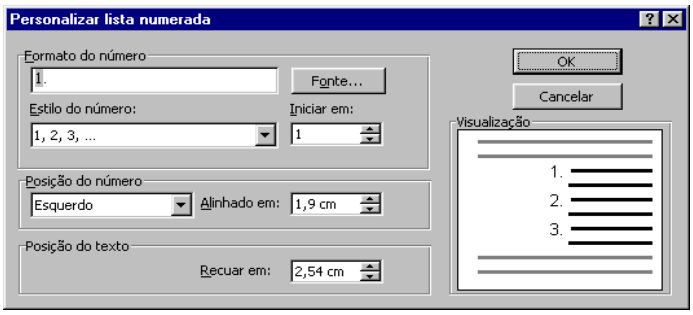


Fig. 116 – Valores definidos como *default* – Personalizar listas numeradas.

- COMENTÁRIO 2: Caso um conjunto de valores *default* tenha sido definido, em vez de fazer o usuário ter de aceitar cada valor individualmente, poderá ser mais rápido permitir que ele aceite todo o conjunto de *defaults* por meio de uma única ação (Figura 117).

- COMENTÁRIO 3: Quando os projetistas da interface não puderem predizer quais valores *default* serão úteis, deve-se permitir aos usuários (ou talvez a um administrador de sistemas) definir, mudar ou remover os valores *default* para qualquer campo de entrada de dados (Figura 118).

- COMENTÁRIO 4: Seria útil distinguir os valores entrados por *default* de entradas de dados novas (Figura 119).

REFERÊNCIA: Smith & Mosier [1986]

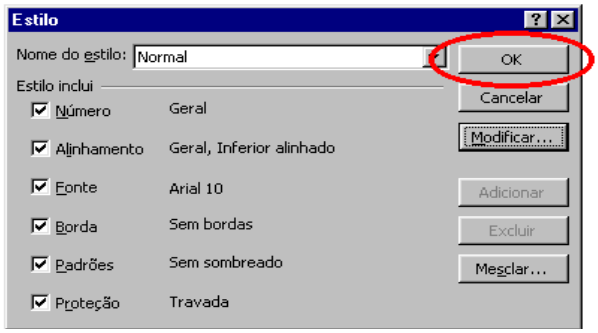


Fig. 117 – Botão OK – conjunto de valores *default* em uma única ação.

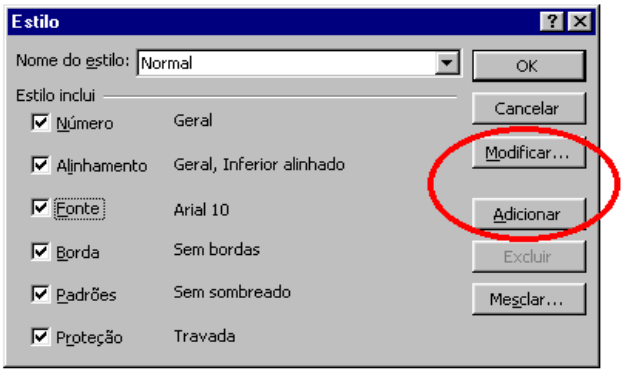


Fig. 118 – Botões Modificar e Adicionar – mudança dos valores *default*.

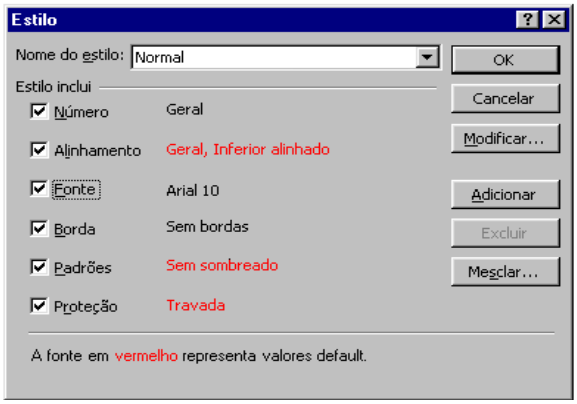


Fig. 119 – Distinção de valores *default* – caractere com fonte diferenciada.

- A interface deve possibilitar a repetição da entrada de dados quando as mesmas puderem ser reaproveitadas (Figuras 120 e 121).

  - COMENTÁRIO 1: Caso os dados inseridos na última entrada sirvam para a inserção seguinte, a interface deverá ser capaz de permitir o aproveitamento dos mesmos, de forma total ou parcial.

4. Na entrada de dados valores *default* devem ser exibidos nos campos apropriados.

   - COMENTÁRIO 1: Os campos de dados que pré-supõe possíveis valores *default* devem exibi-los, objetivando a minimização das ações do usuário (Figura 122).

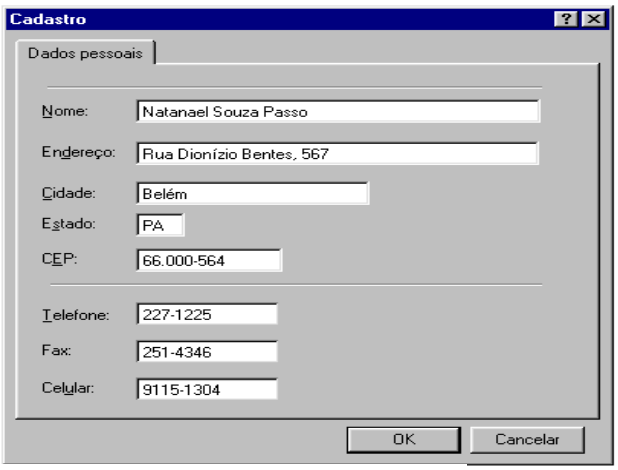


Fig. 120 – Última entrada de dados.

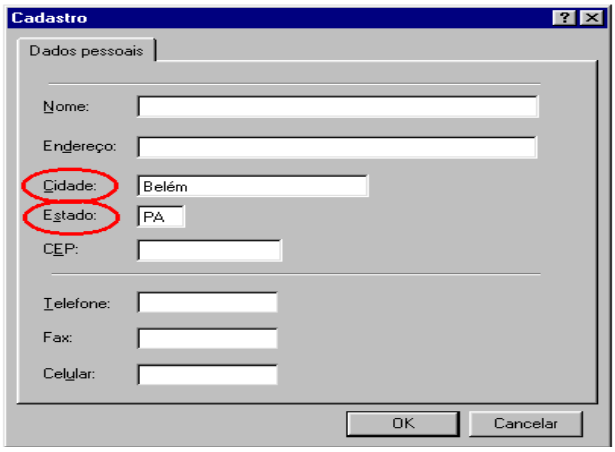


Fig. 121 – Nova entrada de dados – Dados reutilizáveis.

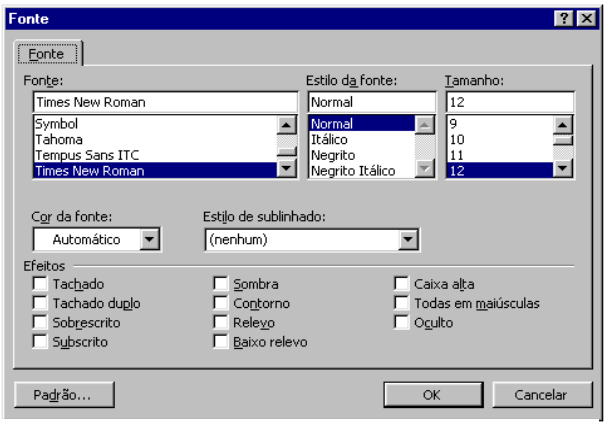


Fig. 122 – Valores default – Fonte, Estilo de fonte, Tamanho, Cor da fonte.

2.1.2. Ações Mínimas

Definição: O critério *Ações Mínimas* diz respeito à carga de trabalho em relação ao número de ações necessárias à realização de uma tarefa. O que temos aqui é uma questão de limitar tanto quanto possível o número de passos pelos quais o usuário deve passar.

Justificativa: Quanto mais numerosas e complexas forem as ações necessárias para se chegar a uma meta, a carga de trabalho aumentará e com ela a probabilidade da ocorrência de erros.

Recomendações

1. Somente as informações necessárias e utilizáveis devem ser apresentadas.

   - COMENTÁRIO 1: A quantidade de informações e ações a serem utilizadas deve ser a mínima necessária para se obter o resultado pretendido. Deve-se evitar ações que não contribuam para a tarefa estabelecida (Figura 123).

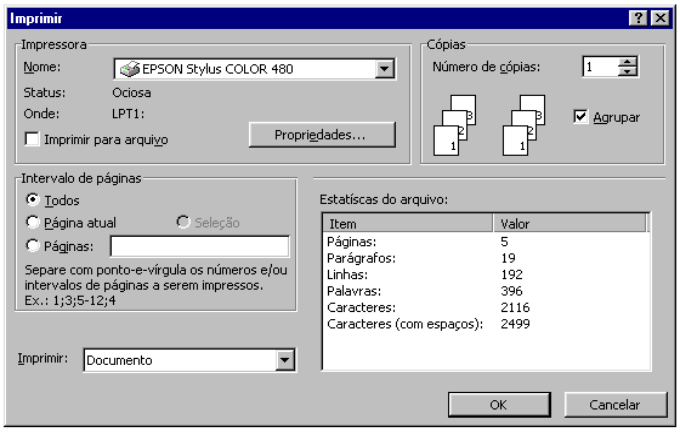


Fig. 123 – Exemplo negativo – Estatísticas do arquivo.

2. O número de passos necessários para se realizar uma seleção em menu deve ser minimizado (Figura 124)

   - COMENTÁRIO 1: Em alguns casos será interessante que a passagem pelas páginas intermediárias seja feita, objetivando ambientação do aplicativo (Figura 125 e 126).

Fig. 124 – Menu principal – Primeira página.

Fig. 125– Menu “Física”.

Fig. 126 – Menu “Força e Movimento”.

EXEMPLO NEGATIVO: Acesso à página "Pêndulos", percorrendo todas as páginas intermediárias (Figuras 127 e 128).

REFERÊNCIA: ISO 9241 parte 14 [1995], Smith & Mosier [1986]

3. A interface deve possibilitar a repetição da entrada de dados quando as mesmas poderem ser reaproveitadas.
   -

Fig. 127 – Página "Ressonância" – Link "Pêndulo".

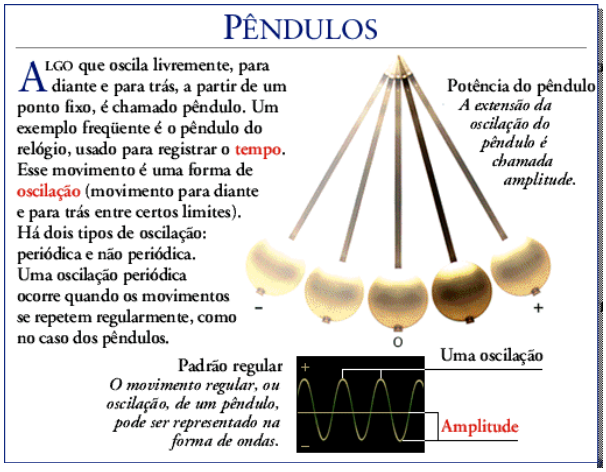


Fig. 128 – Página "Pêndulos".

COMENTÁRIO 1: Caso os dados inseridos na última entrada sirvam para a inserção seguinte, a interface deverá ser capaz de permitir o aproveitamento dos mesmos, de forma total ou parcial (Figuras 129 e 130).

4. A interface deve possibilitar que uma unidade de medida, quando associada a um campo, seja incluída como parte do campo de dados (Figura 131).

   - COMENTÁRIO 1: Na entrada de dados de valores que pressupõe uma unidade de medida a interface deve proporcionar a inserção dessa unidade de forma automática, sem que o usuário necessite digitá-la.

5. Na entrada de dados valores *default* devem ser exibidos nos campos apropriados.

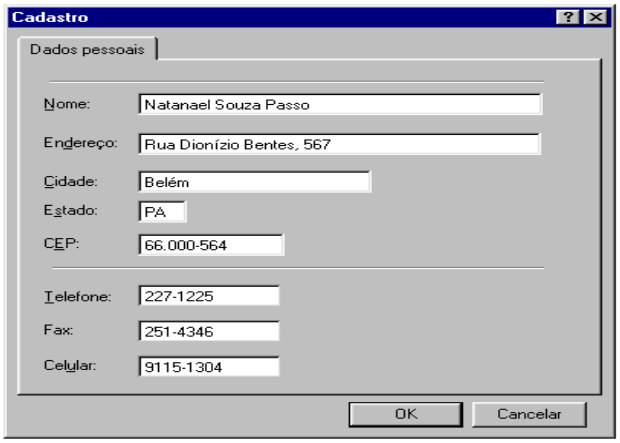


Fig. 129 – Última entrada de dados.

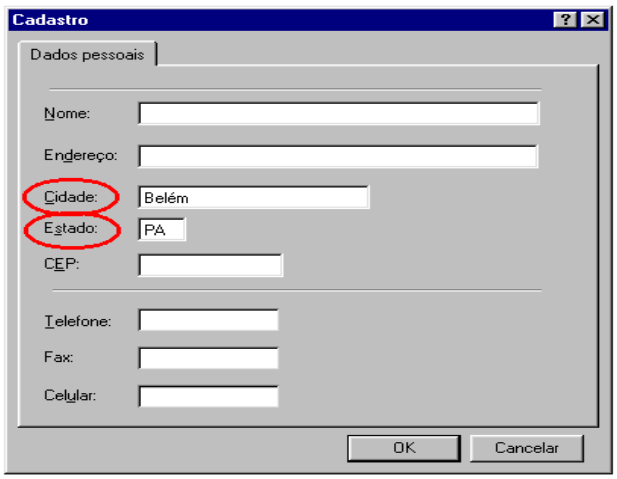


Fig. 130 – Nova entrada de dados – Dados reutilizáveis.

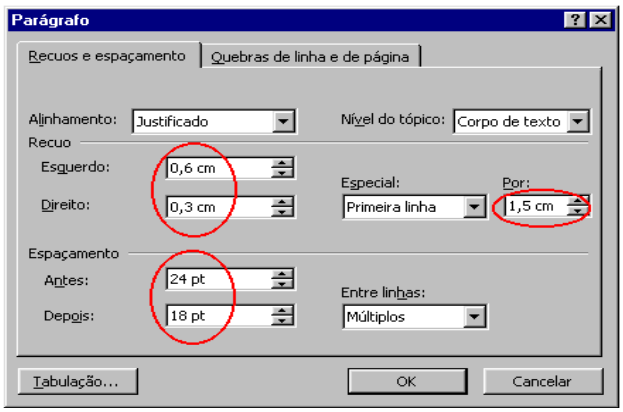


Fig. 131 – Unidade de medida inserida como parte do campo.

COMENTÁRIO 1: Os campos de dados que pré-supõe possíveis valores *default* devem exibi-los, objetivando a minimização das ações do usuário (Figura 132).

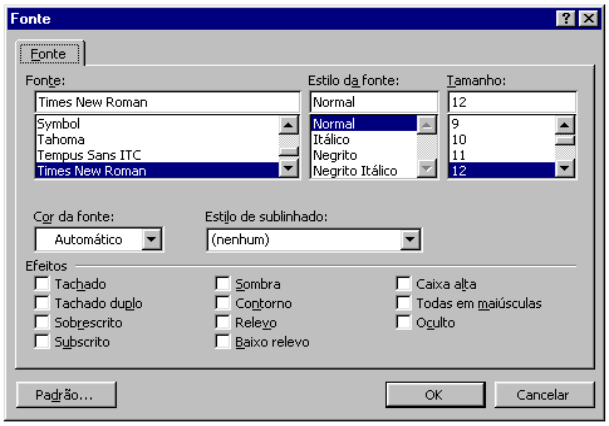


Fig. 132 – Valores default – Fonte, Estilo de fonte, Tamanho, Cor da fonte.

6. Quando várias páginas estiverem envolvidas, o sistema deve permitir que se vá, diretamente, a uma página sem ter que passar pelas intermediárias (Figura 133 e 134).

   - COMENTÁRIO 1: O sistema deve permitir que o usuário acesse qualquer página envolvida no processo, independentemente da ordem das mesmas, sem que haja prejuízo ou perda de informação.

   - COMENTÁRIO 2: O sistema deve disponibilizar diversas formas de acesso aleatório de suas páginas, podendo ser através da utilização de mouse, barra de ferramentas, teclado ou menu.

Fig. 133 – Receitas – primeira página

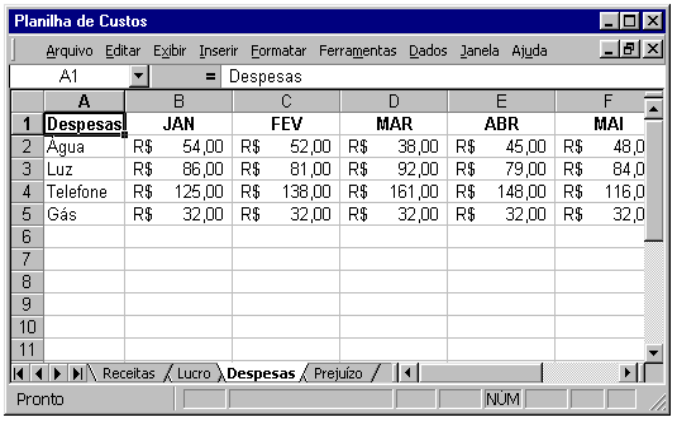


Fig. 134 – Despesas – terceira página.

2.2. Densidade informacional

Definição: O critério *Densidade Informacional* diz respeito à carga de trabalho do usuário de um ponto de vista perceptivo e cognitivo, em relação ao conjunto total de itens de informação apresentados ao usuário e não a cada elemento ou item individual.

Justificativa(s): Na maioria das tarefas, o desempenho dos usuários piora quando a densidade de informação é muito alta ou muito baixa. Nestes casos, é mais provável a ocorrência de erros. Itens que não estão relacionados com a tarefa devem ser removidos.
A carga de memorização do usuário deve ser minimizada. O sistema deve facilitar os meios para que o usuário possa concentrar-se diretamente na aprendizagem e memorização dos conceitos pedagógicos e não na memorização de listas de dados ou procedimentos complicados. O usuário não deve executar tarefas cognitivas complexas quando estas não estão relacionadas com a tarefa em questão. Esta deve ser orientada para a aquisição do conhecimento específico em questão e ser suportada por uma interface simples e de fácil utilização.

Recomendações

1. As informações devem estar bem distribuídas na tela e devem evitar a poluição visual (Figura 135).

   - COMENTÁRIO 1: A distribuição das informações deve ser feita de forma a permitir que o usuário mantenha a atenção apenas na aprendizagem e memorização dos itens, evitando a ocorrência de erros.

   - COMENTÁRIO 2: A poluição visual da tela pode acarretar desvio de atenção ou confusão na compreensão dos objetivos da tarefa.

Fig. 135 - Poluição visual – Tela com densidade informacional alta.

2. Todas as informações contidas na tela devem ser imprescindíveis para guiar ou auxiliar o usuário na compreensão dos conteúdos.

   - COMENTÁRIO 1: As informações contidas na tela devem contribuir de forma positiva para a assimilação dos conteúdos, nenhuma informação deve ser disponibilizada sem que tenha total participação na compreensão da tarefa (Figura 136).

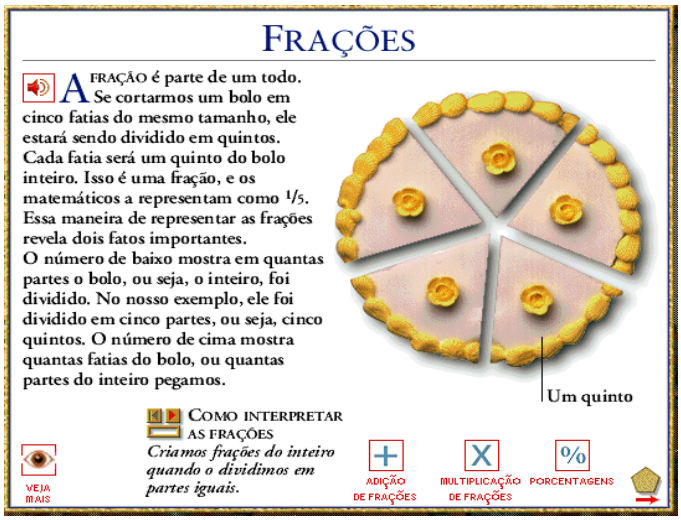


Fig. 136 – Exemplo positivo – Frações.

3. A densidade global das janelas deve ser reduzida.

   - COMENTÁRIO 1: A densidade de apresentação de uma tela, de uma caixa de diálogo ou de uma janela não deve ultrapassar 40%.

   - COMENTÁRIO 2: Não mais do que 10% da apresentação deve ser destacado de uma só vez.

- COMENTÁRIO 3: O tempo de busca de dados aumenta com a densidade da tela (Figura 137).

- COMENTÁRIO 4: Deve-se levar em conta que a tradução de um sistema do inglês para o português certamente ocupará mais espaço em tela.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

4. Em qualquer transação só devem ser fornecidos ao usuário os dados necessários e diretamente usáveis.COMENTÁRIO 1: Adaptar as apresentações de dados às necessidades dos usuários, oferecendo em qualquer situação, somente dados necessários e imediatamente úteis; não sobrecarregue apresentações com dados impertinentes (Figura 138).

- COMENTÁRIO 2: Apresentação de dados estranhos pode confundir o usuário e prejudicar a assimilação dos itens necessários (Figura 139).

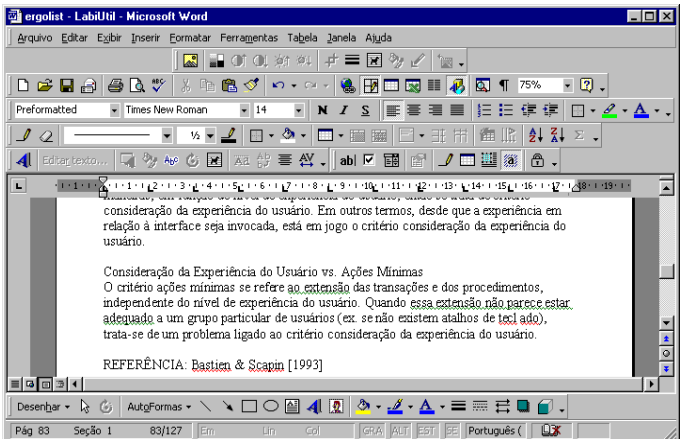


Fig. 137 – Exemplo negativo – Excesso de barra de ferramentas.

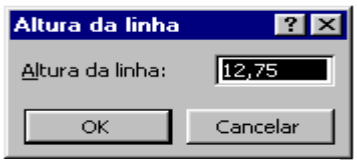


Fig. 138 – Exemplo positivo.

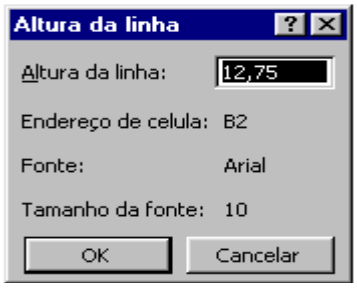


Fig. 139 – Exemplo negativo.

- COMENTÁRIO 3: Quando o projetista não puder antecipar as necessidades do usuário, permita que ele, usuário, controle as apresentações através da supressão/inserção dos dados nas telas (Figura 140).

REFERÊNCIA: Smith & Mosier [1986]

5. Tanto a Barra de Menu como as Opções de Menu devem apresentar apenas as opções necessárias para atingir os fins específicos.

- COMENTÁRIO 1: Nenhum item que seja independente da tarefa do usuário pode ser apresentado no menu. Todos os itens devem ter sua função clara e bem definida em relação à tarefa desejada.

- COMENTÁRIO 2: As opções de menu devem ser necessárias e suficientes, evite comandos sem objetivos claros ou redundantes (Figura 141).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

6. Para que o usuário não perca a concentração ou sobrecarregue sua memória, o sistema deve evitar apresentar um grande número de janelas.

- COMENTÁRIO 1: Informação em demasia na tela desvia a atenção do usuário de sua tarefa principal. Deve-se Evitar, portanto, veicular muita informação em muitas janelas ao mesmo tempo. A criação de janelas deve levar em conta uma restrição fundamental do desempenho do usuário - a

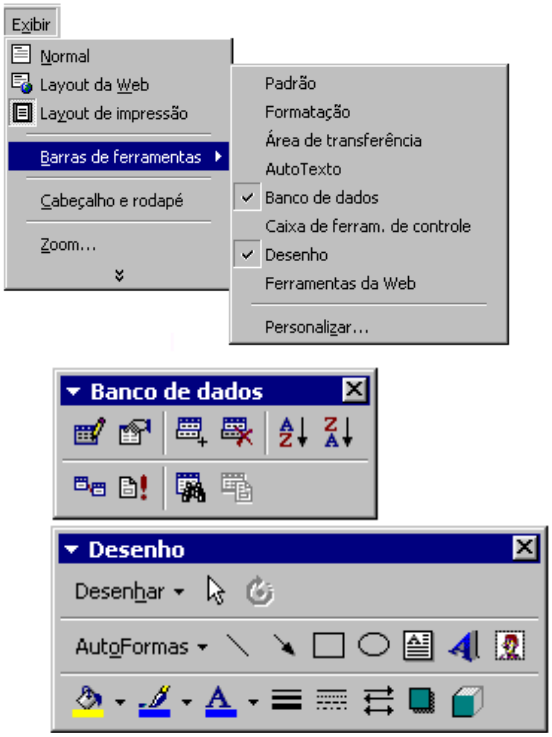


Fig. 140 – Menu exibir, Barra de ferramentas – Banco de dados e Desenho.

Fig. 141 – Exemplo negativo – Menu Cores na calculadora.

memória humana de curto termo (Figura 142).

REFERÊNCIA:Bodart & Vanderdonckt [1993]

- O sistema deve minimizar a necessidade do usuário em lembrar dados exatos na mudança de uma tela para outra.
  - COMENTÁRIO: O usuário não deve ter de decidir de memória qual ação ele deve tomar. Falhas na memória levarão a erros e a tempo perdido.
  - COMENTÁRIO: É melhor apresentar uma lista das opções atualmente ativas do que exigir que os usuários memorizem quais opções estão disponíveis a cada passo no diálogo (Figura 143).

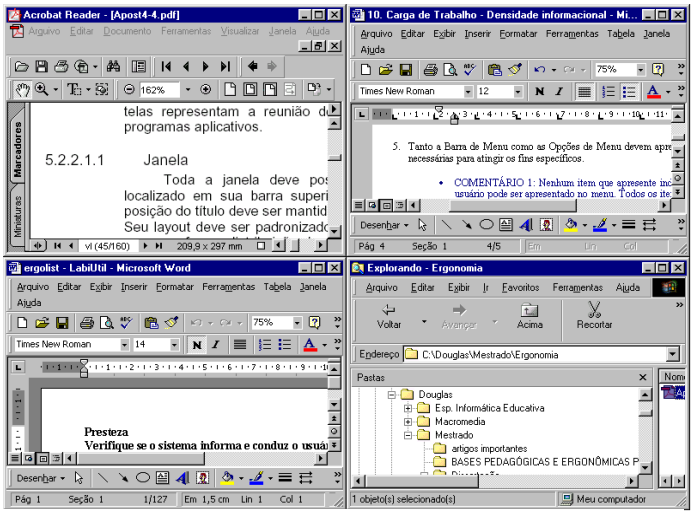


Fig. 142 – Número excessivo de janelas abertas – Informação em demasia na tela.

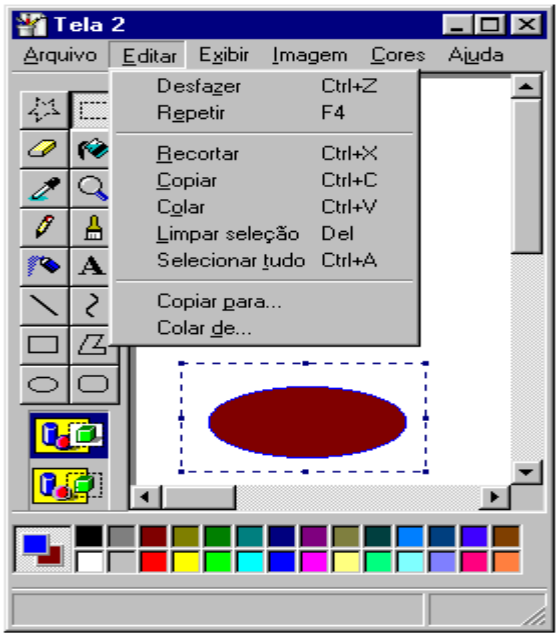


Fig. 143 – Tela 2 – Objeto selecionado.

- COMENTÁRIO: As opções temporariamente inativas (aquelas que não são aplicáveis no estágio atual do diálogo) também podem ser apresentadas se seu status inativo for mostrado explicitamente (Figura 144).

REFERÊNCIA: Brown [1988]

7. As listas de seleção e combinação devem apresentar uma altura máxima de nove linhas.

   - COMENTÁRIO 1: O comprimento recomendado de uma lista deve permitir a visualização imediata de apenas 7 itens, podendo variar dois itens para mais ou para menos (Figuras 145 e 146).

   - COMENTÁRIO 2: O projetista deve especificar a ativação de mecanismos de navegação internos (barras de rolamento) quando o número de escolhas possíveis se torne elevado.

   - COMENTÁRIO 3: O limite máximo é de algo como 50 itens que devem ser ordenados logicamente, segundo a freqüência de uso e/ou uma ordem alfabética.

REFERÊNCIA:Bodart & Vanderdonckt [1993]

8. O sistema deve promover a computação automática de dados derivados já disponíveis no computador, sem a necessidade do usuário calcular ou entrar com os mesmos.

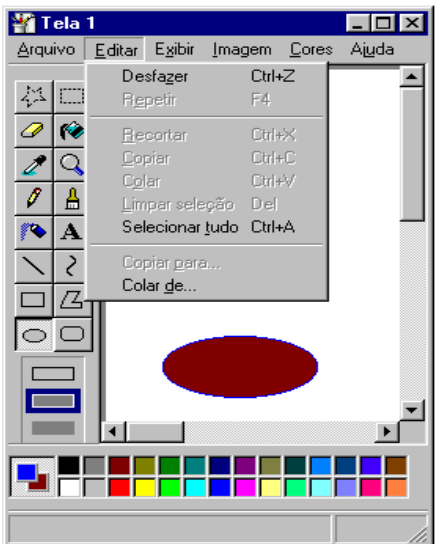


Fig. 144 – Tela 1 – Objeto não selecionado.

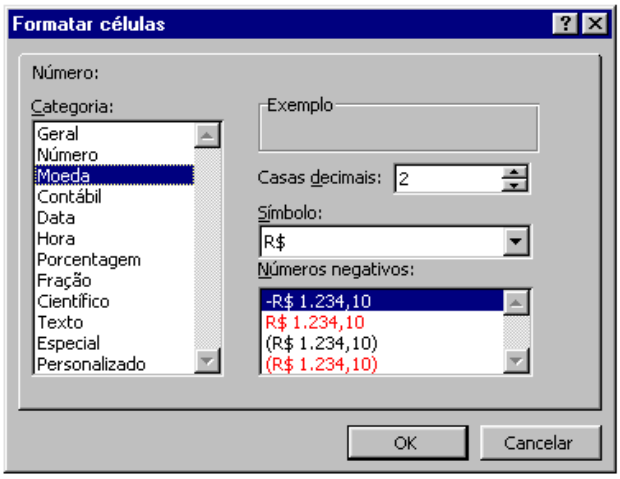


Fig. 145 – Exemplo negativo – Lista de seleção com altura para 12 itens.

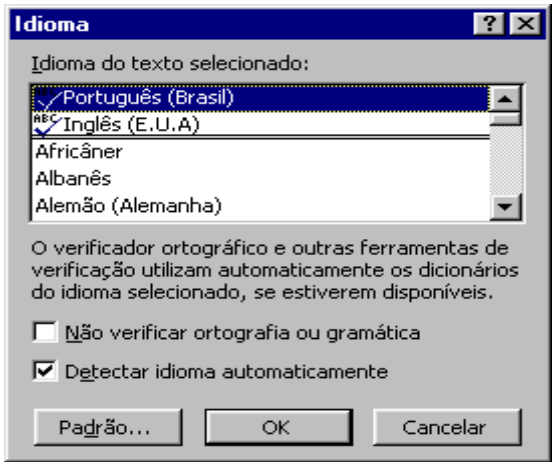


Fig. 146 – Exemplo positivo – Lista de seleção com altura para 5 itens.

- COMENTÁRIO 1: Deve-se apresentar os dados diretamente aos usuários na forma usual e não fazer com que o usuário tenha que converter dados apresentados (Figuras 147 e 148).

REFERÊNCIA: Smith & Mosier [1986]

9. As opções de codificação por cores devem ser limitadas em número.

- COMENTÁRIO 1: Minimize a codificação por cores, usando poucas cores para designar categorias críticas de dados apresentados (Figura 149)

- COMENTÁRIO 2: Use cores com parcimônia; duas ou três cores por vez são quase sempre suficientes. Mais cores podem ser usadas, se tornarem clara a estrutura lógica da informação. Nunca use cores extras por conta própria. A cor deve complementar a qualidade, e não compensar os defeitos de uma apresentação.

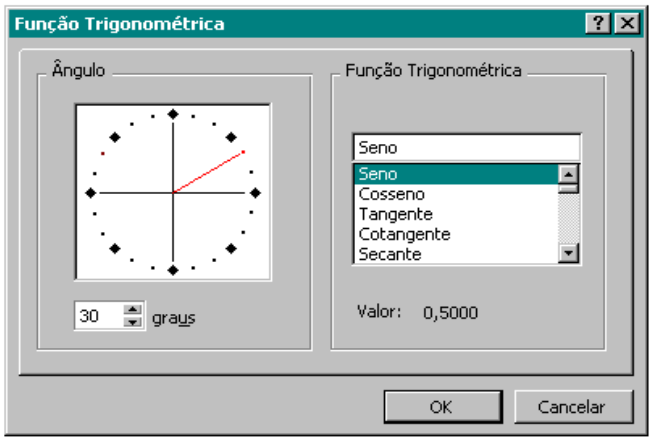


Fig. 148 – Ângulos e funções trigonométricas.

Fig. 147 – Temperatura – Graus Celsius ou Fahrenheit.

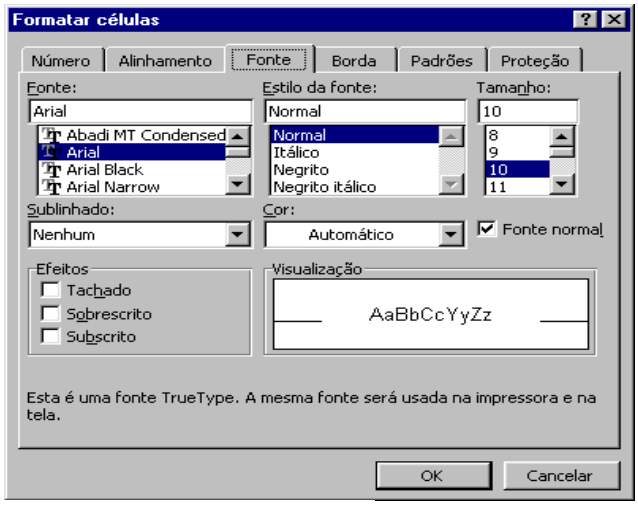


Fig. 149 – Exemplo positivo.

- COMENTÁRIO 3: O uso de cores casuais ou arbitrárias nas apresentações pode gerar telas excessivamente carregadas. O uso de cores sem critério também pode reduzir a precisão da interpretação por semelhança de uma determinada apresentação, diminuindo a rapidez e precisão da interpretação do usuário (Figura 150).

REFERÊNCIA: Smith & Mosier [1986]

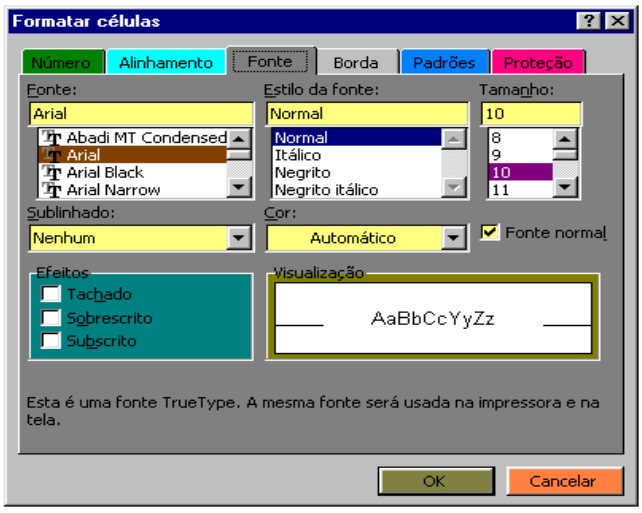


Fig. 150 – Exemplo negativo.

2.3. Carga Informacional

Definição: Este critério diz respeito à objetividade com que a informação pedagógica é apresentada. Avalia se a carga de conteúdo informacional apresentada é confortável e adequada ao usuário, em relação tanto aos conteúdos teóricos como práticos do *software* educacional.

Justificativa: Geralmente, o excesso de informações prejudica a compreensibilidade e a memorização dos elementos. A carga educacional deve ser dimensionada para que o aluno possa assimilar adequadamente as informações. Quanto mais objetivas e ilustrativas forem as informações de caracter pedagógico e quanto mais sucintos forem os itens, menor será o tempo de leitura e consequentemente melhor será a capacidade de memorização do conteúdo apresentado.

Recomendações

1. A carga informacional apresentada deve ser equilibrada e deve estar bem distribuída em unidades de informação.

   - COMENTÁRIO 1: O conteúdo informacional deve ser distribuído em unidades de modo a facilitar o processo de aprendizagem apresentando a quantidade de informação necessária, proporcionando uma assimilação clara e confortável (Figura 151).

2. Os conteúdos teóricos apresentados devem ser objetivos.

   - COMENTÁRIO 1: Todo conteúdo teórico deve ser relevante em relação ao objetivo educacional pretendido, não devendo conter informações que possam desviar a atenção do usuário.

3. Os exercícios práticos propostos devem ser objetivos.

   - COMENTÁRIO 1: Todo exercício contido no software deve ter como meta a complementação do conteúdo teórico apresentado.

   - COMENTÁRIO 2: Os exercícios devem ser objetivos, tendo seu foco direcionado para cada unidade desenvolvida, proporcionando o reforço adequado para a assimilação do conteúdo.

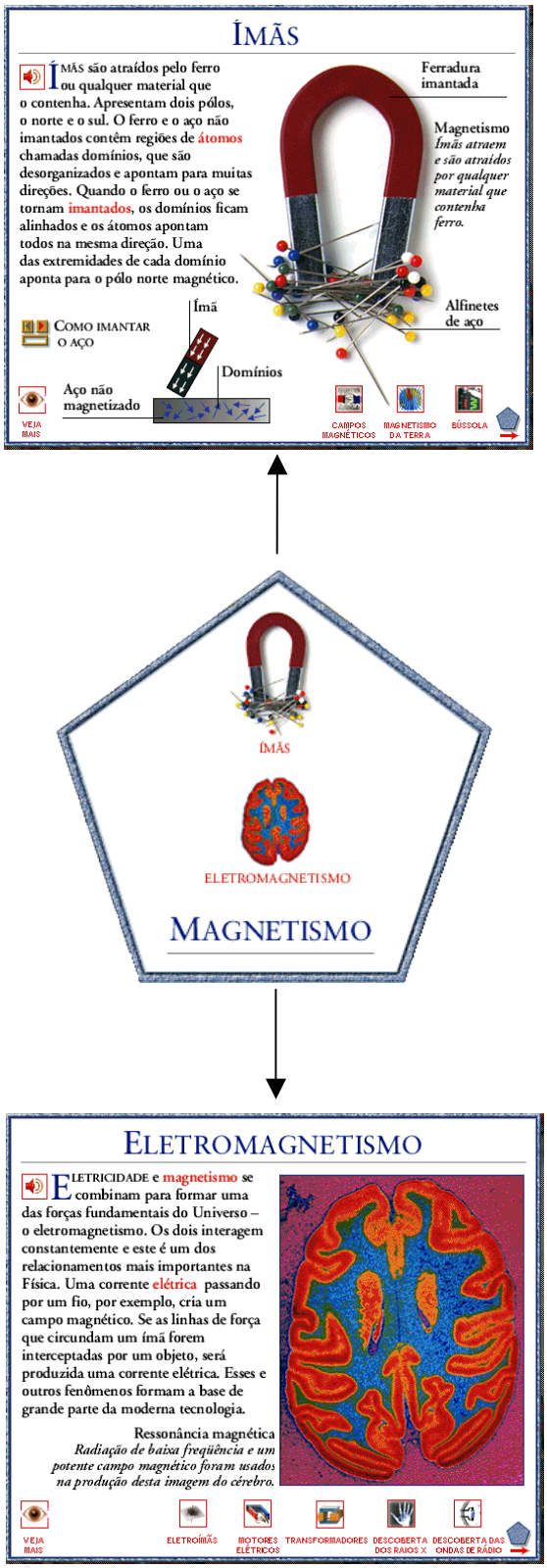


Fig. 151 – Unidade Magnetismo e suas sub-unidades Ímãs e Eletromagnetismo.

4. O software deve favorecer que o aluno mantenha um nível adequado de atenção e concentração.

   - COMENTÁRIO 1: O software deve manter a atenção do aluno, evitando recursos que desviem sua atenção ou que possam prejudicar sua concentração.

   - COMENTÁRIO 2: Deve-se ter cuidado na utilização de recursos multimídia evitando que os mesmos atraiam a atenção do aluno mais do que o objetivo principal da aprendizagem.

5. O software deve utilizar adequadamente os estímulos para fixação de conceitos e habilidades cognitivas.

   - COMENTÁRIO 1: Deve-se evitar a utilização de recursos audiovisuais, multimídia ou outro tipo de estímulo que não colabore para a estimulação das habilidades cognitivas e/ou para a assimilação e fixação de conteúdos pedagógicos.

6. A carga de informação apresentada deve estar adequada aos usuários previamente definidos.

   - COMENTÁRIO 1: É necessário que se faça um estudo prévio para se avaliar as características do usuário a quem se destina o software.

   - COMENTÁRIO 2: O software deve apresentar uma carga de informação adequada para cada tipo de usuário, não devendo o mesmo ser desenvolvido de forma generalista.

   - COMENTÁRIO 3: A definição do tipo de usuário deve levar em conta idade, escolaridade, assim como todo critério que possa influenciar no processo de aprendizagem.

7. A carga de informação apresentada deve estar adequada à disciplina de ensino.

   - COMENTÁRIO 1: A carga de informação deve conter dados envolvendo somente ao conteúdo didático proposto. Deve-se evitar dados, referências ou exercícios que necessitem de conhecimentos relacionados com outras disciplinas.

8. A carga de informação apresentada deve possuir bom equilíbrio entre a teoria e a prática.

   - COMENTÁRIO 1: Uma grande carga de conhecimento teórico pode ser difícil de assimilar, sem que se apresente, de forma intercalada, um bom conjunto de exercícios.

   - COMENTÁRIO 2: Muitos exercícios apresentados sem uma carga teórica que justifique a presença destes, pode gerar exercícios semelhantes e cansativos ou exercícios de difícil solução por falta de base teórica.

3. Controle Explícito

Definição: O critério *Controle Explícito* diz respeito tanto ao processamento explícito pelo sistema das ações do usuário, quanto do controle que os mesmos têm sobre o processamento de suas ações pelo sistema.
O critério *Controle Explícito* subdivide-se em dois critérios: *Ações Explícitas do Usuário* e *Controle do Usuário*.

Justificativa(s): Quando os usuários definem explicitamente as suas entradas e quando estas entradas estão sob o seu controle, os erros e ambiguidades são limitados. Além disso, o sistema será mais bem aceito se houver a possibilidade de controlar a apresentação dos diálogos de interação. A autonomia dos controles contribui para a adequação ao ritmo do processo de ensino/aprendizagem e para o incremento da motivação na interação com o programa.

Sub-critérios - Controle Explícito

3.1. Ações Explícitas do usuário

Definição: O critério *Ações Explícitas do Usuário* refere-se às relações entre o processamento pelo computador e as ações do usuário. Esta relação deve ser explícita, i.e., o computador deve processar somente aquelas ações solicitadas e somente quando solicitado para o fazer.

Justificativa(s): Quando o processamento pelo computador resulta de ações explícitas dos usuários, estes aprendem e entendem melhor o funcionamento da aplicação e menos erros são cometidos.

Recomendações

- O processamento das ações deve ser efetuado somente quando solicitadas pelo usuário.

    - COMENTÁRIO 1: O sistema deve aguardar até que o usuário decida o momento exato do sistema executar suas ações, utilizando alguma ação explicita através de um periférico de entrada para disparar o processo.

1. O sistema deve exigir uma ação explícita de ENTER ou similar (clique do mouse), para dar inicio ao processamento dos dados.

    - COMENTÁRIO 1: Não inicie o processamento como um efeito colateral de alguma outra ação, adie o processamento até que uma ação explícita de "ENTER" seja desencadeada permitindo ao usuário a revisão dos dados e a correção dos erros antes do processamento. Isso será, particularmente, útil quando a entrada dos dados for complexa e/ou for de difícil reversão.

    - EXEMPLO 1: A ação de retornar a um menu não deverá provocar o processamento de dados incompletos digitados na tela. (Figura 152)

Fig. 152 – O acionamento do menu Ajuda não processa a tarefa incompleta.

REFERÊNCIA: Smith & Mosier [1986]

2. O sistema deve adiar os processamentos até que as ações de entrada tenham sido encerradas.

    - COMENTÁRIO 1: Permita aos usuários controlar a seqüência das transações através de ações explícitas; adie o processamento

até que uma ação explícita do usuário seja comandada.

- COMENTÁRIO 2: Em um software educativo, mesmo que dados incorretos tenham sido digitados se faz necessário esperar pela ação "ENTER" do usuário, pois, a resposta incorreta deverá ser analisada e avaliada (Figura 153).

- EXEMPLO 1: Quando um usuário estiver digitando uma entrada de dados extensa, o computador não deve interromper o usuário para exigir correção imediata de qualquer erro de entrada, em vez disso, deve esperar pela ação "ENTER" do usuário.

- COMENTÁRIO 3: Ao interromper a ação do usuário, o computador retira deste a possibilidade de controlar a seqüência. Sendo assim, o usuário é forçado a efetuar uma seqüência de correção do erro, segundo a concepção do projetista da interface, prejudicando o aprendizado.

- COMENTÁRIO 4: Em geral, a detecção automática de problemas nas entradas correntes dos usuários pode ser negociada no final de uma transação, antes de sua implementação. Alarmes não interruptores e mensagens de alerta podem ser apresentados para informar sobre o monitoramento computacional de eventos externos a fim de que o usuário possa escolher quando realizar a correção.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

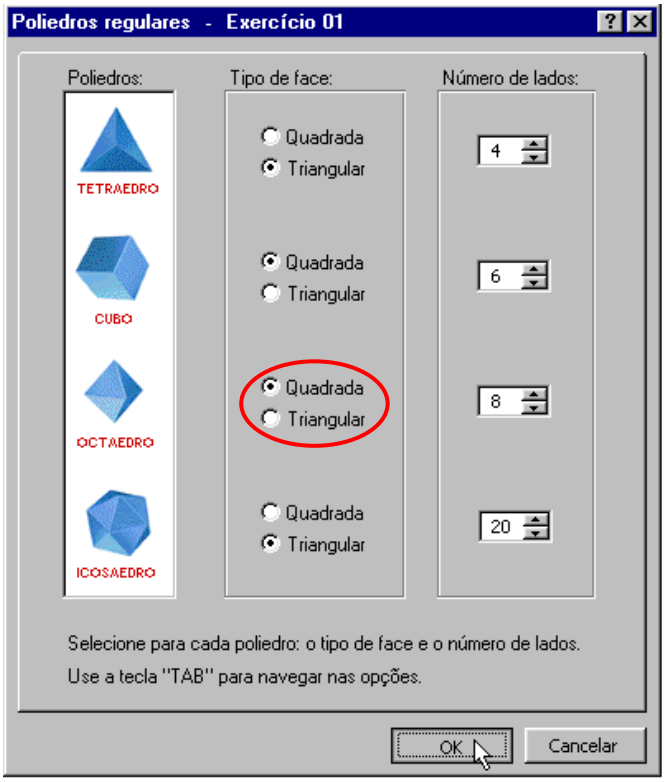


Fig. 153 – A resposta incorreta será analisada após a ação "ENTER" (OK).

- Durante a seleção de uma opção de menu o sistema deve permitir a separação entre indicação e execução da opção (Figura 154).

  - COMENTÁRIO 1: Se a seleção do menu for feita através de dispositivo de apontamento, faça a ativação em dois passos. Primeiramente, posicione o cursor para designar a opção selecionada, e a seguir, faça uma entrada de controle explïcita.

  - COMENTÁRIO 2: Se a seleção de menu é efetuada por apontamento, forneça a ativação em duas etapas, na qual a primeira ação designa a opção selecionada, seguida por uma segunda ação que confirma essa entrada tornando-a explícita.

  - EXEMPLO 1: Em uma tela tátil, o computador pode apresentar uma caixa "ENTER" separada, que pode ser tocada pelo usuário para comandar a ação de processar a opção selecionada.

REFERÊNCIA: Bodart & Vanderdonckt [1993], Smith & Mosier [1986]

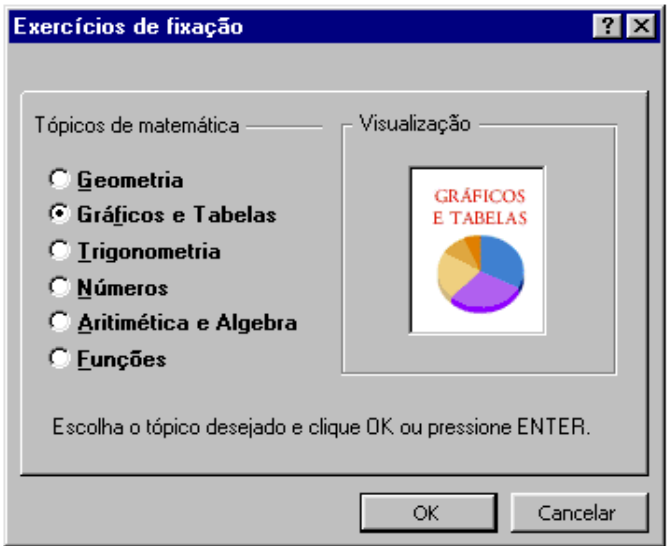


Fig. 154 – Duas etapas – Indicação e execução (OK ou ENTER).

3. Nas opções de preenchimento, o usuário deve comandar a navegação entre os campos.

   - COMENTÁRIO 1: Solicite aos usuários que explicitamente acionem uma tecla (por exemplo o "TAB") para mover o cursor de um campo de entrada de dados para o seguinte. O computador não fornecerá tal controle automaticamente (Figura 155).

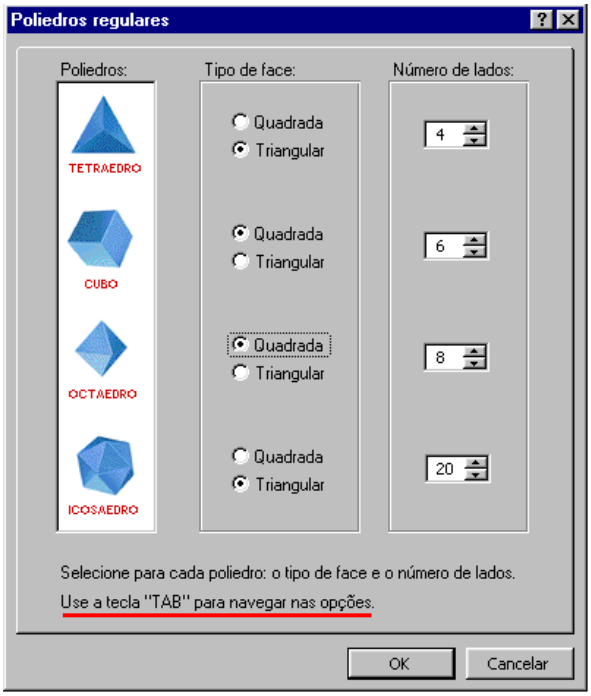


Fig. 155 – Uso da tecla de navegação – “TAB”.

- COMENTÁRIO 2: O automatismo pode ocasionar uma cascata de erros, por exemplo, um digitador hábil pode digitar uma série de itens sem olhar para a tela e, acidentalmente, "estourar" um dos campos de dados anteriores. Uma solução aceitável aqui seria projetar cada campo com um sinal sonoro para alertar o usuário, quando do estouro do campo. Isso permitiria o uso consistente da técnica de navegação pelo teclado para o movimento cuidadoso de um campo para outro.

REFERÊNCIA: Smith & Mosier [1986]

3.2. Controle do Usuário

Definição: O critério *Controle do Usuário* refere-se ao fato de que os usuários deveriam ter sempre no controle sobre o processamento do sistema (i.e., interromper, cancelar, suspender e continuar, avançar, retroceder, ou parar a apresentação). Cada ação possível deve ser antecipada e devem ser oferecidas opções apropriadas.

Justificativa(s): O controle sobre as interações favorece a aprendizagem e diminui a probabilidade de erros. Como consequência, o computador torna-se mais previsível. Quando os usuários têm controle sobre o sistema, podem melhor adequar a sequência da apresentação ao seu ritmo de aprendizagem, bem como o nível de complexidade dos exercícios e conteúdos propostos perante as suas dificuldades.

Recomendações

1. O usuário deve possuir controle sobre as ações dos botões de comando.

   - COMENTÁRIO 1: Para que ocorra uma boa aprendizagem é necessário que o usuário assuma o controle dos comandos, permitindo que o mesmo aplique o seu próprio ritmo durante a atividade desenvolvida.

   - COMENTÁRIO 2: Quando o usuário não possui o controle, comandos automatizados podem interromper a atividade, prejudicando o desenvolvimento natural da aprendizagem.

2. O usuário deve poder controlar a sequenciação dos conteúdos.

   - COMENTÁRIO 1: A seqüência de apresentação do conteúdo não deve ser única e rígida. Cada individuo possui sua forma própria de aprendizagem. Em função disso, o sistema deve permitir várias maneiras de percorrer o conteúdo de acordo com a vontade e/ou necessidade do usuário.

3. O usuário deve poder controlar o ritmo da apresentação.

   - COMENTÁRIO 1: O ritmo de aprendizagem de cada individuo varia bastante. O sistema deve permitir ao usuário controlar, de acordo com seu próprio ritmo, a apresentação para que ocorra a assimilação dos conteúdos.

4. O usuário deve poder controlar a apresentação por meio de opções de escolha entre diferentes níveis de complexidade.

   - COMENTÁRIO 1: Levando em conta os diferentes níveis de conhecimento dos usuários em potencial, o sistema deve conter diversos níveis de complexidade e/ou dificuldade na apresentação de um determinado conteúdo. Possibilitando que uma faixa maior de usuários utilizem o sistema.

5. O usuário deve poder interromper, retomar e reiniciar um diálogo sequencial a qualquer instante.

   - COMENTÁRIO 1: Se apropriado para a seqüência de controle, forneça opções de "INTERROMPER ou PAUSAR" e "RETOMAR ou CONTINUAR", que terão efeito de interromper e mais tarde retomar a seqüência de transação sem qualquer mudança para os dados entrados ou para a lógica dos controles da transação interrompida.

   - COMENTÁRIO 2: Funções desse tipo deveriam ser realizadas de forma rápida e fácil, o que sugere que elas sejam oferecidas por uma tecla de função.

- EXEMPLO 1: O usuário pode querer interromper, momentaneamente, uma sessão de exercícios para posteriormente, prosseguir retornando à ação (Figuras 156 e 157).

REFERÊNCIA: Smith & Mosier [1986]

6. O sistema deve possibilitar a interrupção ou o cancelamento da transação ou processo em andamento.

   - COMENTÁRIO 1: Se for necessário para o usuário controlar a interrupção ou o cancelamento de algum processo em andamento, o sistema deve apresentar as opções de "INTERROMPER" ou "CANCELAR", objetivando a parada ou o término da operação (Figura 158).

   - COMENTÁRIO 2: Neste caso o sistema não poderá retornar ao estado inicial da transação.

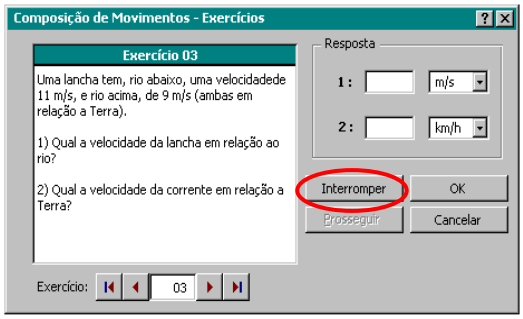


Fig. 156 – Botão interromper - O processo pode ser interrompido a qualquer momento.

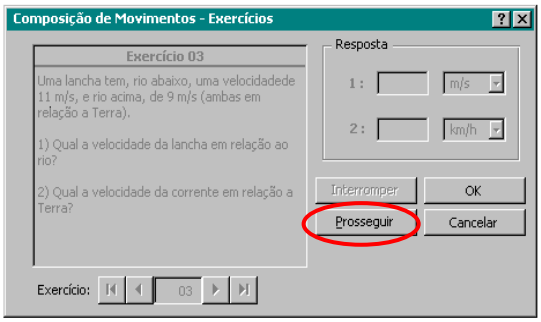


Fig. 157 – Botão prosseguir - O processo pode ser retomado a qualquer momento.

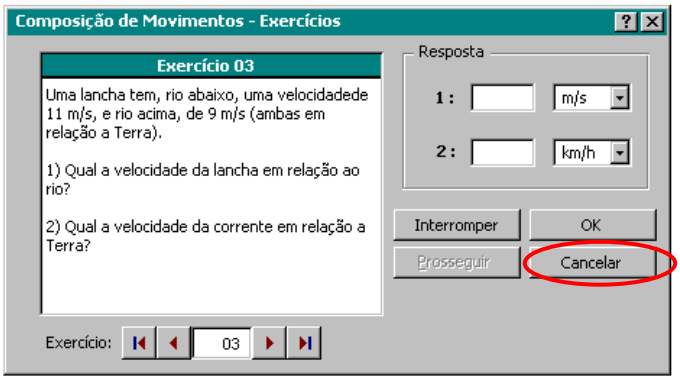


Fig. 158 – Botão Cancelar – O processo pode ser cancelado.

7. O sistema deve fornecer a opção CANCELAR com o objetivo de apagar qualquer mudança efetuada pelo usuário e trazer a tela para seu estado anterior (Figura 159).

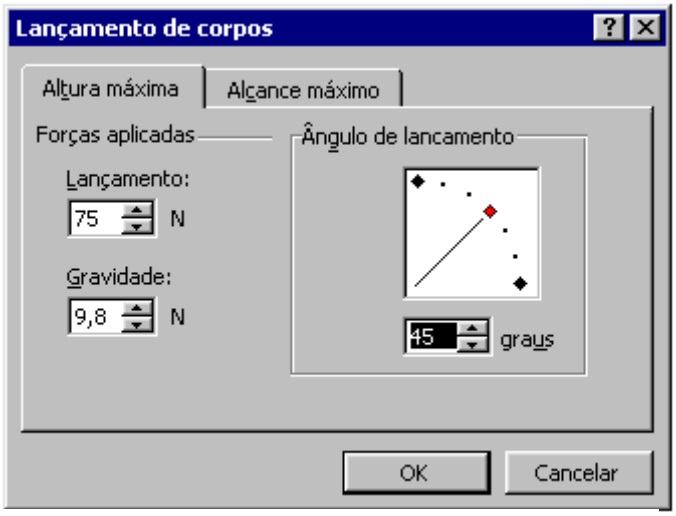


Fig. 159 – Botão Cancelar – Retorna a transação ao estado anterior.

- COMENTÁRIO 1: Se apropriado para o controle da seqüência, forneça uma opção de "REINICIAR" que terá o efeito de cancelar qualquer entrada que tenha sido feita em uma determinada seqüência de ações retornando ao início da transação.

- COMENTÁRIO 2: Quando entradas ou alterações de dados forem anuladas pela ação de "REINICIAR", exija uma ação do usuário de "CONFIRMAR".

- EXEMPLO 1: Na seqüência de entrada de dados relacionados através de diversas telas encadeadas, a opção de "REINICIAR" deve apagar todos os dados já entrados e retornar para a primeira tela.

REFERÊNCIA: Smith & Mosier [1986]

8. Durante os períodos de bloqueio dos dispositivos de entrada, o sistema deve fornecer uma opção para interromper o processo que causou o bloqueio.

- COMENTÁRIO 1: Durante os períodos de bloqueio dos dispositivos de entrada, um meio auxiliar deve ser fornecido ao usuário, tal como uma tecla de função especial, para interromper o processo que causou o bloqueio.

- COMENTÁRIO 2: Essa capacidade de interrupção será

especialmente útil nas situações em que o usuário detecta o erro cometido e quer interromper uma transação desnecessária, agindo como um comando "DESFAZER" (Figura 160).

- COMENTÁRIO 3: Alternativamente, para algumas transações pode ser útil projetar essa interrupção como um comando "FINALIZAR", que encerra processos em andamento sem cancelá-los. Por exemplo, se um usuário solicitou ao computador rolar para o final de um longo arquivo apresentado, o usuário pode simplesmente desejar interromper num certo ponto, sem ter que necessariamente voltar ao início (Figura 161).

REFERÊNCIA: Smith & Mosier [1986]

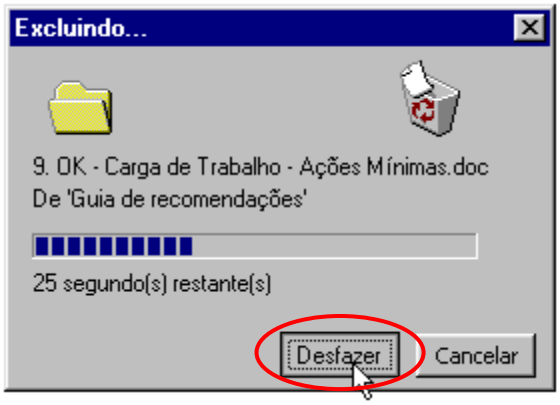


Fig. 160 – Detecção do erro e interrupção da transação.

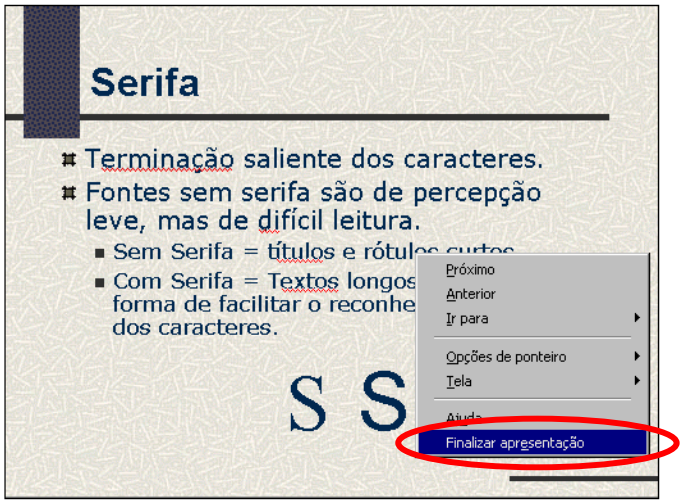


Fig. 161 – Finalizar apresentação – interrompe e permanece na tela atual.

4. Recursos de Apoio à Compreensão dos Conteúdos

Definição: Este critério refere-se ao apoio fornecido pelo *software* para auxiliar a compreensão dos conteúdos pedagógicos. A utilização de recursos multimídia, recursos motivacionais e recursos de verificação da aprendizagem contribuem para este fim.

*Justificativ*a: O *software* deve possuir recursos que auxiliem no processo de aquisição de um determinado conhecimento. Deve promover situações estimulantes para o aluno, não apenas despertando a sua atenção, mas mantendo-a ao longo da sua interação. Para que este fim seja atingido, os conteúdos pedagógicos apresentados pelo *software* educacional devem ser claros, consistentes e compreensíveis. Os recursos multimídia e os recursos motivacionais devem provocar o interesse pelo assunto ao mesmo tempo em que facilitam a situação de ensino/aprendizagem.

Recomendações

- O software deve possuir recursos motivacionais para despertar e manter a atenção do usuário ao longo de sua interação.

  - COMENTÁRIO 1: O sistema deve apresentar estímulos capazes de manter a atenção do usuário durante todo o tempo de sua interação de forma confortável e estimulante, contribuindo para fixação dos conteúdos estudados.

1. Os recursos motivacionais utilizados devem permanecer interessantes ao longo do tempo, sem causar aborrecimento através de repetições constantes.

   - COMENTÁRIO 1: Caso os recursos motivacionais utilizados apresentem algum tipo de movimentação ou efeito visual, deve-se observar para que o mesmo não seja repetitivo, evitando causar cansaço ou aborrecimento ao usuário.

2. Recursos multimídia devem ser utilizados de maneira moderada, sem provocar a distração do aluno ao principal foco na tela.

   - COMENTÁRIO 1: Ao utilizar a multimídia com recurso motivacional, o projetista deve faze-lo de maneira cuidadosa, sem provocar o desvio de atenção do foco principal ou tomar para si a atenção do usuário, causando diminuição ou perda no processo de aprendizagem.

   - COMENTÁRIO 2: O uso da multimídia não deve ser feito de maneira aleatória e sem critério, caso contrário o excesso pode causar cansaço e aborrecimento ao usuário.

3. Recursos sonoros devem ser bem explorados, e utilizados de forma pertinente.

   - COMENTÁRIO 1: A música deve ser utilizada quando se deseja determinar o clima para o ambiente ou para enfatizar as emoções.

   - COMENTÁRIO 2: Os efeitos sonoros devem ser utilizados com propósitos específicos de acordo com a necessidade da apresentação.

   - COMENTÁRIO 3: A narrativa deve ser utilizada de acordo com o público alvo, levando-se em conta as características pessoais como: cor, raça, sexo, etc.

REFERÊNCIA: Wolfman [1994]

4. Os recursos sonoros empregados devem contribuir para a motivação e compreensão dos conteúdos.

   - COMENTÁRIO 1: Os recursos sonoros devem ser utilizados de forma a proporcionar a facilitação da assimilação dos conteúdos apresentados.

   - COMENTÁRIO 2: Se possível, devem funcionar como recurso para memorização de conteúdos, por exemplo, a narração de um fato histórico.

5. Imagens, desenhos, gráficos, etc. devem ser utilizados com

pertinência e devem contribuir para a motivação e compreensão dos conteúdos.

- COMENTÁRIO 1: Os recursos gráficos devem ajudar no processo de aprendizagem e compreensão dos conteúdos, evitando desviar a atenção do usuário ou tornar o aprendizado de difícil assimilação (Figuras 162 e 163).

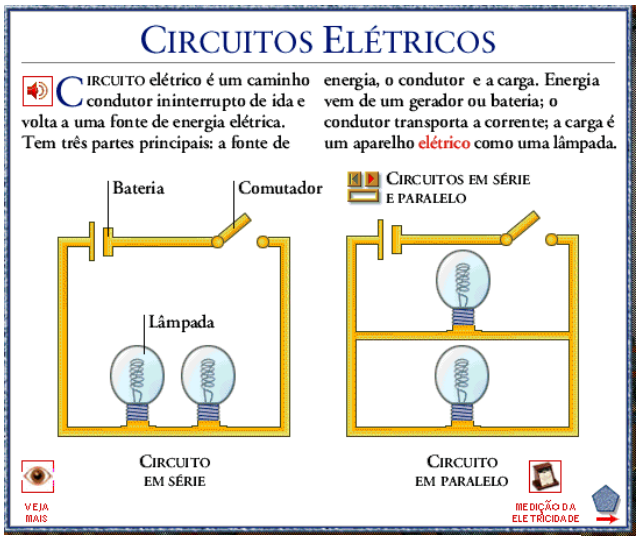


Fig. 162 – Exemplo positivo – Circuitos elétricos.

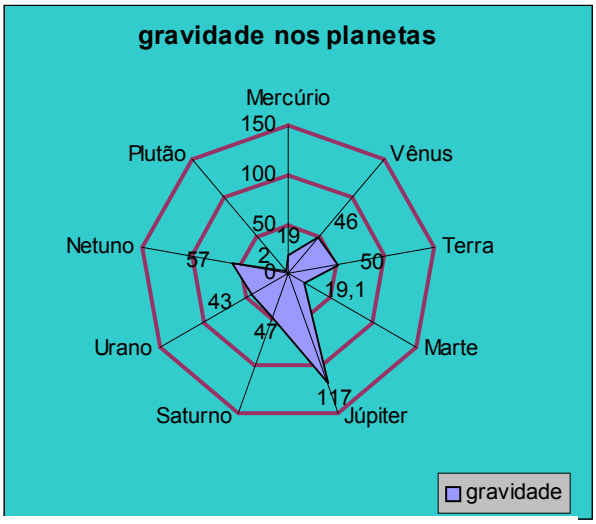


Fig. 163 – Exemplo negativo – Gráfico complexo.

6. Animações devem ser utilizadas com pertinência e devem contribuir para a motivação e compreensão dos conteúdos.

   - COMENTÁRIO 1: O projetista deve utilizar animações que tenham relação com o conteúdo apresentado. Animações sem critérios podem desviar atenção ou causar desinteresse do usuário.

7. Exercícios de aplicação e resolução de problemas devem ser utilizados com pertinência e devem contribuir para a motivação e compreensão dos conteúdos.

   - COMENTÁRIO 1: Os exercícios e as resoluções de problemas podem utilizar recursos multimídia desde que contribuam para o processo de aprendizagem dos conteúdos.

8. Os exercícios de aplicação e soluções de problemas devem evitar aborrecimento, constrangimento ou desânimo por parte do usuário.

   - COMENTÁRIO 1: O projetista deve considerar os diversos níveis de usuários, propondo exercícios com graus de dificuldade de acordo com a característica de cada faixa proporcionando estímulo a aprendizagem.

9. Os exercícios de aplicação e soluções de problemas devem evitar a sensação de frustração, causada por dificuldade na utilização do programa.

   - COMENTÁRIO 1: Os exercícios de aplicação e soluções de problemas devem apresentar grau de operacionabilidade que permita fácil compreensão e manipulação de seus dados, evitando que sua utilização represente um aumento de dificuldade na resolução da tarefa.

10. Jogos são utilizados com pertinência e devem contribuir para a motivação e compreensão dos conteúdos.

    - COMENTÁRIO 1: Os jogos devem ser utilizados sempre que proporcionarem um aumento na capacidade de assimilação de conteúdo através de estímulos de determinados tipos de aptidões como: memorização, estratégia, comparação, rapidez de raciocínio, etc.

11. Simulações devem ser utilizadas com pertinência e devem contribuir para a motivação e compreensão dos conteúdos.

    - COMENTÁRIO 1: As simulações devem ser utilizadas para demonstração de fenômenos que são difíceis ou impossíveis de serem realizados experimentalmente.

    - COMENTÁRIO 2: As simulações permitem uma maior observação dos fenômenos envolvidos aumentando a motivação e compreensão dos conteúdos.

12. Exercícios de criatividade devem ser utilizados com pertinência e devem

contribuir para a motivação e compreensão dos conteúdos.

- COMENTÁRIO 1: Os exercícios de criatividade devem estimular a capacidade de gerar novas associações como forma de compreensão de conteúdos aumentando os níveis de aprendizagem.

13. Diálogos devem ser utilizados ao longo do software com o objetivo de apoiar e verificar a compreensão dos conteúdos.

    - COMENTÁRIO 1: o diálogo deve permitir rever, cancelar, copiar e alterar qualquer item, possibilitando ao usuário a capacidade de controlar e verificar a aprendizagem dos conteúdos.

14. O usuário deve ter controle sobre a ordem de apresentação e sequenciação das informações.

    - COMENTÁRIO 1: A apresentação dos conteúdos não deve seguir uma seqüência fixa, o usuário deve ter o controle sobre a apresentação.

    - COMENTÁRIO 2: As apresentações retilíneas podem causar desinteresse ou cansaço no usuário.

15. O software deve estimular a imaginação do usuário através de um contexto ou situação que pode ser usada para auxiliar a aprendizagem.

    - COMENTÁRIO 1: O software pode apresentar um caso para estudo, observação e/ou interação de acordo com o conteúdo a ser assimilado, proporcionando maior realismo no processo de aprendizagem.

16. A apresentação das informações deve possuir geração aleatória, variando estímulos textuais, visuais e ou sonoros.

    - COMENTÁRIO 1: As informações devem ser apresentadas de diversas formas com o objetivo de estimular tanto a visão com a audição aumentando a assimilação de conteúdos.

    - COMENTÁRIO 2: Entre os estímulos sensoriais, a captação por estímulos auditivos e visuais representa, respectivamente, 11% e 83%.

17. O software deve oferecer a possibilidade de consulta a outras referências bibliográficas sobre o tema em estudo, tais como livros e outros materiais instrucionais.

    - COMENTÁRIO 1: O software deve fazer referências bibliográficas e de meios eletrônicos disponíveis que possam contribuir na assimilação e aprendizagem dos conteúdos apresentados.

5. Adaptabilidade

Definição: A *adaptabilidade* de um sistema diz respeito à sua capacidade de reagir conforme o contexto e conforme as necessidades e preferências do usuário. Dois sub-critérios compõem a adaptabilidade: a *flexibilidade* e a *consideração da experiência do usuário*.

*Justificativ*a: Um *software* educacional não pode atender ao mesmo tempo a todo o seu potencial público alvo, devido às diferenças individuais de cada um. Para que a interface de um sistema não tenha efeitos negativos, esta deve adaptar-se ao contexto dos usuários.
Quanto mais variadas forem as maneiras de realizar uma tarefa, maiores são as hipóteses do usuário escolher e dominar uma delas no curso da sua aprendizagem. Deste modo, deve-se fornecer procedimentos, opções, comandos diferentes, entre os quais os diferentes usuários possam escolher a fim de alcançarem um mesmo objetivo.

Sub-critérios – Adaptabilidade

5.1. Flexibilidade

Definição: A *flexibilidade* refere-se aos meios colocados à disposição do usuário que lhe permite personalizar a interface a fim de levar em conta as exigências da tarefa, de suas estratégias ou hábitos de trabalho. Ela corresponde também ao número das diferentes maneiras à disposição do usuário para alcançar um certo objetivo. Em outros termos, trata-se da capacidade da interface em se adaptar às variadas ações do usuário.

Justificativa: Quanto mais formas existirem para efetuar uma tarefa, maiores serão as hipóteses do usuário poder escolher e dominar uma delas no curso de sua aprendizagem.

Tratando-se de um *software* educacional, o sistema deve possuir recursos que permitam ajustar o nível de complexidade na apresentação da informação. A tarefa de aprendizagem varia de indivíduo para indivíduo. Ao passo em que certos alunos aprendem determinados conceitos rapidamente, outros podem levar um tempo muito maior. O *software* educacional deve ser capaz de prever e acomodar as diferenças individuais de seus potenciais usuários.

Recomendações

1. O Software deve permitir a introdução de novos elementos, personalizando-o de modo a acomodar diferenças individuais (Figura 164).

   - COMENTÁRIO 1: Não se pode considerar o usuário como um ser estático, passivo, que o sistema deve modelar ou dirigir. Ele deve poder criar ou escolher seus próprios ícones como meio de apresentação, de início de diálogo.

2. O software deve conter a opção de seleção de entrada para níveis intermédios de dificuldade (Figura 165).

3. O sistema deve fornecer meios para que o usuário tenha total controle sobre a sequência de apresentação das informações (Figura 166).

4. O sistema deve propor formas variadas de apresentação das mesmas informações a diferentes tipos de usuário.

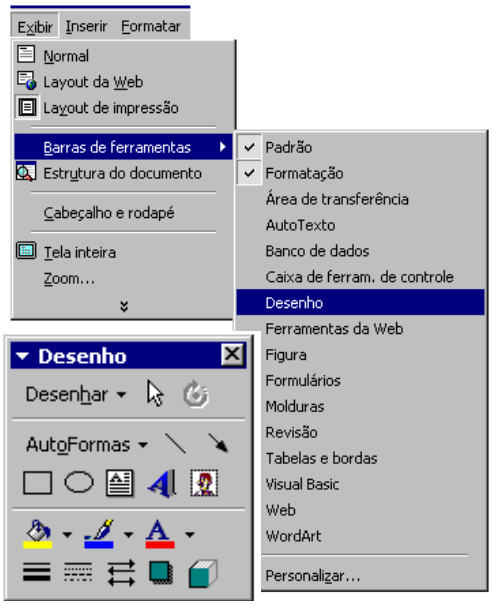


Fig. 164 – Introdução da barra de ferramenta Desenho.

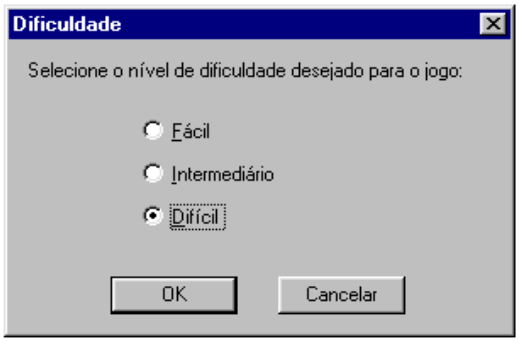


Fig. 165 – Seleção para nível de dificuldade.

Fig. 166 – Menu opcional – O usuário escolhe a seqüência de apresentação.

- COMENTÁRIO 1: Um usuário experimentado é capaz de compreender uma tela de apresentação complexa mais facilmente que um usuário novato. Em conseqüência, os projetistas devem oferecer uma variedade de apresentações diferentes. Trata-se de um compromisso entre as dificuldades de aprendizagem do sistema versus a simplicidade de concepção (Figuras 167 e 168).

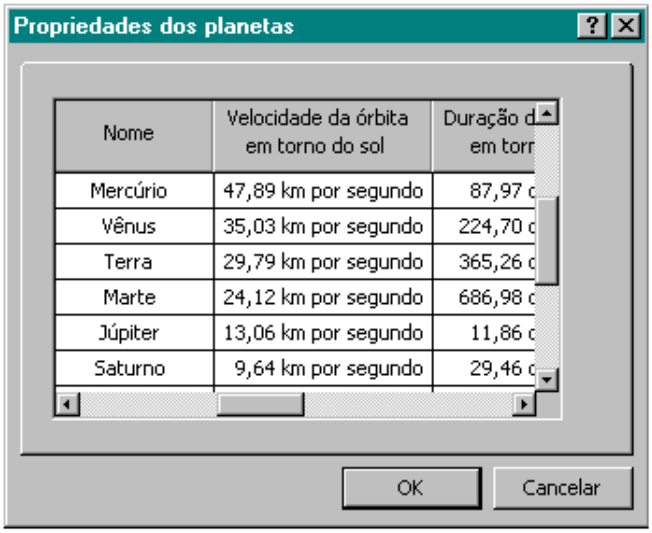


Fig. 167- Apresentação complexa - Tela para usuário experiente.

5. O sistema deve fornecer a possibilidade de desativar temporariamente a apresentação de certas janelas.

- COMENTÁRIO 1: As definições dos dados passíveis de apresentações e feitas originalmente para um propósito podem não ser adequadas para outro. Quando as solicitações da tarefa mudam rapidamente, pode ser mais eficiente suprimir, temporariamente, a apresentação de categorias de dados não necessárias, do que regenerar uma apresentação com diferentes critérios de apresentação. Nesses casos, informe ao usuário sobre quais foram os dados suprimidos e forneça aos usuários algum meio para que, rapidamente, restaurem a apresentação em sua forma completa e, originalmente, gerada.

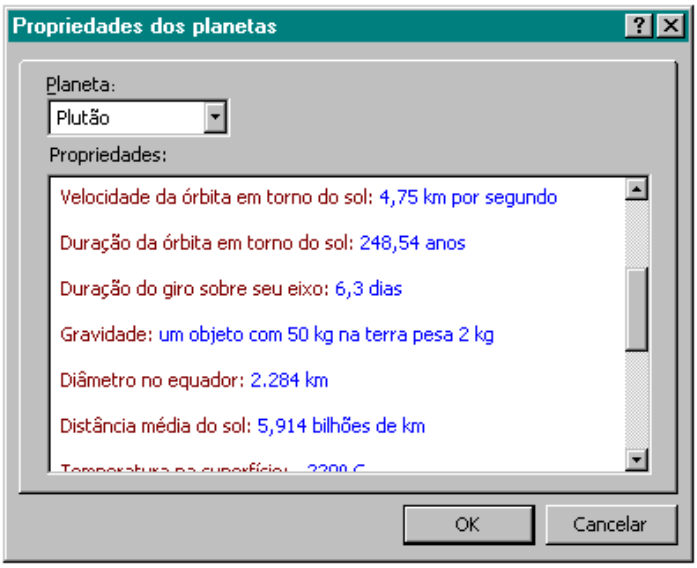


Fig. 168- Apresentação simplificada - Tela para usuário novato.

- COMENTÁRIO 2: Em algumas aplicações, poderia ser desejável restaurar os dados suprimidos automaticamente, após expiração de um tempo pré-determinado, do que contar com a memória do usuário para fazê-lo (Figuras 169 e 170).

6. Os usuários devem ter a possibilidade de modificar ou eliminar itens irrelevantes das janelas (Figura 171).

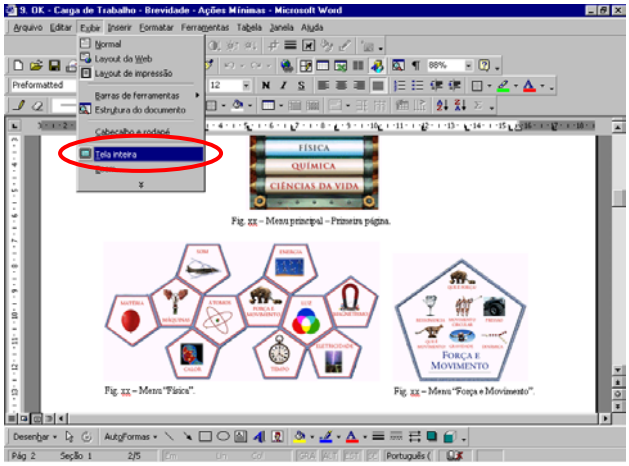


Fig. 169 – Menu Exibir, Tela inteira – Supressão temporária da janela original.

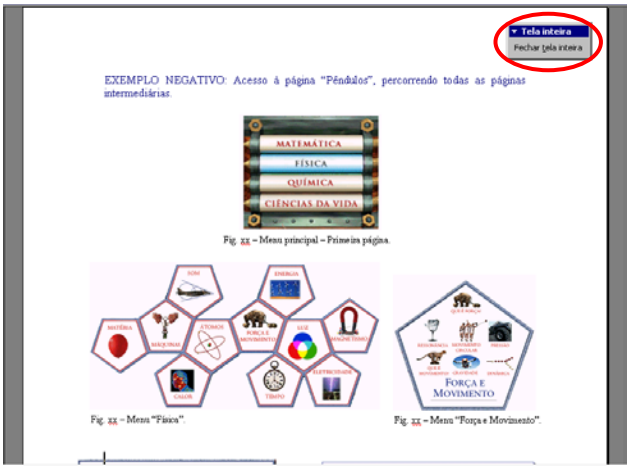


Fig. 170 – Tela inteira, temporária – Opção para restaurar a janela original.

Fig. 171 – Configuração de barra de ferramentas – Incluir e excluir opções.

7. O sistema deve permitir que se defina, mude ou retire os valores definidos por *default*, permitindo sua alteração e personalização (Figura 172).

   - COMENTÁRIO 1: Quando os valores por default não são previamente conhecidos, o sistema deve permitir que o usuário defina, mude ou suprima valores.

   - COMENTÁRIO 2: Apresente valores default para dados de tal forma que os usuários possam revê-los e confirmá-los para o processamento computacional.

8. O usuário deve ter a possibilidade de modificar a ordem e a sequência de entrada de dados, adaptando-a de acordo com sua ordem de preferência (Figura 173).

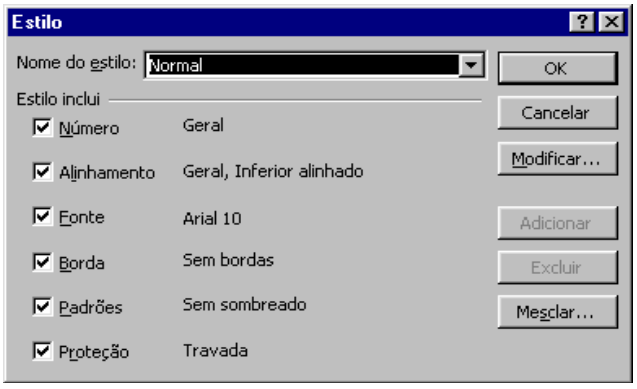


Fig. 172 – Configuração de Estilo – Alteração de valores.

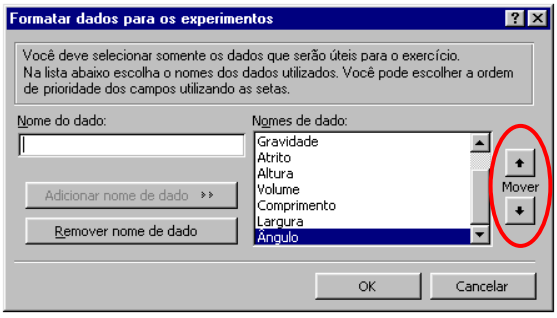


Fig. 173 – Formulário de configuração – Alteração de ordem e seqüência de dados.

9. Quando o formato de um texto não puder ser previsto com antecedência, o sistema deve proporcionar meios para definir e salvar os formatos que ele possa precisar (Figuras 174 e 175).

10. Deve ser permitido ao usuário definir os nomes dos campos de dados que ele (a) necessite criar (Figura 176).

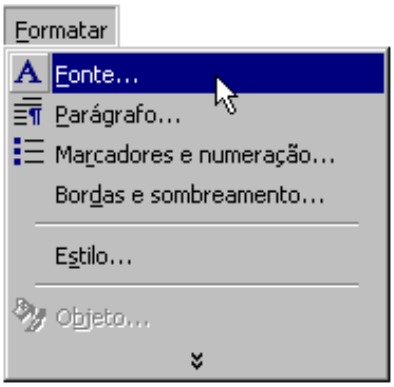


Fig. 174 – Menu Formatar – Opção Fonte.

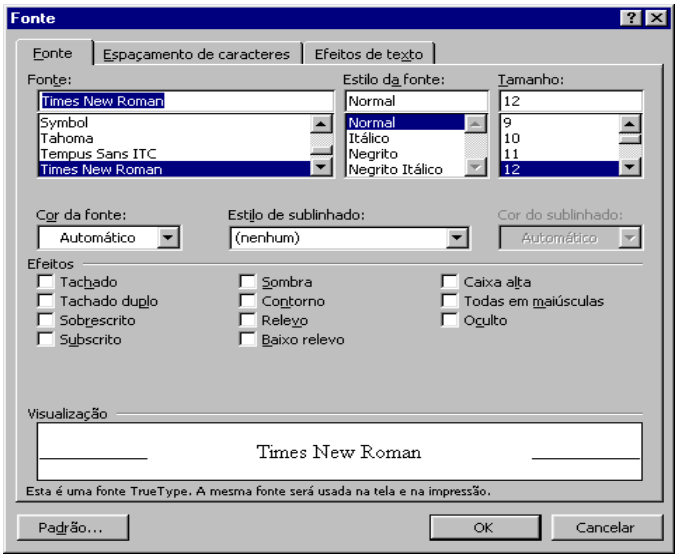


Fig. 175 – Fonte – Configurações e definições.

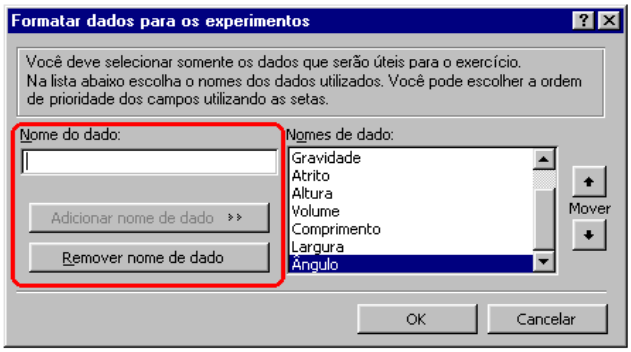


Fig. 176 – Formulário de configuração – Criação de nomes dos novos campos.

11. Deve ser permitido ao usuário, personalizar o diálogo, através da definição de macros (Figuras .177 e 178)

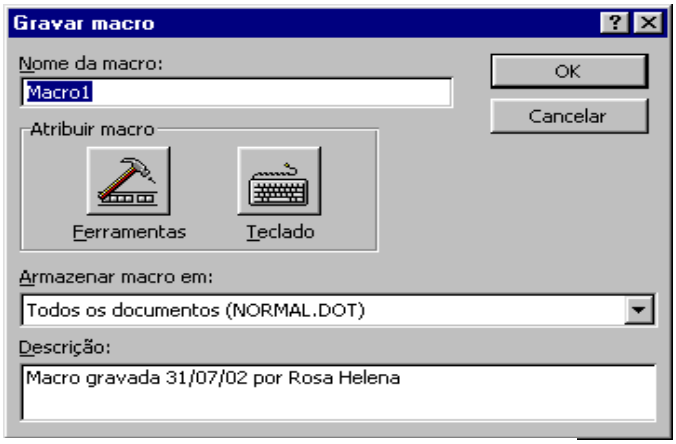


Fig. 177 – Gravação de macros.

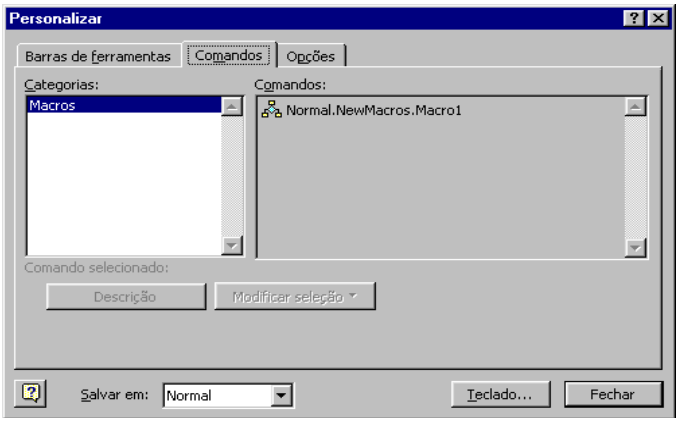


Fig. 178 – Personalização do diálogo - Definição de macros.

5.2. Consideração da Experiência do Usuário

Definição: A *consideração da experiência do usuário* diz respeito aos meios implementados que permitem que o sistema respeite os níveis de experiência individuais.

Justificativa: O grau de experiência e especialização dos usuários na interação com um dado sistema pode variar consoante a sua utilização continuada, ou devido aos longos períodos de não utilização dos sistemas. A interface deve, neste sentido, ser concebida para lidar com as variações de nível de experiência. Usuários experientes não têm as mesmas necessidades informacionais que os novatos. Os conteúdos teóricos são exemplos disto. Sua apresentação deve ser organizada de forma a permitir que os mais experientes possam avançar na apresentação sem ter que visualizar conceitos elementares. Na interface, nem todos os comandos ou opções precisam ser visíveis o tempo todo. Diálogos de iniciativa do computador podem entediar e diminuir o rendimento dos mais experientes. Ao contrário, os atalhos podem-lhes permitir rápido acesso às funções do sistema. Pode-se fornecer aos inexperientes diálogos fortemente conduzidos, podendo, se necessário, fazer-se a progressão mesmo passo a passo.
Em suma, devem ser previstos meios diferenciados para lidar com as diferenças individuais, permitindo que o usuário adapte o seu estilo de interação, mediante a sua experiência.

Recomendações

1. A sequência da apresentação dos conceitos deve evoluir significativamente em grau de complexidade.

   - COMENTÁRIO 1: A apresentação dos conceitos deve variar de acordo com o grau de experiência do usuário, aumentando sua complexidade tanto nos conceitos como nos exercícios.

2. Usuários mais experientes devem poder acessar direto aos módulos mais avançados.

   - COMENTÁRIO 1: O sistema deve permitir que os usuários acessem os módulos de acordo com seu grau de experiência, evitando que os experientes tenha que, necessariamente, interagir com módulos básicos.

   - COMENTÁRIO 2: Usuários experientes podem perder o interesse no sistema se forem forçados a acessar os módulos básicos sempre que buscarem os módulos avançados.

3. O software deve permitir que o aluno possa retornar novamente no exato nível em que atingiu no seu último acesso.

   - COMENTÁRIO 1: O sistema deve ser capaz de armazenar a informação referente ao último acesso feito pelo usuário, permitindo que o mesmo, ao acessar novamente o sistema, retorne ao mesmo nível.

   - COMENTÁRIO 2: Os casos em que o sistema não armazena o último acesso do aluno, forçando seu retorno ao nível inicial pode causar desestímulo e, conseqüentemente, desinteresse pela atividade.

4. O sistema deve permitir efetuar alterações em suas estruturas de modo a contemplar a experiência do usuário.

   - COMENTÁRIO 1: O sistema deve prever sua utilização por uma grande variedade de usuários, concebendo um diálogo em diversos níveis com referência às apresentações, às mensagens de erro e à linguagem de comando.

- COMENTÁRIO 2: O sistema deve apresentar níveis de configuração que possibilite alterações em seu ambiente de acordo com as características de cada usuário, levando em consideração a experiência deste (Figura 179).

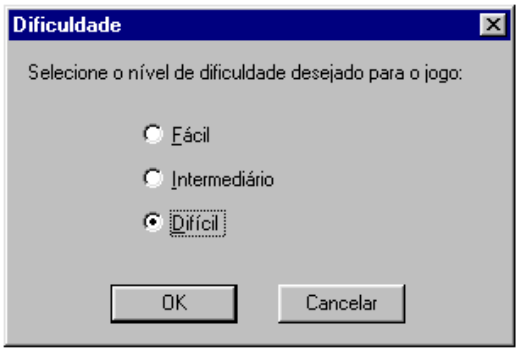


Fig. 179 – Seleção para nível de dificuldade.

5. O software deve permitir flexibilização na resolução dos problemas propostos, não requerendo do aluno que o mesmo complete tarefas básicas antes que lhe seja permitido continuar no programa.

   - COMENTÁRIO 1: O sistema deve permitir que usuários possam escolher o nível de dificuldade de resolução dos problemas propostos, sem que sejam forçados a realizar tarefas básicas iniciais.

   - COMENTÁRIO 2: Forçar o usuário a refazer tarefas básicas ou iniciais sem a solicitação do mesmo, tornará sua tarefa desagradável, comprometendo a utilização do aplicativo.

6. O sistema deve prever a escolha de entradas simples ou múltiplas conforme a experiência do usuário.

   - COMENTÁRIO 1: O sistema deve permitir a digitação de vários comandos antes de uma confirmação do usuário experiente.

   - COMENTÁRIO 2: Essa prática permite que o usuário experiente abrevie as seqüências de comandos. O agrupamento de comandos reduz o trabalho da memória do usuário. Os comandos realizados em grupos são mais eficazes do que a entrada separada de cada um deles, especialmente em sistemas multi-usuários.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

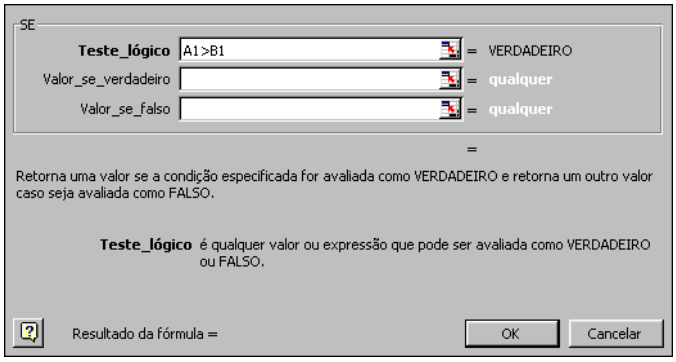


Fig. 180 – Função lógica SE – 1º passo: Teste_lógico.

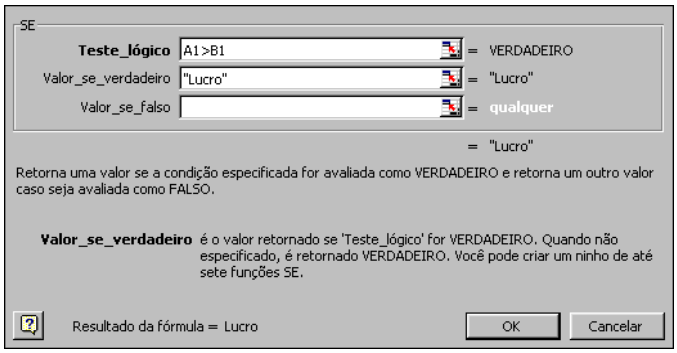


Fig. 181 – Função lógica SE – 2º passo: Valor_se_verdadeiro.

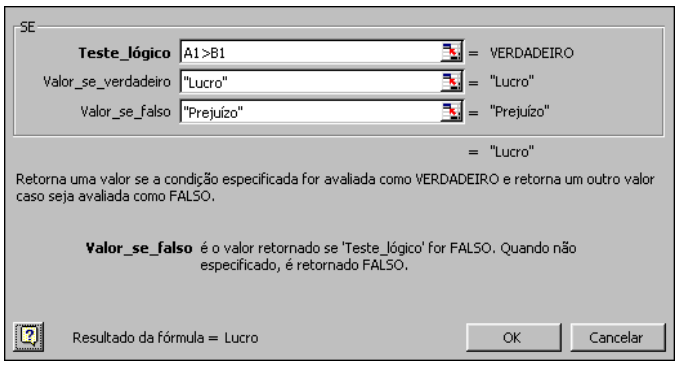


Fig. 182 – Função lógica SE – 3º passo: Valor_se_falso.

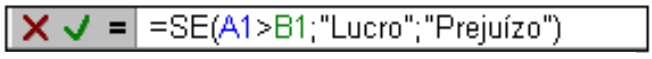


Fig. 183 – Função lógica SE – Digitada na barra de fórmula.

7. O sistema deve fornecer um tutorial passo a passo para os novatos e a entrada de comandos mais complexos para os mais experientes.

   - COMENTÁRIO 1: O sistema deve garantir que os modos do controle de seqüência sejam compatíveis com as capacidades dos usuários, permitindo ações passo a passo simples pelos principiantes, mas permitindo entradas de comando mais complexas pelos usuários experientes (Figuras 180, 181, 182 e 183).

   - COMENTÁRIO 2: A maioria dos sistemas terá usuários com níveis variados de experiência. Qualquer usuário particular pode se tornar mais hábil com aumento da experiência, ou talvez menos experiente depois

de longo período de desuso. Para acomodar os usuários com vários níveis de experiência será necessário uma mistura de diferentes tipos de diálogo, com alguns meios suaves de transição de um modo de diálogo para outro.

- COMENTÁRIO 3: As tarefas mais complexas devem ser reservadas aos usuários experimentados. Estas últimas devem ser acessíveis, mas sem desviar a atenção dos usuários não experimentados (Figura 184).

REFERÊNCIA: Smith & Mosier [1986]

8. O sistema deve permitir que usuários experientes contornem uma série de seleções por menu através da especificação de comandos e/ou atalhos de teclado.

   - COMENTÁRIO 1: Preveja atalhos de forma a permitir que usuários experientes contornem uma série de seleções por menu através da especificação de comandos ou de atalhos de teclado (Figura 185).

REFERÊNCIA: ISO 9241 parte 14 [1995]

9. Deve ser fornecida a possibilidade de escolha de nível de detalhe das mensagens de erro em função do nível de conhecimento.

   - COMENTÁRIO 1: O sistema deve dispor de recursos que permitam qualificar o tipo de mensagem de erro, variando de acordo o a experiência do usuário.

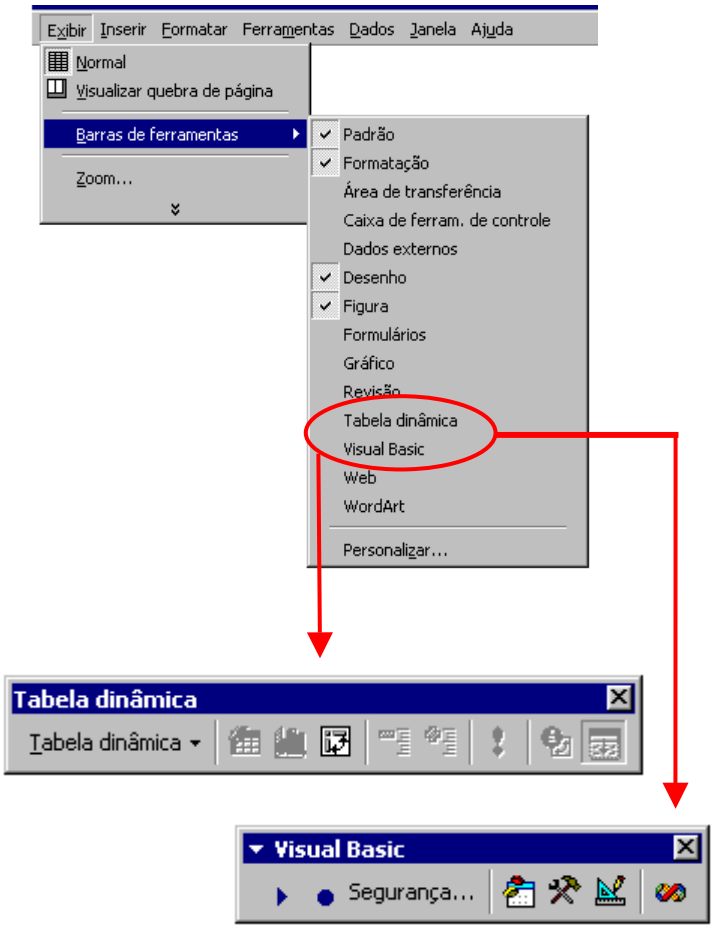


Fig. 184 – Tarefas complexas acessíveis para usuários experientes – Ativação por barra de ferramenta.

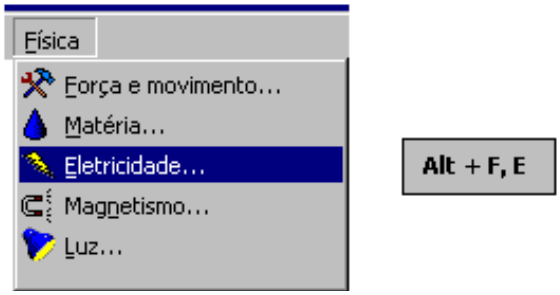


Fig. 185 – Menu Física, opção Eletricidade – Teclas de atalho = Alt + F, E.

- COMENTÁRIO 2: Para usuários experientes, a mensagem apresentada deve conter a informação do tipo de erro, além dos possíveis passos para a solução do referido problema (Figura 186).

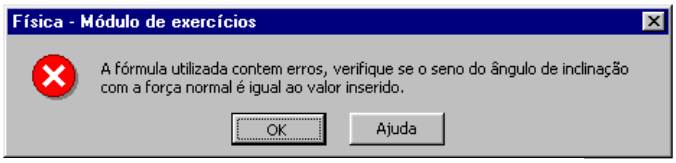


Fig. 186 – Mensagem de erro – Usuário experiente.

- COMENTÁRIO 3: Para usuários inexperientes, a mensagem apresentada deve conter a informação do tipo de erro e a recomendação de acionar a pessoa qualificada para a solução do referido problema (Figura 187).

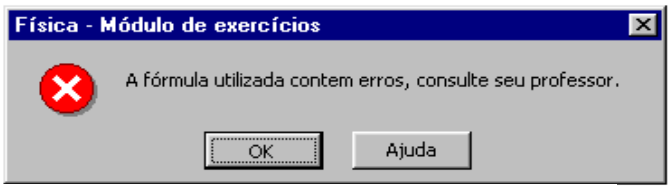


Fig. 187 – Mensagem de erro – Usuário inexperiente.

10. O sistema deve oferecer a facilidade para que usuários de níveis de familiaridade diferentes possam facilmente adequar-se ao sistema. (através da disponibilização de teclas de atalho/aceleração).

- COMENTÁRIO 1: O sistema deve disponibilizar tanto menus básicos para usuários iniciantes, como recursos avançados de acesso rápido para usuários experientes, que podem ser na forma de teclas de atalho, teclas de função e/ou teclas de aceleração.

6. Gestão de erros

Definição: A *gestão de erros* diz respeito aos mecanismos que permitem evitar ou reduzir a ocorrência de erros e, quando eles ocorrem, que favoreçam sua correção. Os erros são aqui considerados como entrada de dados incorretos, entradas com formatos inadequados, entradas de comandos com sintaxes incorretas, etc. Considera-se erro também, as respostas inadequadas ao sistema mediante os recursos de verificação de aprendizagem dos conteúdos no *software* educacional. Três sub-critérios participam da manutenção dos erros: a *proteção contra os erro*s, a *qualidade das mensagens de erro* e a *correção dos erro*s.

Justificativa: As interrupções provocadas pelos erros têm consequências negativas sobre a atividade do usuário, atrasam e dificultam a aprendizagem. Geralmente, elas prolongam as transações e perturbam a planificação. Quanto menor for a probabilidade de erros, menos interrupções ocorrem, melhor é o desempenho e mais eficaz é o contexto de aprendizagem.

Sub-critérios – Gestão de Erros

6.1. Proteção Contra Erros

Definição: A *proteção contra os erros* diz respeito aos mecanismos empregados para detectar e prevenir os erros de entradas de dados, comandos, possíveis ações de consequências desastrosas e/ou não recuperáveis.

Justificativa: É preferível detectar os erros no momento da digitação do que no momento da validação. Isto pode evitar perturbações na planificação da tarefa.

Recomendações

1. O sistema deve permitir que o usuário interrompa uma operação indesejada, contornando esta operação.

   - COMENTÁRIO 1: Ao detectar o andamento de uma operação não desejada, o usuário deverá ter condições de causar sua interrupção podendo não executá-la (Figura 188).

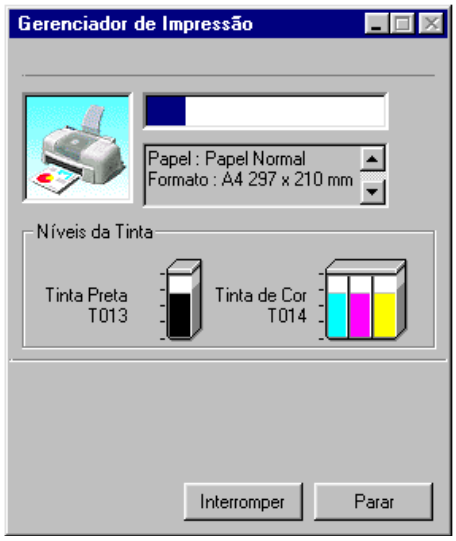


Fig. 188 – Interrupção de relatório impresso indevidamente.

2. O sistema deve permitir que o usuário possa voltar atrás, e deve informá-lo em caso de comandos que induzam a erro.

   - COMENTÁRIO 1: O sistema deve fornecer ao usuário recursos que lhe permitam retornar uma operação indevida, utilizando recursos do tipo "desfazer" ou "voltar" (Figuras 189 e 190).

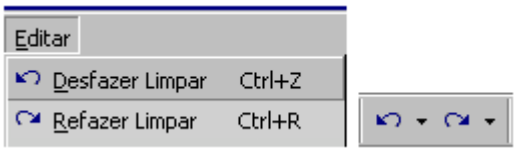


Fig. 189 – Comando Desfazer – Barra de menu e de ferramentas.

Fig. 190 – Comando Voltar – Barra de ferramentas.

3. Quando o usuário terminar uma secção e existir o risco de perda de dados, o sistema deve emitir uma mensagem que o avise deste fato, pedindo-lhe a confirmação do final da secção.

   - COMENTÁRIO 1: O usuário pode pensar que o trabalho foi completado e, no entanto, restarem ainda diálogos ou aplicações sendo executadas. Ao fechar sua sessão interativa ou ao se desconectar do sistema, ele deve ser avisado desse fato através de mensagens de aviso (Figura 191).

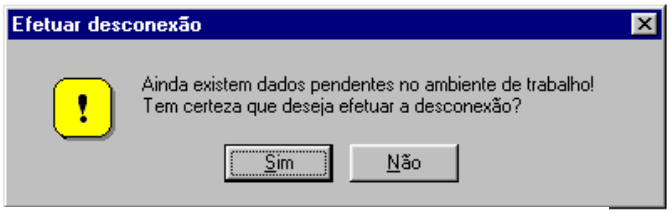


Fig. 191 – Mensagem de aviso - Dados pendentes.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

4. As apresentações que acompanham as entradas de dados devem estar protegidas, de modo que não possam ser modificadas as informações contidas nestes campos.

   - COMENTÁRIO 1: As apresentações têm como função conduzir o usuário na correta utilização do ambiente, devendo permanecer protegidas contra qualquer tipo de alteração (Figura 192).

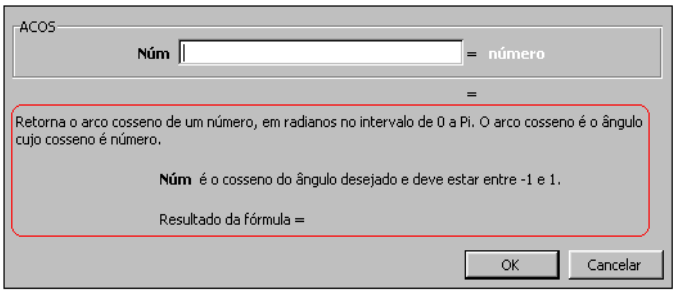


Fig. 192 – Apresentações de entrada de dados protegidas

5. Os títulos dos campos devem estar protegidos e impedidos de serem alterados pelo usuário.

   - COMENTÁRIO 1: Os títulos dos campos têm a função de representar, de maneira curta e simples, com exatidão o tipo de dado e ser utilizado. Seu contexto não pode ser alterado pelo usuário (Figuras 193).

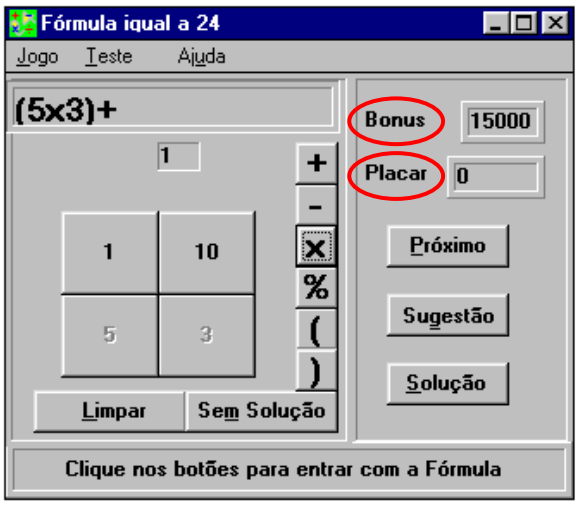


Fig. 193 – Título de campos – Não podem ser alterados.

6. O sistema deve emitir sinais sonoros quando ocorrem problemas na entrada de dados.

   - COMENTÁRIO 1: Durante a edição/entrada de dados, deve ser apresentado um sinal sonoro quando for necessário chamar a atenção do usuário para a tela.

   - COMENTÁRIO 2: Um digitador treinado inserindo um texto de uma cópia escrita, nem sempre estará olhando para a apresentação na tela, e pode não notar alguma indicação visual de erros ou mudanças, a menos que sejam acompanhadas por sinais sonoros.

- COMENTÁRIO 3: O sistema deve emitir sinal sonoro quando o usuário, ao inserir dados em um formulário, ultrapassar o limite de um campo, evitando que nenhum dado fique perdido.

- COMENTÁRIO 4: Deve-se observar que em um ambiente de trabalho em grupo, os sinais sonoros podem distrair outras pessoas e, mesmo, embaraçar o usuário, cujo erro está sendo assinalado. Nesse caso, deve ser permitido ao usuário desabilitar o sinal sonoro.

REFERÊNCIA: Smith & Mosier [1986]

7. Ao final de uma sessão de trabalho, antes de fechar o aplicativo, o sistema deve solicitar a opção salvar e informa sobre o risco de perda dos dados.

    - COMENTÁRIO 1: Quando o usuário solicitar o encerramento da atividade e o sistema detectar que a informação não foi salva, o aplicativo deve informar o ocorrido e possibilitar o salvamento da operação (Figura 194).

8. No caso de ocorrência de erros de digitação de um comando ou de dados, o sistema deve permitir que o usuário corrija somente a parte dos dados ou do comando que está errado.

    - COMENTÁRIO 1: O sistema deve permitir que a digitação de qualquer informação seja passível de edição, permitindo sua alteração total ou parcial para possíveis correções (Figura 195).

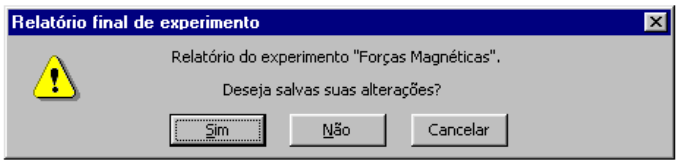


Fig. 194 – Aviso de pendências – Arquivo não salvo.

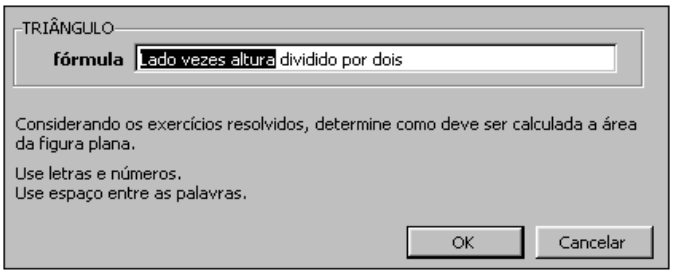


Fig. 195 – Erro de digitação – Parte do texto selecionada para correção.

9. Em toda ação destrutiva, os botões selecionados por *default* devem realizar a anulação dessa ação (Figuras 196 e 197).

   - COMENTÁRIO: Se na entrada de comandos, um dos botões for definido por default, deve-se ter certeza de que esse comando default protegerá o usuário contra a perda de seus dados, ou que, ao menos, não contribuirá para esse risco.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

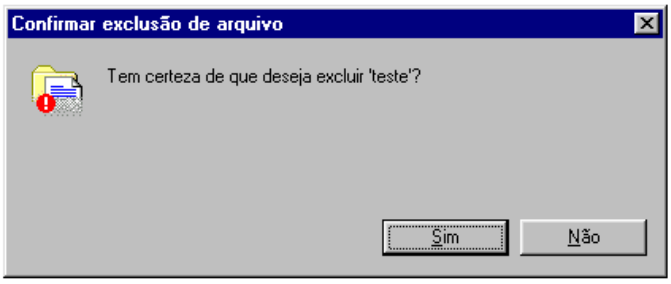


Fig. 196 – Exemplo negativo – Botão *default* = Sim.

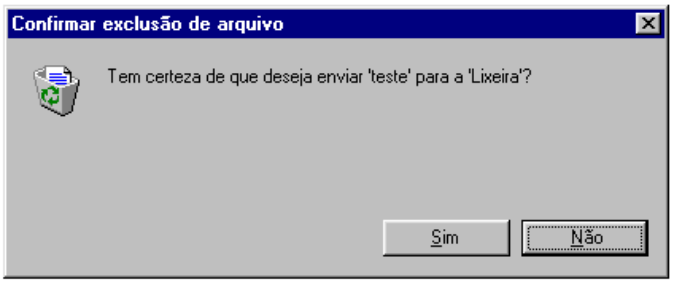


Fig. 197 – Exemplo positivo – Botão *default* = Não.

10. As teclas de funções perigosas e/ou rotineiras devem encontrar-se agrupadas e/ou separadas das demais no teclado.

    - COMENTÁRIO: As teclas associadas a funções potencialmente destrutivas devem ser fisicamente protegidas, seja através de uma localização segura no teclado (longe de teclas freqüentemente acionadas), seja pela combinação de teclas (Ctrl + Alt + Del)

    - COMENTÁRIO: As teclas de funções freqüentes devem estar localizadas em posições facilmente acessíveis.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

6.2. Qualidade das mensagens de erros

Definição: A qualidade das mensagens refere-se à pertinência, legibilidade e exatidão da informação dada ao usuário sobre a natureza do erro cometido (sintaxe, formato, etc.) e sobre as ações a executar para corrigi-lo.

Justificativa: A qualidade das mensagens favorece a aprendizagem do sistema indicando ao usuário a razão ou a natureza do erro cometido, o que ele fez de errado, o que ele deveria ter feito e o que ele deve fazer.

Recomendações

- Na ocorrência de erros durante a resolução dos exercícios, as mensagens de erro devem auxiliar e informar o usuário na superação do erro.

  - COMENTÁRIO 1: As mensagens de erro, além de informar sua ocorrência, devem indicar as possíveis soluções para o problema, auxiliando o usuário na operacionalização do sistema (Figura 198).

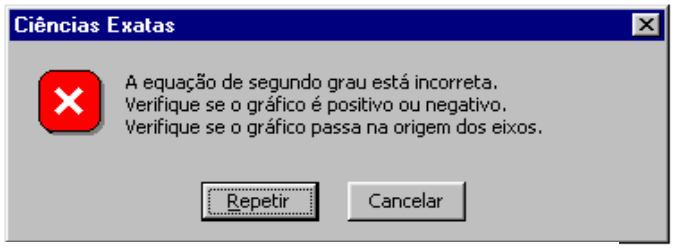


Fig. 198 – Mensagem de erro – Possíveis soluções.

1. O *feedback* das respostas às dificuldades deve ser encorajador e livre de conotação negativa.

   - COMENTÁRIO 1: As respostas devem apresentar conteúdo estimulante, sem conotações negativas que possam desencorajar o usuário a prosseguir na tarefa (Figuras 199 e 200).

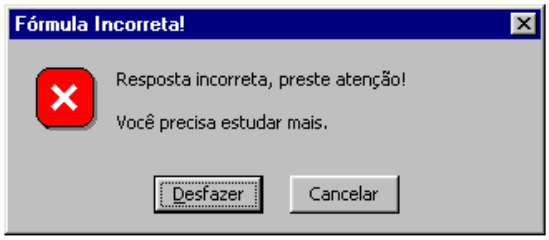


Fig. 199 – Exemplo negativo.

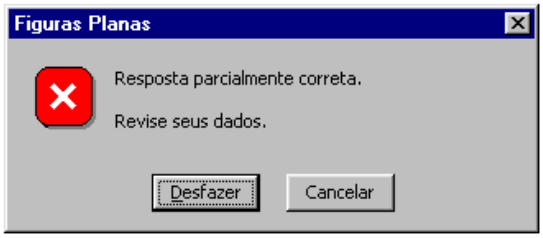


Fig. 200 – Exemplo positivo.

2. Perante uma dificuldade na resolução dos exercícios, o software deve evitar a monotonia oferecendo mensagens de erro variadas.

   - COMENTÁRIO 1: Caso o usuário reincida no erro, o sistema deve apresentar outra mensagem, contendo informações diferentes da anterior, proporcionando novo estímulo para a resolução (Figuras 201 e 202).

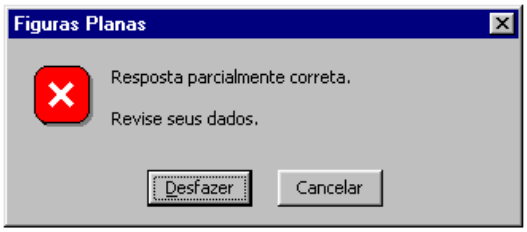


Fig. 201 – Mensagem inicial.

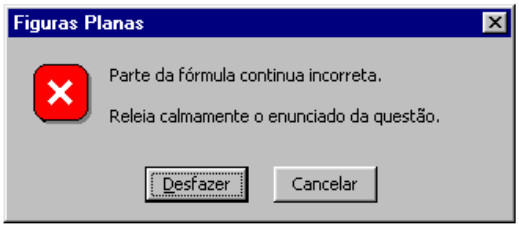


Fig. 202 – Mensagem seguinte.

3. As frases das mensagens de erro devem adotar um vocabulário neutro, não personalizado, não repreensivo e devem evitar o sentido de humor (Figura 203).

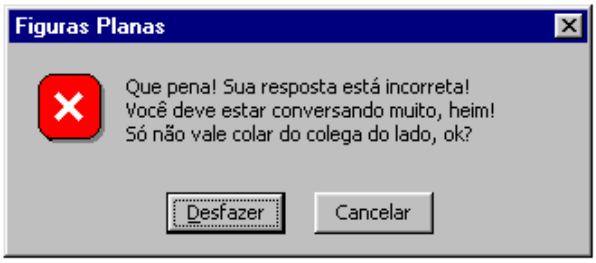


Fig. 203 – Exemplo negativo.

4. As frases das mensagens de erro devem ser curtas e construídas a partir de palavras curtas, significativas e de uso comum.

   - COMENTÁRIO 1: As mensagens de erro devem explicar os erros utilizando a linguagem do usuário (Figuras 204 e 205).

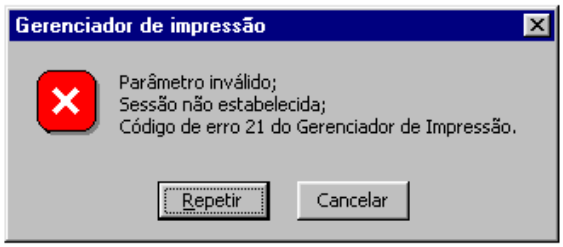


Fig. 204 – Exemplo negativo.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

5. As mensagens de erro devem ser neutras e polidas.

   - COMENTÁRIO 1: As mensagens de erro devem ser neutras, polidas e educadas, devem evitar qualquer terminologia hostil ou agressiva ao usuário, não devem julgá-lo, embaraçá-lo ou insultá-lo e não devem ser autoritárias ou humorísticas (Figuras 206 e 207).

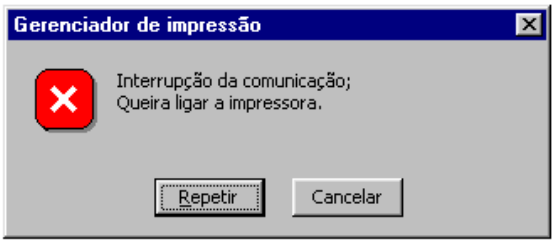


Fig. 205 – Exemplo positivo.

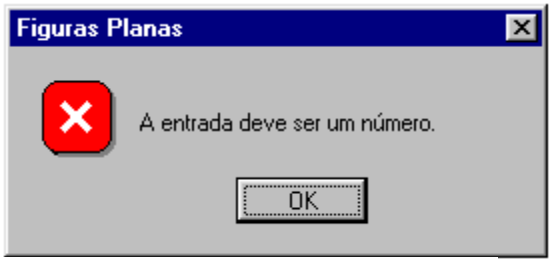


Fig. 206 – Exemplo positivo.

   - COMENTÁRIO 2: As mensagens de erro devem refletir uma visão consistente de que o computador é um instrumento com certas limitações, as quais o usuário deve considerar para fazer a ferramenta funcionar apropriadamente .

   - COMENTÁRIO 3: Se as mensagens de erro refletirem uma atitude do computador (ou do seu programador) de impor regras, ou determinar o que seja

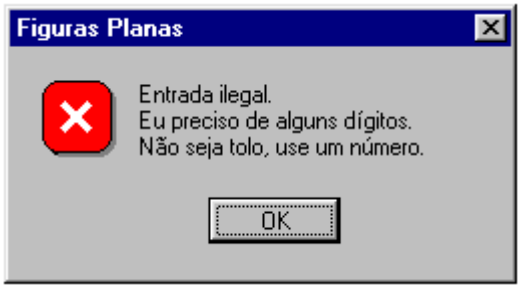


Fig. 207 – Exemplo negativo.

legal ou ilegal, o usuário poderá se sentir ofendido.

- COMENTÁRIO 4: Se as mensagens de erro refletirem personalização do computador, como se esse fosse um amigo, um usuário principiante e ingênuo pode ser mal conduzido a esperar habilidades humanas que a máquina não possui.

- COMENTÁRIO 5: Se as mensagens de erro fizerem uso de palavras com humor, qualquer brincadeirinha pode se tornar inconveniente com a repetição, e pode parecer intrusa no interesse de um usuário com relação à realização eficiente da tarefa.

- COMENTÁRIO 6: As mesmas considerações se aplicam ao vocabulário dos rótulos e a outros materiais instrucionais.

REFERÊNCIA: Bodart & Vanderdonckt [1993], Smith & Mosier [1986]

6. As mensagens de erro devem estar isentas de abreviaturas e/ou códigos gerados pelo sistema operacional.

   - COMENTÁRIO 1: Os termos das mensagens de erro não devem ser abreviados ou codificados, devendo ser diretamente compreensíveis.

   - COMENTÁRIO 2: Evite apresentar mensagens diretamente geradas pelos programas utilitários, pelo sistema operacional e pelo aplicativo. Traduza tais mensagens em linguagem natural antes de mostrá-las.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

7. As mensagens de erro devem estar orientadas à tarefa.

   - COMENTÁRIO 1: As mensagens de erro deve ser orientada à tarefa, utilizando termos e jargão técnico que sejam usualmente empregados nas tarefas dos usuários.

   - COMENTÁRIO 2: Os jargões podem ser úteis, se eles representarem o jargão do usuário e não o do projetista ou o do programador. O ponto de partida deve ser o conhecimento dos usuários, com a adaptação do projeto de interface ao seu vocabulário, em vez de forçá-los a aprender novas nomenclaturas.

REFERËNCIA: Smith & Mosier [1986]

8. As mensagens de erro devem ter seu conteúdo modificado quando na repetição imediata do mesmo erro pelo mesmo usuário.

   - COMENTÁRIO 1: Na ocorrência de um mesmo erro repetidas vezes pelo mesmo usuário, o sistema deve apresentar mensagens de erro com conteúdos diferentes com o objetivo de conduzi-lo até a solução.

   - COMENTÁRIO 2: Respostas com conteúdos diferentes podem ser baseadas na conduta incorreta do usuário durante a resolução e utilizadas para indicar as prováveis causas dos erros ou possíveis correções.

- COMENTÁRIO 3: As mensagens devem se adaptar de acordo com o erro, variando quando o erro ocorrer no mesmo ponto ou em pontos distintos da resolução.

9. O usuário deve poder escolher o nível de detalhe das mensagens de erro em função de seu nível de conhecimento (Figuras 208 e 209).

   - COMENTÁRIO 1: Para cada mensagem de erro, deve-se fornecer ao menos dois níveis de detalhe, no momento de sua apresentação.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

10. As mensagens de erro devem ajudar a resolver o problema do usuário, fornecendo com precisão o local e a causa específica ou provável do erro, bem como as ações que o usuário poderá realizar para corrigi-lo.

    - COMENTÁRIO 1: As mensagens de erro devem ajudar a resolver o problema do usuário, sendo instrutivas e informativas, fornecendo com precisão o local e a causa do erro da forma mais específica possível.

    - COMENTÁRIO 2: Se a causa do erro não for identificável, então a mensagem de erro deverá indicar as causas possíveis.

    - COMENTÁRIO 3: Na medida do possível as mensagens de erro devem indicar as ações que o usuário poderá realizar para corrigir o erro.

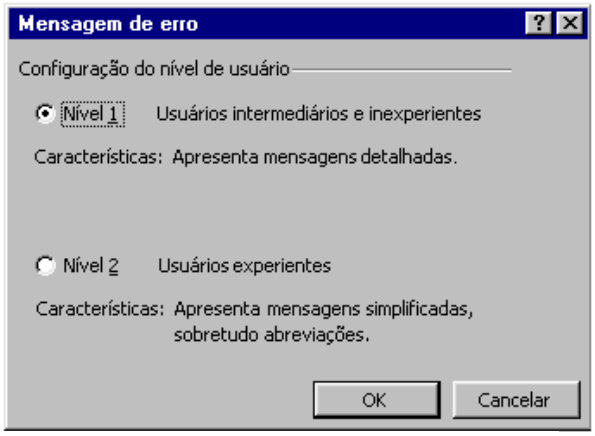


Fig. 208 – Exemplo positivo – Dois níveis.

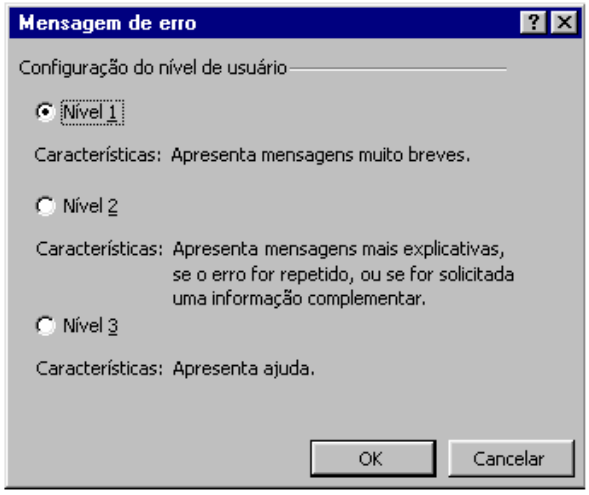


Fig. 209 – Exemplo positivo – Três níveis.

- COMENTÁRIO 4: Quando um usuário estiver em dificuldades, necessitando de mais informações do que a mensagem de erro fornece, pode-se oferecer uma função de ajuda auxiliar "*on line*", ou uma referência à documentação.

- COMENTÁRIO 5: Se nenhuma ajuda estiver disponível, o usuário pode se referir a uma lista codificada dos erros possíveis. Em tais circunstâncias, a mensagem de erro pode se resumir a um código que faça referência à documentação, tornando ágil sua identificação. Essa prática facilita o trabalho dos usuários experientes que já reconhecem diretamente os códigos de erros.

- COMENTÁRIO 6: Freqüentemente, o usuário reconhece ter cometido um erro e a mensagem só vem confirmar essa impressão. Assim, as mensagens, quando curtas, são rapidamente reconhecidas e lidas.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

6.3. Correção dos Erros

Definição: O critério *correção dos erros* diz respeito aos meios colocados à disposição do usuário com o objetivo de permitir a correção dos seus erros.

Justificativa: Em todo *software* educacional, no módulo prático e de resolução dos exercícios, o sistema deve informar adequadamente o usuário quando este erra ou tem uma dificuldade específica na sua resolução, orientando-o para a solução do problema. A ocorrência de erros pode ser positiva, por exemplo, nos *software* do tipo aprendizagem por descoberta. Neste contexto, os erros são bem vindos na medida em se pretende estimular a capacidade do aluno para a superação, por si só, do erro cometido. Mesmo assim, esta estratégia não exime a responsabilidade do *software* em orientar o aluno na solução das dificuldades que enfrenta.

Recomendações

1. A correção de erros durante a execução de exercícios deve ser otimizada, ou seja, deve permitir que o usuário faça a correção sem ter que refazer vários passos anteriores.

    - COMENTÁRIO 1: O sistema deve permitir ao usuário corrigir os erros sem a necessidade de interromper ou refazer todo o exercício. Deve ser possível a correção no momento da detecção do erro, permitindo a continuidade sem perder a lógica da resolução.

2. Na ocorrência de erros na resolução dos exercícios propostos, o software deve orientar e oferecer ao aluno a possibilidade de tentar refazer o exercício.

    - COMENTÁRIO 1: O sistema deve possibilitar ao usuário, na ocorrência de erros, retornar ao exercício não resolvido, permitindo novas tentativas.

    - COMENTÁRIO 2: O sistema deve informar ao usuário essa possibilidade.

3. Persistindo no erro durante a resolução dos exercícios, o software deve conduzir o usuário, fornecendo-lhe sequências explicativas para a correção das respostas inadequadas.

    - COMENTÁRIO 1: Nos casos de persistência de erro por parte do usuário na resolução de um exercício, o sistema deve disponibilizar, em forma de tutorial, explicando passo-a-passo a solução do mesmo.

    - COMENTÁRIO 2: O sistema deve conduzir o usuário para que ele compreenda, de forma clara, toda a sequência de resolução, permitindo a identificação e compreensão da incorreção.

4. O software deve fornecer a resolução dos exercícios após longa persistência no mesmo erro.

    - COMENTÁRIO 1: Caso ocorra uma longa persistência de um mesmo erro, o sistema deve apresentar a solução, para que o usuário não se sinta desestimulado e abandone a questão, concluindo que o exercício não tem resolução.

    - COMENTÁRIO 2: A resposta apresentada pode ajudar ao aluno a descobrir a solução da questão.

5. O software deve permitir a mudança automática de exercício, se o aluno persistir no erro, conduzindo-o a outro tipo de exercício, com um menor grau de dificuldade.

    - COMENTÁRIO 1: O sistema deve conduzir o usuário, em casos de persistência de erro, a exercícios que apresentem um grau de dificuldade menor em relação ao atual, estimulando seu raciocínio e sua capacidade de resolução.

    - COMENTÁRIO 2: O aluno pode não apresentar conhecimentos suficientes para a solução de determinados exercícios, necessitando ser conduzido a uma

categoria abaixo em relação ao grau de dificuldade.

- COMENTÁRIO 3: A resolução de exercícios com menor grau de dificuldade pode estimular a confiança no aluno e prepara-lo melhor para uma fase mais difícil.

6. O software deve possuir registros das dificuldades enfrentadas pelo aluno na resolução dos exercícios.

   - COMENTÁRIO 1: O sistema deve apresentar um recurso para registrar as dificuldades encontradas pelo usuário, permitindo uma análise posterior desses dados, buscando informações que possam contribuir na melhor forma de conduzir as deficiências dos alunos.

7. Caso o usuário tenha a necessidade de recorrer a teoria para a resolução dos exercícios, este acesso deve ser facilitado por meio de um atalho.

   - COMENTÁRIO 1: O sistema deve dispor de atalhos, em qualquer nível dos exercícios, que permitam ao usuário a consulta do conteúdo teórico, disponível no aplicativo, para a resolução de eventuais dúvidas ou em busca de embasamento para a solução dos exercícios.

8. Deve ser disponibilizada a opção de menu “gravar” para registrar a resolução dos exercícios.

   - COMENTÁRIO 1: O sistema deve disponibilizar ao usuário o recurso de gravação, para que ele possa, futuramente, rever seus métodos de condução e/ou resolução de exercícios.

   - COMENTÁRIO 2: Essa opção deve estar sempre disponível e em todos os níveis em forma de menu.

9. Qualquer ação deve poder ser revertida através da opção DESFAZER.

- COMENTÁRIO 1: A regressão do diálogo (opção "DESFAZER") deve ser prevista sempre e em local apropriado, devendo estar disponível ao usuário caso queira retroceder passos em uma tarefa ou etapas de um exercício (Figura 210).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

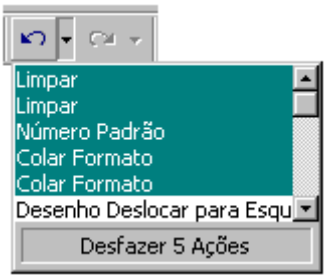


Fig.210 – Opção DESFAZER – várias ações.

10. A regressão do diálogo deve poder ser desfeita, através da opção REFAZER.

- COMENTÁRIO 1: A ação "DESFAZER" deve, também ela, ser reversível, de maneira que uma segunda ação restabeleça o estado anterior à regressão (trata-se da reprogressão do diálogo) (Figura 211).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

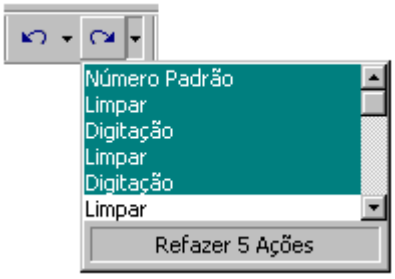


Fig.211 – Opção REFAZER – várias ações.

11. Os comandos para a opção DESFAZER e REFAZER o diálogo devem estar diferenciados.

- COMENTÁRIO 1: Se diferentes interrupções são possíveis em um diálogo, elas devem ser facilmente identificáveis e diferenciadas.

- COMENTÁRIO 2: Como a regressão do diálogo pode também ser anulada, pode-se facilmente voltar ao estado que havia sido anulado retomando a mesma ação. Todavia, essa combinação é confusa para o usuário. As ações de regressão ("desfazer", em Inglês "undo") e

de reprogressão ("refazer", em Inglês "redo") devem estar separadas (Figuras 212, 213 e 214).

REFERÊNCIA: Bodart & Vanderdonckt [1993]

Fig.212 – Menu Editar – Opções Desfazer e Refazer.

Fig.213 – Barra de Ferramenta – Opções Desfazer e Refazer.

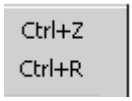


Fig.214 – Teclas de atalho – Opções Desfazer e Refazer.

7. Avaliação da aprendizagem

Descrição: Este critério refere-se aos meios disponíveis no sistema para verificar a aprendizagem dos conteúdos.

Justificativa: O *software* educacional geralmente apresenta uma componente prática, porém o tipo de exercícios propostos irá depender da modalidade de *softwar*e, seja ele um tutorial, um simulador, etc. Por exemplo:

– Os tutoriais apresentam geralmente diálogos que simulam uma relação professor-aluno e apresentam questões simultaneamente com a apresentação do conteúdo, verificando a aprendizagem na medida em que o aluno evolui na apresentação.

– Os simuladores geralmente apresentam um módulo teórico e um módulo prático, onde é requerido ao aluno que realize simulações de determinados sistemas.

Independentemente do tipo de *softwar*e, é importante que o sistema permita, de alguma maneira, verificar se os conceitos estão sendo aprendidos pelo aluno. Esta verificação dá-se, em geral, mediante perguntas ou situações-problema que o aluno deve resolver. O estilo de interação nos tutoriais apresenta um determinado número de questões que devem ser completadas com precisão, permitindo acompanhar o desenvolvimento do aluno periodicamente. No *software* do tipo aprendizagem por descoberta, estes objetivos tem um enfoque menor.

As respostas requeridas pelos usuários devem conter o mínimo possível de digitação dos dados. Isto pode ser conseguido pelo estilo de interação *"click* do *Mouse*".

Recomendações

1. No caso em que questões são apresentas para a verificação de um determinado conceito, estas devem ser formuladas de maneira clara e objetiva, evitando que o usuário faça uma interpretação errônea da questão.

   - COMENTÁRIO 1: As questões apresentadas pelo sistema devem apresentar clareza e objetividade de forma a não dificultar ou impedir sua compreensão.

   - COMENTÁRIO 2: É importante que as questões sejam corretamente interpretadas e que apresentem, apenas, as dificuldades inerentes aos objetivos pretendidos para a verificação.

2. O software deve dispor de algum recurso que permita avaliar o grau de compreensão dos alunos na resolução de problemas.

   - COMENTÁRIO 1: O sistema pode apresentar o recurso de digitação de relatório como forma de registro, permitindo que seu conteúdo seja gravado e posteriormente impresso.

3. Durante a sequência de apresentação, o software deve propor questões para verificar a compreensão dos conteúdos, simulando uma relação entre professor e aluno.

   - COMENTÁRIO 1: O sistema deve dispor, em cada sequência de conteúdo apresentado, de uma bateria de exercícios, cujo objetivo será avaliar o nível de compreensão do aluno.

   - COMENTÁRIO 2: A apresentação dos exercícios somente após a finalização de toda a apresentação dos conteúdos pode dificultar sua resolução se forem referentes a algum conteúdo inicial.

4. O software deve possuir bom grau de coerência no conteúdo das questões apresentadas em função dos objetivos a que se propôs.

   - COMENTÁRIO 1: As questões devem apresentar um grau de coerência em relação ao conteúdo apresentado, obedecendo a suas sequências e níveis de aprendizagem.

5. O software deve armazenar informações relativas à interação dos alunos tais como pontuações, tempo de resposta, nível atingido.

   - COMENTÁRIO 1: Devem ser armazenadas as informações que podem indicar o comportamento e o aproveitamento do aluno durante a interação com o sistema.

   - COMENTÁRIO 2: A analise dessas informações podem indicar o perfil de cada aluno, suas deficiências e seus pontos fracos em relação aos conteúdos e a aprendizagem.

6. O software deve permitir gravar automaticamente os registros do desempenho dos alunos mesmo que estes abandonem o programa.

- COMENTÁRIO 1: O software deve ser capaz de salvar automaticamente todos os dados referentes a atividade desenvolvida pelo aluno em qualquer momento, mesmo nos casos em que ocorra abandono e o aluno não solicite sua gravação.

- COMENTÁRIO 2: O registro dessas atividades possivelmente servirá para avaliar qual o motivo da desistência do aluno naquele momento.

7. O programa deve permitir o registro seguro dos resultados obtidos pelos alunos, sem que se corra o risco dos mesmos serem facilmente alterados por outrem.

   - COMENTÁRIO 1: O sistema deve possuir um sistema seguro de registro de informações para cada aluno, utilizando recursos de senha, para impedir que os dados sejam acessados e/ou alterados por qualquer pessoa

   - COMENTÁRIO 2: Em alguns casos as informações poderão ser acessadas, também por professores ou tutores.

8. Homogeneidade/Coerência

Definição: O critério *homogeneidade* refere-se ao modo como as escolhas na concepção da interface (códigos, denominações, formatos, procedimentos, etc.) são conservadas idênticas em contextos idênticos e diferentes em contextos diferentes, de uma tela para outra.

Justificativa(s): Os procedimentos, rótulos, comandos, etc., são mais bem reconhecidos, localizados e utilizados, quando o seu formato, localização, ou sintaxe são estáveis de uma tela para outra, de uma secção para outra. Nestas condições, o sistema é mais previsível, a aprendizagem mais generalizável e os erros são reduzidos. É necessário escolher opções similares de códigos, procedimentos, denominações para contextos idênticos e utilizar os mesmos meios para obter os mesmos resultados. É conveniente padronizar, tanto quanto possível, todos os objetos quanto ao seu formato e sua denominação e padronizar também a sintaxe dos procedimentos. A falta de homogeneidade nos menus, por exemplo, pode aumentar consideravelmente os tempos de procura. A falta de homogeneidade dificulta a intuitividade do *software* e a situação de ensino/aprendizagem.

Recomendações

- Os ícones devem ser distintos uns dos outros e devem possuir sempre o mesmo significado de uma tela para outra.

  - COMENTÁRIO 1: Os símbolos e outros códigos devem possuir significados consistentes de uma apresentação ou de uma tela para outra.

  - COMENTÁRIO 2: Essa prática ajuda o usuário na aprendizagem dos novos códigos, e na obtenção de familiaridade. Quando os códigos possuírem significados especiais, eles devem estar definidos na tela.

REFERÊNCIA: Smith & Mosier [1986]

1. Os formatos de apresentação dos dados devem ser mantidos homogêneos de uma tela para outra.

   - COMENTÁRIO 1: O sistema deve apresentar arranjos de telas consistentes para dados similares em diferentes telas.

   - COMENTÁRIO 2: Formas consistentes para apresentar dados auxiliam os usuários novatos a aprender a interagir eficientemente com o sistema. Devem ser criadas formas de apresentação com uma estrutura evidente e consistente para o usuário, de modo que qualquer tipo particular de dados seja sempre apresentado da mesma maneira (Figuras 215 e 216).

- COMENTÁRIO 3: Os formatos devem ser alterados apenas para diferenciar tarefas claramente.

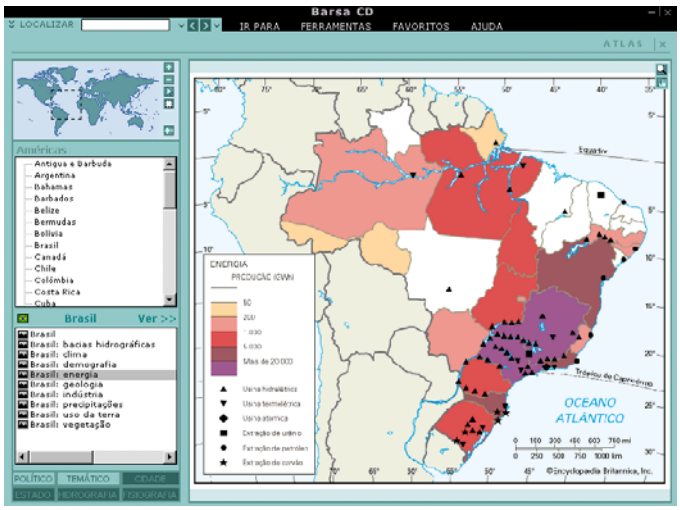


Fig. 215 – Geografia política – Brasil : dados sobre energia.

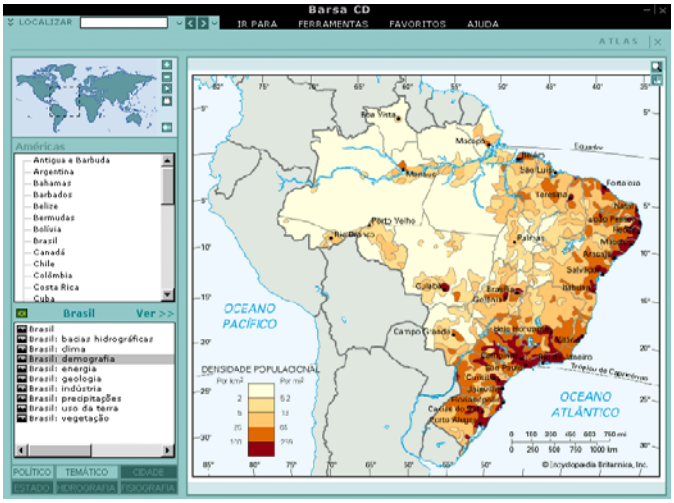


Fig. 216 – Geografia política – Brasil : dados demográficos.

REFERÊNCIA: Smith & Mosier [1986], Brown [1988]

2. A organização em termos da localização das várias características das janelas deve ser mantida homogênea de uma tela para outra.

   - COMENTÁRIO 1: Deve ser adotada uma organização consistente para a localização das várias características das janelas de uma tela para outra.

   - COMENTÁRIO 2: Áreas funcionais similares devem permanecer na mesma localização relativa em situações diversas.

   - COMENTÁRIO 3: A regularidade consiste em uniformizar os objetos de interação, segundo um planejamento definido (margens, dimensões, espaçamento,etc).

REFERÊNCIA: Smith & Mosier [1986], Bodart & Vanderdonckt [1993]

3. Os significados dos códigos de cores devem ser seguidos de maneira homogênea.

   - COMENTÁRIO 1: Ao utilizar código de cores, deve-se assegurar que cada cor represente uma única categoria de dados.

- COMENTÁRIO 2: A cor se constitui na dimensão de codificação dominante das telas. Se diferentes categorias de dados forem apresentadas, por exemplo em vermelho, ocorrerá um problema indesejado de incoerência visual que pode atrapalhar a correta assimilação da informação pelo usuário (Figuras 217, 218 e 219).

REFERÊNCIA: Smith & Mosier [1986]

4. A localização dos diferentes elementos funcionais deve ser mantida homogênea de uma tela para outra.

   - COMENTÁRIO 1: Deve-se apresentar um determinado tipo de informação na mesma localização de tela para tela. O usuário pode ler e interpretar dados de forma mais rápida, se eles forem apresentados de maneira previsível e familiar.

   - EXEMPLO 1: Se houver uma linha de status do sistema que apareça em várias telas ou apresentações, deve-se coloca-la na mesma localização em todas essas telas ou apresentações.

REFERÊNCIA: Brown [1988]

5. Os procedimentos de acesso às opções dos menus devem ser homogêneos.

   - COMENTÁRIO 1: A falta de homogeneidade nos menus, pode aumentar consideravelmente os tempos de procura de um conteúdo, tarefa ou comando.

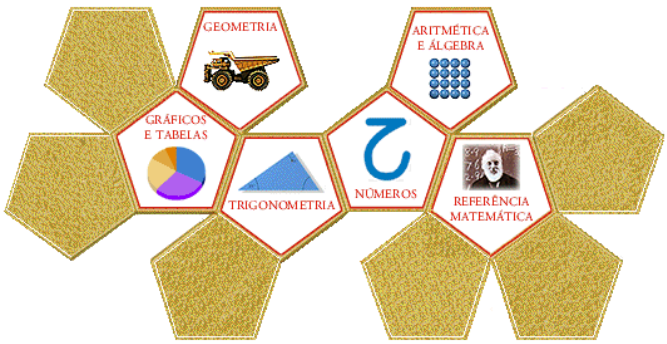


Fig. 217 – Módulo de matemática.

Fig. 218 – Módulo de física.

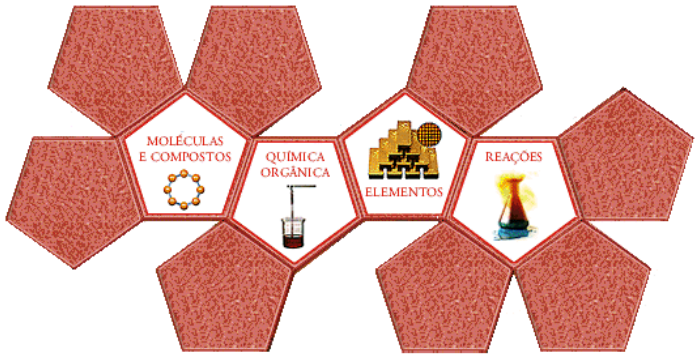


Fig. 219 – Módulo de química.

- COMENTÁRIO 2: Acesso a menus sem homogeneidade pode causar perda de concentração no usuário em função da procura de um determinado comando, desviando sua atenção da resolução de um exercício ou do aprendizado de conteúdos.

- EXEMPLO 1: Os vários aplicativos ou módulos que apresentam semelhanças entre si devem manter um padrão homogêneo de localização de menus de modo a facilitar seu acesso em diferentes ambientes, tais como: editores de texto, planilhas de cálculo, software de apresentação, etc (Figuras 220, 221 e 222).

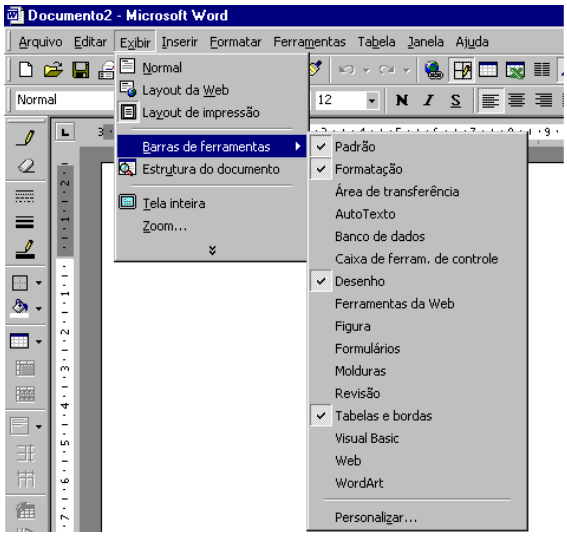


Fig. 220 - Editor de texto – Menu Exibir -> Barra de ferramentas.

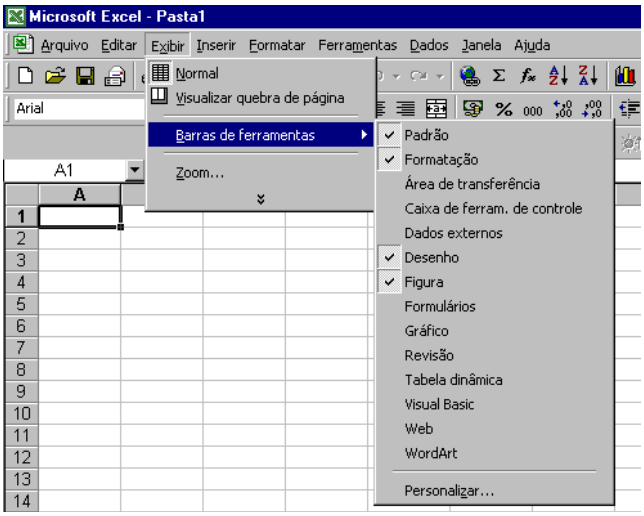


Fig. 221 – Planilha de cálculo – Menu Exibir -> Barra de ferramentas.

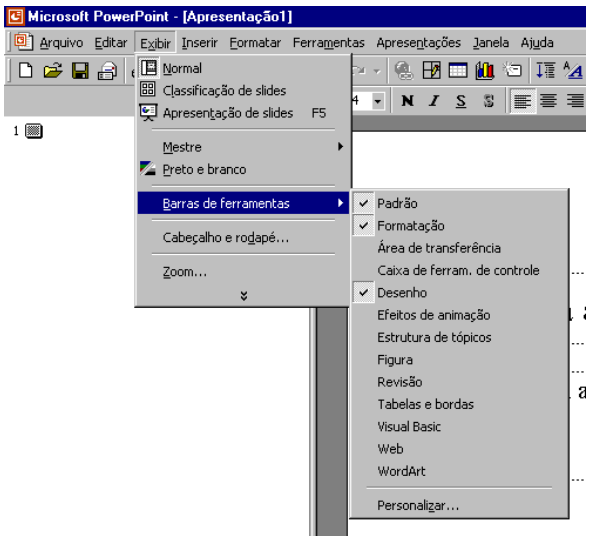


Fig. 222 – Software de apresentação – Menu Exibir -> Barra de ferramentas.

9. Significado dos Códigos e Denominações

Definição: O critério *significado dos códigos e denominações* diz respeito à adequação entre o objeto, informação apresentada ou pedida e sua referência. Códigos e denominações significativas possuem uma forte relação semântica com seu referente. Termos pouco expressivos para o usuário podem ocasionar problemas de condução que o podem levar à seleção de uma opção errada.

Justificativa: Quando a codificação é significativa, a recordação e o reconhecimento são melhores. Códigos e denominações não significativos podem sugerir operações inadequadas ao contexto, conduzindo os usuários a cometer erros. Deve ser evitado o uso de abreviações nos títulos das janelas, na barra e nas opções de menu. Situações em que seja pertinente o uso de abreviaturas e siglas como exemplo, durante a apresentação de um texto teórico, estas devem estar corretamente identificadas por meio de um glossário de siglas e abreviações.

Recomendações

1. As denominações dos títulos devem estar de acordo com o que eles representam.

   - COMENTÁRIO 1: O título deve ser único e pequeno, contudo deve ser suficientemente significativo a fim de garantir sua facilidade de recordação.

   - COMENTÁRIO 2: A identificação da tela deve servir como uma identificação telegráfica.

   - COMENTÁRIO 3: Por uma questão de flexibilidade, pode ser interessante deixar aos usuários a possibilidade de atribuir nomes para conjuntos particulares de dados que constituem apresentações frequentemente usadas. Os nomes podem ser nomes formais ou apelidos associados pelo computador aos nomes formais (Figuras 223 e 224).

REFERÊNCIA: Smith & Mosier [1986]

2. O vocabulário técnico utilizado deve ser familiar ao usuário.

   - COMENTÁRIO 1: Devem ser adotados vocabulários técnicos com os quais o usuário esteja familiarizado ou que lhe seja significativo.

   - COMENTÁRIO 2: Nos casos onde ocorrer a necessidade da utilização de terminologia técnica, deve-se disponibilizar um glossário para auxiliar o usuário.

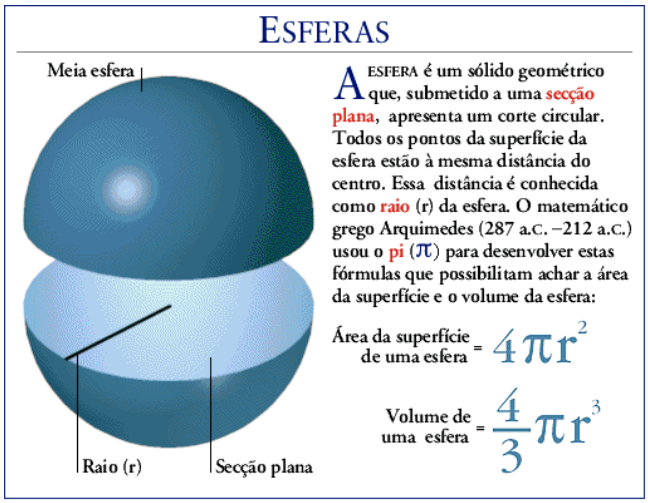


Fig. 223 – Título significativo.

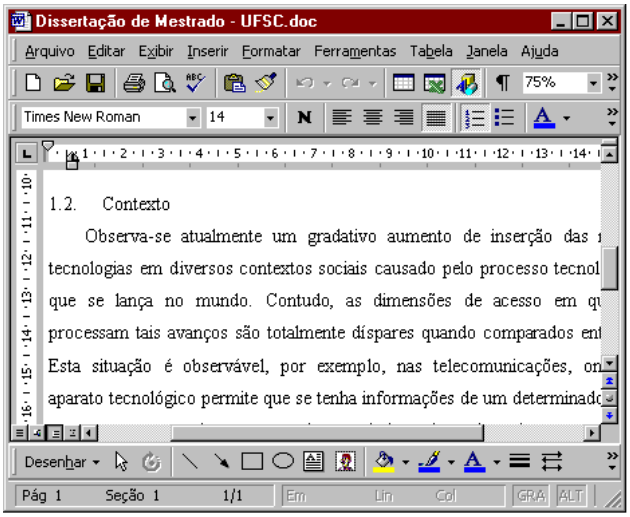


Fig. 224 – Título atribuído.

- COMENTÁRIO 3: Em aplicações voltadas para usuários especialistas o vocabulário técnico deve ser apropriado, utilizando termos especiais e jargão técnico (Figuras 225 e 226).

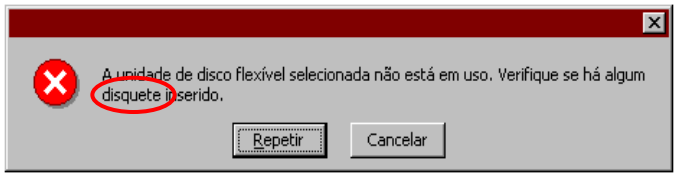


Fig. 225 – Exemplo positivo.

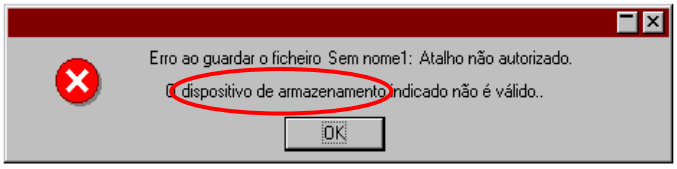


Fig. 226 – Exemplo negativo.

REFERËNCIA: Smith & Mosier [1986]

3. Deve existir um glossário para os termos técnicos.

- COMENTÁRIO 1: Códigos e denominações como pouco ou nenhum significado expressivo ou que tenham conotações técnicas devem estar contidos em um glossário.

- COMENTÁRIO 2: Ambientes desenvolvidos para usuários iniciantes ou sem nenhuma formação específica devem apresentar um glossário com informações detalhadas de seus procedimentos e funções.

4. O vocabulário utilizado nos títulos, rótulos, convites e mensagens de orientação devem ser familiares ao usuário e devem evitar palavras difíceis (Figuras 227, 228 e 229).

- COMENTÁRIO 1: Ao definir as palavras a serem utilizadas nos rótulos, prompts, mensagens para o usuário, deve-se adotar uma terminologia familiar aos usuários.

- COMENTÁRIO 2: Os testes com o usuário são freqüentemente necessários para eliminar palavras difíceis, abreviaturas e acrônimos que não são, geralmente, familiares para todos usuários.

REFERÊNCIA: Smith & Mosier [1986]

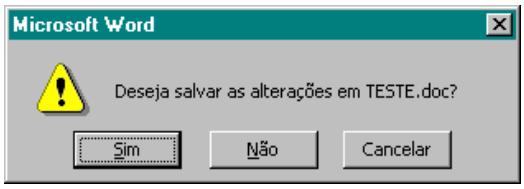


Fig. 227 – Exemplo positivo

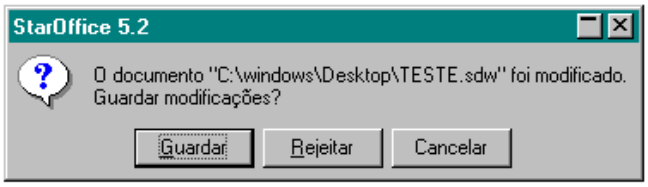


Fig. 228 – Exemplo negativo

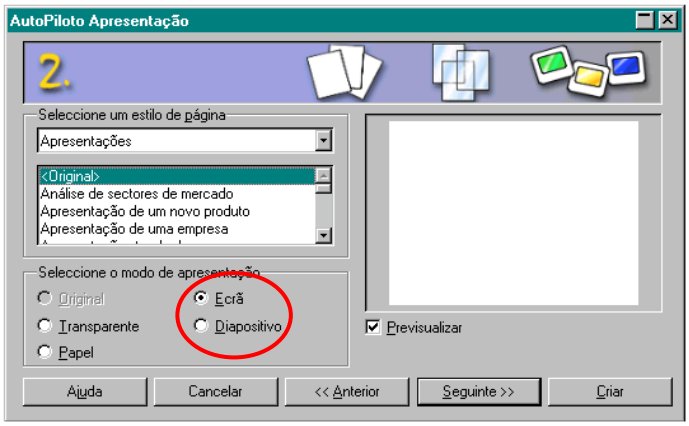


Fig. 229 – Rótulos com terminologia desconhecida

5. As denominações das opções de menu devem ser familiares ao usuário.

COMENTÁRIO 1: A terminologia utilizada nos nomes das opçõe dos menus deve ser familiar aos usuários (Figuras 230 e 231).

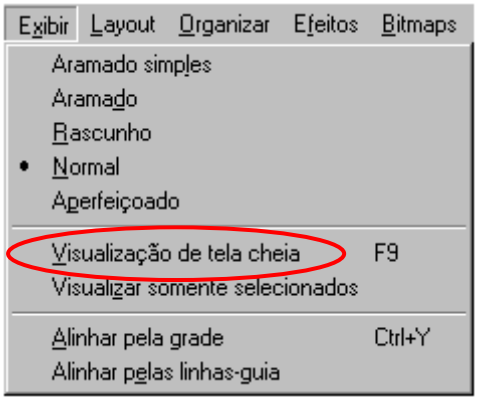


Fig. 230 – Terminologia Familiar.

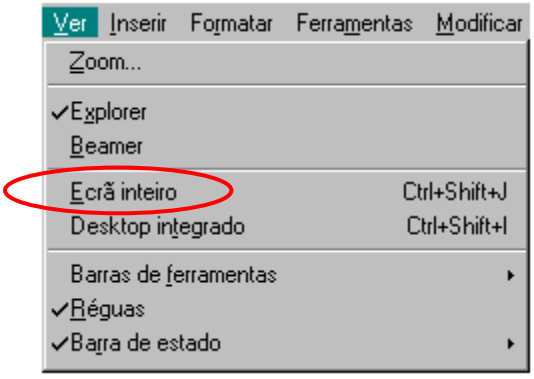


Fig. 231 – Terminologia desconhecida.

- COMENTÁRIO 2: Os títulos de menus devem combinar para que possam ser associados a outros termos, com o objetivo de formar títulos contendo diversas palavras (Animais/Mamíferos, Peixes, Répteis), a fim de representar a estrutura do menu (Figura 232).

Fig. 232 – Título representa a estrutura do menu.

- COMENTÁRIO 3: Os títulos de menus devem ser distintos e descritivos, curtos e representativos da opção e devem utilizar palavras-chave (Figuras 233 e 234).

Fig. 233 – Menus descritivos/explicativos.

- COMENTÁRIO 4: De uma maneira geral, é desejável adaptar a terminologia em função das tarefas dos usuários.

REFERÊNCIA: ISO 9241 parte 14 [1995]

- Os títulos das páginas de menus devem ser explicativos, e devem refletir a natureza da escolha a ser feita.

Fig. 234 – Menus com denominação arbitrária.

- COMENTÁRIO 1: Os títulos das páginas de menu devem permitir, por si só, um entendimento claro dos comandos e funções ali encontrados, dando ao usuário uma pré-indicação da escolha a ser feita (Figuras 235 e 236).

REFERËNCIA: Smith & Mosier [1986]

6. O sistema deve adotar códigos significativos ou familiares aos usuários em vez de códigos e denominações arbitrárias.

- COMENTÁRIO 2: Um código arbitrário, tal como um número de CGC ou CPF, pode, eventualmente, tornar-se familiar de acordo com sua frequência de uso (Figura 237).

REFERÊNCIA: Smith & Mosier [1986]

7. As abreviaturas devem ser significativas.

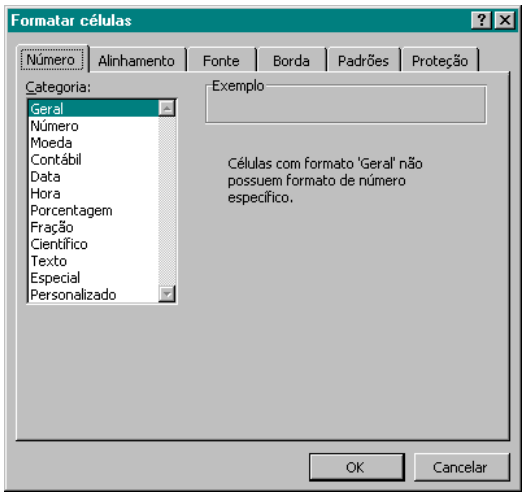


Fig. 235 – Formatar células – Número.

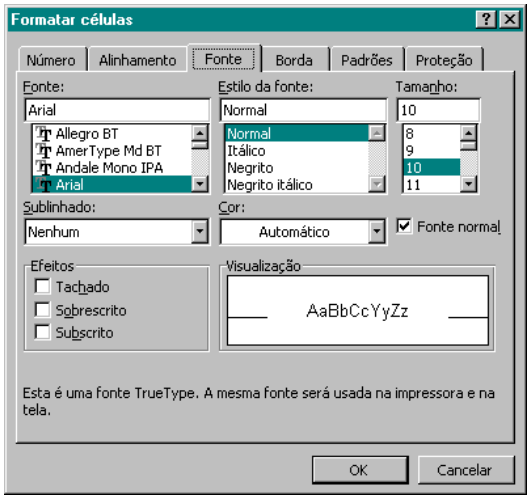


Fig. 236 – Formatar células – Fonte.

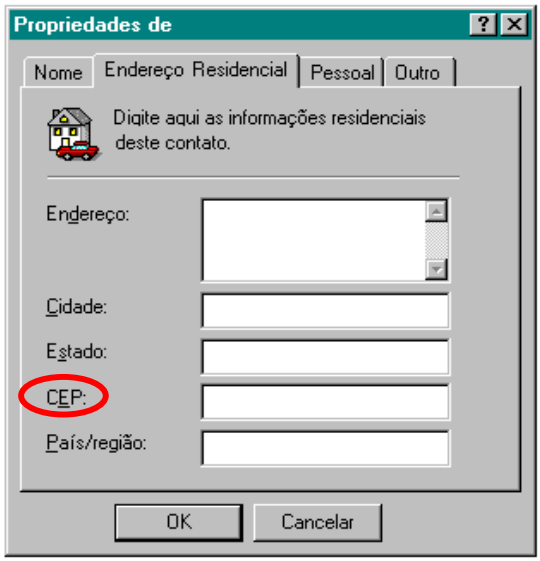


Fig. 237 – Código arbitrário, familiarizado com a frequência de uso.

- COMENTÁRIO 1: Quando houver restrições de espaço, deve-se utilizar abreviaturas conhecidas dos termos. Algumas abreviaturas que podem parecer naturais e óbvias para o projetista podem se constituir, na verdade, em abreviações obscuras para o usuário (Figura 238).

REFERÊNCIA: Brown [1988]

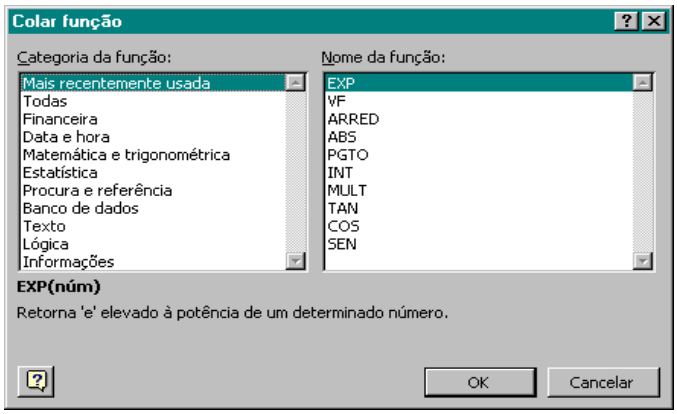


Fig. 238 – Abreviatura utilizada pelos usuários.

8. As abreviaturas devem ser facilmente distinguíveis umas das outras, evitando confusões geradas por similaridade (Figura 239).

- COMENTÁRIO 1: Na elaboração de abreviaturas ou outros códigos para a entrada de pequenos dados, deve-se observar a necessidade de manter uma distinção entre os mesmos buscando evitar confusões causadas por similaridade.

REFERÊNCIA: Smith & Mosier [1986]

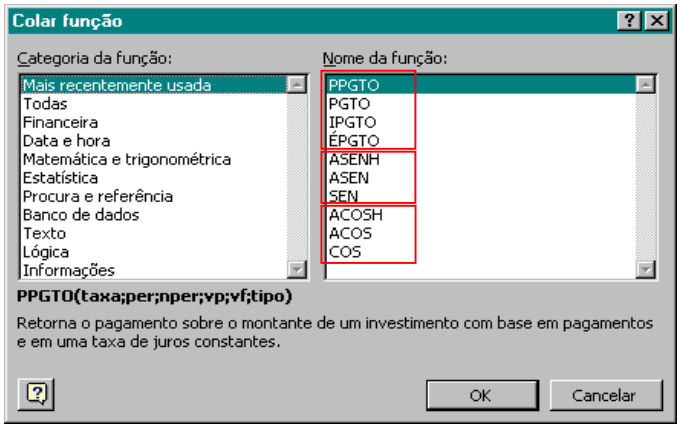


Fig. 239 – Abreviatura que apresentam similaridade.

9. No caso de gráficos, as denominações das linhas e colunas devem ser significativas e distintas (Figura 240).

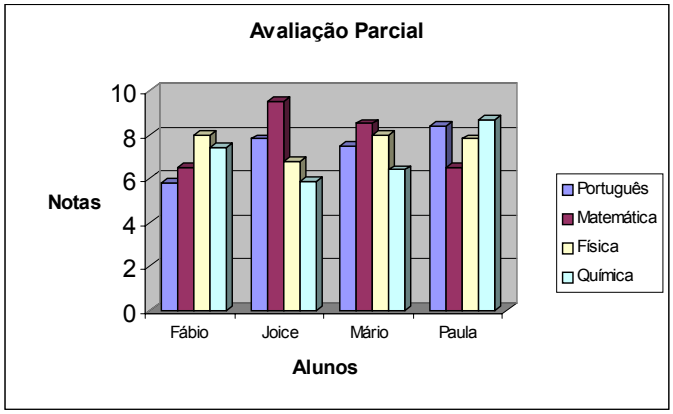


Fig. 240 – Linha e coluna – denominações significativas e distintas.

10. Os significados usuais das cores devem ser respeitados nos códigos de cores definidos.

- COMENTÁRIO 1: Deve-se definir a codificação com base nas associações convencionais de determinadas cores.

- COMENTÁRIO 2: Códigos arbitrários, expressos a partir de

cores, devem ser claramente indicados aos usuários através de uma legenda.

- COMENTÁRIO 3: Outras associações podem ser assimiladas pelo usuário, se a codificação de cores for empregada consistentemente.

- COMENTÁRIO 4: O código de cores deve prever a utilização de 10 ou 11 cores, no máximo. Deve-se utilizar um número reduzido, buscando mantendo uma consistência equilibrada de cores.

- COMENTÁRIO 5: As cores não devem estar associadas a mais do que um significado e deve respeitar os seguintes estereótipos naturais:

- Obs. 1: O vermelho deve ser utilizado para perigo, alarme, calor e comandos de interrupção (Figura 241);

- Obs. 2: O amarelo para advertências, teste e lentidão (Figura 242);

- Obs. 3: O verde para passagem livre, normalidade, vegetação e segurança (Figura 243);

- Obs. 4: O laranja para valor limite, radiação e substâncias corrosivas (Figura 244);

- Obs.5: O azul para frio, água, céu e calma (Figura 245);

- Obs. 6: O cinza para inatividade, neutralidade (Figura 246);

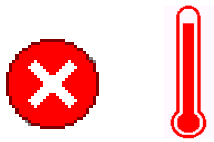


Fig. 241 – Perigo, alarme, calor ou atenção.

Fig. 242 – Advertência, atenção, alerta.

Fig. 243 – Passagem livre, confirmação.

Fig. 244 – Placa internacional de advertência contra substâncias químicas corrosivas.

Fig. 245 – Frio, água, neve.

Fig. 246 – Som inativo.

- Obs. 7: O branco é uma cor neutra (Figura 247).

REFERÊNCIAS: Smith & Mosier [1986]

Fig. 247 – Neutralidade.

10. Compatibilidade

Definição: O critério *compatibilidade* refere-se às relações que possam existir entre as características do usuário (sexo, idade, formação e competência) na realização de suas tarefas (conforme suas convenções, habilidades, preferências, expectativas e estratégias) e a organização das apresentações, das entradas e do diálogo de uma dada aplicação. Ela diz respeito também ao grau de similaridade entre diferentes ambientes e aplicações.

Justificativas: A transferência de informações entre um contexto e outro é tanto mais rápida e eficaz quanto menor for o volume de informação a ser recodificado.
A eficiência aumenta quando os procedimentos necessários ao cumprimento da tarefa são compatíveis com as características psicológicas do usuário; os procedimentos e as tarefas são organizados de maneira a respeitar as expectativas ou costumes do usuário; as traduções, as transposições, as interpretações ou referências à documentação são minimizadas.

Os desempenhos são melhores quando a informação é apresentada de uma forma diretamente utilizável (telas compatíveis com o suporte tipográfico, denominações de comandos compatíveis com o vocabulário do usuário, etc).

Recomendações

- Os procedimentos de diálogo devem ser compatíveis com os definidos pelos padrões do ambiente em que roda o software.

    - COMENTÁRIO 1: O sistema deve apresentar as caixas de diálogo para abertura, gravação e fechamento de arquivos, além das caixas para impressão com as formatações mais próximas possíveis das padronizadas por sistemas operacionais, tais como: CUA, MS Windows, Macintosh, etc.

REFERÊNCIA: Bodart & Vanderdonckt [1993]

1. O sistema deve seguir as convenções dos usuários para dados padronizados.

    - COMENTÁRIO 1: O formato dos dados deve respeitar o formato do país em que a aplicação será utilizada.

    - EXEMPLO 1: No Brasil o formato da data é dia/mês/ano, na Inglaterra e EUA é mês/dia/ano.

REFERÊNCIA: Smith & Mosier [1986]

2. O sistema deve utilizar unidades de medida familiares ao usuário.

- COMENTÁRIO 1: O sistema deve apresentar as unidades de medida de acordo com as que são normalmente utilizadas, evitando que o usuário tenha que se adaptar a um sistema métrico novo e fora do contexto (Figura.248)

- COMENTÁRIO 2: O sistema deve apresentar recursos que permitam a configuração do sistema métrico, caso seja do interesse do usuário.

- COMENTÁRIO 3: Quando os dados necessitarem ser convertidos para unidades familiares, o computador deverá fazê-lo automaticamente.

- COMENTÁRIO 4: Quando uma unidade de medida é consistentemente associada a um determinado campo de dados, deve-se incluir essa unidade já como parte do rótulo ou do campo, não solicitando que o usuário a digite (Figura 249).

- COMENTÁRIO 5: Quando unidades de medida alternativas são aceitáveis, deve ser fornecido espaço no campo de dados de tal forma que o usuário possa designar essas unidades.

REFERÊNCIA:Smith & Mosier [1986]

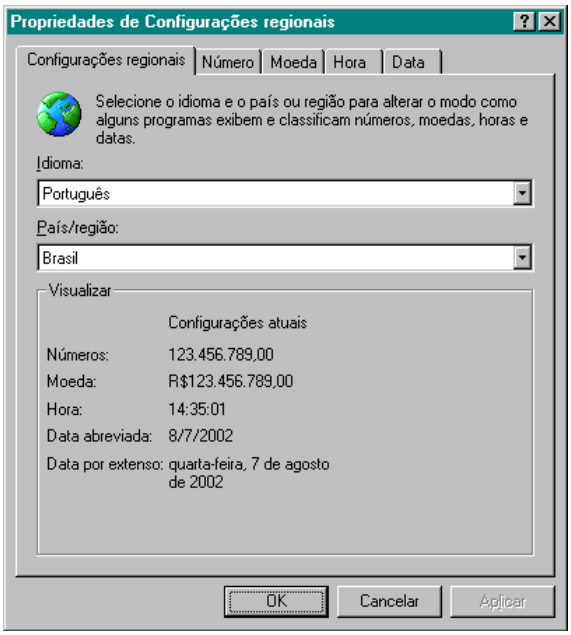


Fig. 248 – Formato dos dados de acordo com o país.

Fig. 249 – Unidade de velocidade - Kilometros por hora ao invés de milhas por hora.

REFERÊNCIA BIBLIOGRÁFICA

BARROS, Jorge Dalledone; D'AMBROSIO, Ubiratan. **Computadores, Escola e Sociedade**. Scipione. São Paulo. 1988.

BATISTA, João; FIGUEIREDO, A. Dias. **Desenvolvimento de Programas Educativos por Portotipagem Continuadamente Evolutiva**. 2º Simpósio Investigação e Desenvolvimento de Software Educativo. Coimbra. Portugal, 1997.

BERGAMO, Marília Lyra. **TUS – Tutorial de Usabilidade de Software**. RT- QSW – 2000/01. Universidade Católica de Brasília. Brasília 2000.

CYBIS, Walter de Abreu. **Ergonomia de Interface Homem-Computador**. Apostila de Curso, Laboratório de Utilizabilidade UFSC/Senai-SC/CTAI. Florianópolis. SC.1995 Disponível em: <www.labiutil.inf.ufsc.br/ergolist> . Acesso em 18 de outubro de 2001.

CYBIS, Walter de Abreu. **Qualidade do software na interação com o usuário: uma abordagem ergonômica**. Florianópolis: LabIUtil, 1997.

DIX, Alan J., et al. **Human-computer interaction**. 2 ed. [s.l]: Prentice Hall, 1997. 638 p.

GAINES, Brian R.; SHAW, Mildred L. D. **A interação computador-usuário: um novo meio de comunicação**. [s.l.]: LTC, 1987. 178 p.

GALVIS-PANQUEVA, Álvaro H. **Software Educativo Multimídia: Aspectos críticos no seu ciclo de vida**. Revista Brasileira de Informática na Educação. Florianópolis, nº 1, setembro, 1997.

GAMES, Luciano. **TICESE- Técnica de Inspeção de Conformidades Ergonômicas para Software Educacional.** Dissertação de Mestrado. Universidade do Minho. Guimarães. Portugal 1998.

HACK, Catapan Araci, et all. **ERGONOMIA EM SOFTWARE EDUCACIONAL: A possível integração entre usabilidade e aprendizagem**. IHC, Campinas – SP, 1999. Disponível em: <www.unicamp.br/~ihc99/ihc99/AtasIHC99/art24.pdf> . Acesso em 21 de outubro de 2001.

HELANDER, Martins G. (Editor). **Handebook of Human-computer interaction**. 2 ed. [s.l.]: Elsevier, 1997. 1582 p

ISO 9241. **Ergonomic requirements for Office work with visual display terminals**, Partes 10-14 Menu Dialogues. Draft International Standard ISO 9241, 1993.

MARTINS, Isa Haro; SOUZA, Clarisse Siecknius. **Uma Abordagem Semiótica na Utilização dos Recursos Visuais em Linguagens de Interface**. IHC, Rio de Janeiro, 1998. Disponível em: <www.inf.puc-rio.br/~ihc98/atas/E-Martins.zip> . Acesso em 14 de outubro de 2001.

MAYHEW, Deborah. J. **Principles and guidelines in software user interface design**. Prentice Hall. Englewood Cliffs, NJ 1992.

MAYHEW, Deborah. J. **The Usability Engineering Lifecycle: a practitioner's handbook for user interface design**. Morgan Kaufmann. 1999.

MEDEIROS FILHO, Dante Alves; CINTRA, Jorge Pimentel. **Desenvolvimento de Software para o Ensino e Aprendizagem**. 4º Fórum de Informática Educativa. Fortaleza. 1999.

NIELSEN, akob. **Usability Engineering**. San Francisco: Morgan Kaufman, 1993. 362 p.

PARIZOTTO, Rosamelia. **Elaboração de um Guia de Estilo para Serviços de Informação em Ciência e Tecnologia via Web**. Dissertação de Mestrado. Universidade Federal de Santa Catarina. Florianópolis 1997.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software**. LTC. Rio de Janeiro, 2001.

PRECE, Jenny. **Human-computer interaction**.[s.l]:Addison-Wisley, 1994. 775 p

PRESSMAN, Roger S. **Engenharia de Software**. Makron Books. São Paulo, 1995.

RAMOS, Edla Maria Faust. **Análise Ergonômica do Sistema HiperNet buscando o aprendizado da cooperação e autonomia**. Tese de doutorado apresentada ao programa de

pós-graduação em Engenharia de Produção da Universidade Federal de Santa Catarina. Florianópolis, 1996.

ROCHA, Ana Regina Cavalcanti (org). **Qualidade de Software: teoria e prática**. [s.l.] Prentice Hall, 2001. 303 p.

SCHUHMANHER, V. R. N. **Análise e Concepção de um Guia de Estilo para Seleção e Configuração de Objetos de Interação**. Dissertação de Mestrado. UFSC. Florianópolis. 1998.

SILVA, Cassandra R. **Bases Pedagógicas e Ergonômicas para Concepção e Avaliação de Produtos Educacionais Informatizados**. Dissertação de Mestrado. Universidade Federal de Santa Catarina. Florianópolis, 1998.

TARJA, Sanmya Feitosa. **Informática na Educação: novas ferramentas pedagógicas para o professor da atualidade**. 2ª ed. Érica. São Paulo. 2000.

VALIATI, Eliane R. de Almeida; Et all. Guia-GEPESE: **Um Guia de Recomendações específico para Software Educacional.** IHC, Gramado – RS 2000.

WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti; NASCIMENTO, Célia Joseli. **Qualidade e Produtividade de Software**. Makron Books, 2001. 188 p.