DESIGN E AVALIAÇÃO DE INTERFACES HUMANO-COMPUTADOR

HELOÍSA VIEIRA DA ROCHA

MARIA CECÍLIA CALANI BARANAUSKAS

INSTITUTO DE COMPUTAÇÃO

UNIVERSIDADE ESTADUAL DE CAMPINAS

*Para **Mariana e Daniel***
*Para **Vitinho (meu anjo), Cibele e Vitor***
nosso amor ...

PREFÁCIO

A comunidade profissional interessada na interação humano-computador data do início da década de 80, época em que os computadores pessoais começaram a ganhar mercado e escapar ao uso restrito de especialistas. O desenvolvimento da área nas últimas duas décadas gerou um corpo de conhecimentos que tem ganhado espaço dia a dia junto à indústria de software. Exemplo disso são os laboratórios de usabilidade de software e departamentos especiais em design criados nas grandes fabricantes de software como é o caso da Xerox, Apple, etc. .

Interação Humano-Computador (IHC) pode ser definida como *a disciplina relativa ao design, avaliação e implementação de sistemas computacionais interativos para uso humano e aos fenômenos que os cercam*. No Brasil é recente a preocupação com a área, refletida na inserção da disciplina nas diretrizes curriculares para os cursos de graduação em Ciência da Computação. Também a formação de uma comunidade de acadêmicos e profissionais da indústria tem se consolidado a partir de eventos científicos na área. Já foram realizados cinco Workshops sobre Fatores Humanos em Sistemas Computacionais. O primeiro (IHC98) aconteceu junto ao Simpósio Brasileiro de Engenharia de Software (SBES), em Maringá, PR. e o segundo (IHC99) ocorreu em paralelo ao Brasiliam Symposium on Computer Graphics and Image Processing (SIBGRAPI), sob os auspícios do Instituto de Computação e da Faculdade de Engenharia Elétrica da Unicamp. No ano 2000 aconteceu o terceiro encontro (IHC2000), em Gramado, RS, seguido do quarto (IHC2001) em Florianópolis, SC e do quinto (IHC2002) em Fortaleza. Este livro, escrito originalmente para a Escola de Computação 2000, aconteceu em um momento conjuntural importante onde havia demanda tanto institucional quanto na área da indústria de software para um texto no assunto, o que se refletiu em sua edição original esgotada em um mês após seu lançamento. Buscando responder à procura que tem acontecido desde então, lançamos esta re-impressão do livro.

Este livro é uma contribuição ao processo de ensino na área de Interfaces e Interação Humano-Computador, a partir de referências clássicas que fundamentam a prática de IHC, bem como através de literatura recente que aponta para novas teorias, métodos e processos de design e avaliação de software. A proposta do livro é resultado da prática das autoras em pesquisa relacionada ao design e avaliação de ambientes computacionais para o usuário final e construída nos últimos dez anos no ensino de disciplinas sobre design, implementação e avaliação de interfaces humano-computador em cursos de graduação e pós-graduação no Instituto de Computação da Unicamp. O livro procura abordar os fundamentos da área, bem como apontar para suas novas fronteiras, em um espaço adequado ao ensino introdutório da disciplina. Não pretendemos, portanto, esgotar o assunto, embora procuremos apresentar o design e avaliação de interfaces, entendendo-os como parte de um mesmo processo de criação de software para o usuário final.

O livro está estruturado em cinco capítulos onde: no Capítulo 1 fazemos uma apresentação geral da área e definimos a terminologia e conceitos básicos utilizados nos demais capítulos; no Capítulo 2, são aprofundados aspectos relativos aos fatores humanos envolvidos no processo de interação humano-computador, tanto capacidades físicas como capacidades cognitivas são tratadas nesse capítulo; no Capítulo 3 é discutido o processo de design de interfaces sob diferentes perspectivas explorando também os aspectos organizacionais e sociais do contexto do usuário e de suas atividades; no Capítulo 4 são discutidas técnicas de avaliação de interfaces nas diferentes fases de sua implementação, do design à implementação completa ou de um protótipo; finalmente no Capítulo 5, fazemos uma discussão geral dos resultados dessa área de estudo e aplicação, analisando os impactos individuais e sociais das atuais interfaces de usuário e apontando para o provável advento de uma nova computação.

Estaremos dando ao tema um tratamento tanto teórico quanto prático, pois todos os aspectos teóricos serão apresentados em paralelo a exemplos reais que os ilustram. Os conceitos apresentados, dependendo do interesse do leitor, podem ser imediatamente utilizados tanto em situações reais de desenvolvimento de software, quanto no ensino de IHC. Pelo menos é isso que temos observado junto aos nossos alunos, muitos deles profissionais vinculados a empresas e universidades.

O livro, portanto, é um esforço no sentido de se ter uma publicação nacional sobre o assunto e é voltado para um amplo público que compreende pesquisadores, profissionais da indústria de software, educadores e estudantes que estejam interessados em explorar e contribuir no desenvolvimento de sistemas computacionais usáveis.

Um *website* acompanha o livro (http://www.ic.unicamp.br/proj-ihc/DAIHC.html), incluindo apontadores para material adicional ao conteúdo de cada capítulo e para outros temas correlatos a IHC e não diretamente tratados no livro. Adicionalmente pretendemos incluir nesse *website* informação e material de apoio para professores e estudantes tornando disponível material didático, projetos e exercícios práticos no sentido de apoiar e documentar o desenvolvimento de cursos na área.

Heloísa Vieira da Rocha

Maria Cecília Calani Baranauskas

AGRADECIMENTOS

Como o final de milênio inspira novos caminhos a explorar, a Escola de Computação do ano 2000 não poderia ser diferente. Agradecemos à Comissão Organizadora pela oportunidade que gerou este livro.

Este livro tem implícita a contribuição de pessoas que têm compartilhado conosco, ao longo dos anos, a necessidade do rompimento de barreiras entre pessoas e computadores. Queremos agradecer a nossos alunos e ex-alunos da Ciência da Computação que nos motivam a discutir e trabalhar para a construção de conhecimento na área de Interfaces e Interação Humano-Computador. Muitos certamente identificarão suas contribuições no livro.

É desnecessário lembrar o papel que teve o Instituto de Computação da Unicamp em nos dar espaço para o projeto inicial do livro e ao Nied-Unicamp pela sua concretização nesta edição. Por isso também agradecemos. Da mesma forma, agradecemos aos nossos colegas, por nos fazerem lembrar desta tarefa a todo o momento, no corredor, e por estarem sempre solícitos a nos ajudar quando preciso.

Agradecemos à Luciana A. S. Romani, que se responsabilizou pela construção do *site* que acompanha este livro e ao Juliano Schimiguel pelo trabalho de revisão desta edição. Também agradecemos ao Daniel R. C. Silva pela ajuda na editoração final e à Letícia Lampert, que generosamente nos autorizou a utilizar o logotipo de sua criação para o IHC99.

Quem já escreveu um livro sabe do tempo e da atenção que roubamos de nossas famílias. Para elas nossos agradecemos especiais.

Caminhante, não há caminho.
Faz-se caminho ao caminhar.
Leonardo Boff

SUMÁRIO

Importante

Sequência de impressão final:

1. capa.doc
2. dedicatoria.doc
3. prefacio.doc
4. Capitulo1.doc
5. Capitulo2.doc
6. Capitulo3.doc
7. Capitulo4.doc
8. Capitulo5.doc

Obs.: os arquivos já possuem páginas em branco, indicando quando não deve haver texto no verso (essas páginas não estão numerdas, mas estão contabilizadas no total)

única exceção para o arquivo capa.doc que é página única , pois textos subsequentes são de responsabilidade da Escola (assumimos isso a partir das publicações de Escolas anteriores)

Qualquer problema é só entrar em contato:

Heloisa (heloisa@dcc.unicamp.br)
F: 19.7885866 (Unicamp)
F: 19.289.2169 (Resid.)

Cecilia (cecilia@dcc.unicamp.br)
F:19.7885870 (Unicamp)
F:19.2895106 (Resid.)

CAPÍTULO 1

O QUE É INTERAÇÃO/INTERFACE HUMANO-COMPUTADOR

The ideal system so buries the technology that the user is not even aware os its presence. The goal is to let people get on with their activities, with the technology enhancing their productivity, their power, and their enjoyment, ever the more so because it is invisible, out of sight, out of mind. People should learn the task, not the technology. They shoul be able to take the tool to the task, not as today, where we must take the task to the tool. And these tools should folow three axioms of design: simplicity, versatility, and pleasurability

Norman, 1998, pg xii

INTRODUÇÃO

Novas tecnologias provêem poder às pessoas que as dominam. Sistemas computacionais e interfaces acessíveis são novas tecnologias em rápida disseminação. Explorar o poder do computador é tarefa para designers que entendem da tecnologia e são sensíveis às capacidades e necessidades humanas.

A performance humana no uso de computadores e de sistemas de informação tem sido uma área de pesquisa e desenvolvimento que muito se expandiu nas últimas décadas. Isso tem sido feito usando-se poderosas ferramentas computacionais na análise de dados coletados de acordo com métodos da Psicologia Experimental. Outras contribuições também advém da Psicologia Educacional, do Design Instrucional e Gráfico, dos Fatores Humanos ou Ergonomia, e bem mais recentemente, da Antropologia e da Sociologia.

Interfaces de usuário têm produzido importantes estórias de sucesso tais como a da Netscape, America Online, Universal Online, ou Yahoo. Elas também tem produzido intensa competição, disputas por direitos autorais (por exemplo, Apple e Microsoft com relação à interface Windows), mega fusões (como a recente entre a America Online e TimeWarners), etc.

Individualmente, interfaces de usuário têm mudado a vida de muitas pessoas: médicos estão podendo fazer diagnósticos mais precisos; crianças estão expandindo os horizontes em ambientes de aprendizagem; artistas gráficos podem explorar mais possibilidades criativas; e pilotos têm mais segurança em seus vôos. Entretanto, algumas mudanças são perturbadoras e até desastrosas; freqüentemente usuários têm que lidar com frustração, medo e falha quando encontram design excessivamente complexos, com terminologia incompreensível e caóticos.

O crescente interesse no projeto de interfaces do usuário é bastante claro nos mais variados tipos de sistemas (Figuras 1.1, 1.2). Processadores de texto, ferramentas de edição, e softwares de manipulação de imagens são amplamente utilizados

FIGURA 1.1 – TELAS DO SIG ARCVIEW COM MAPA MOSTRANDO A MALHA MUNICIPAL DO ESTADO DE SÃO PAULO

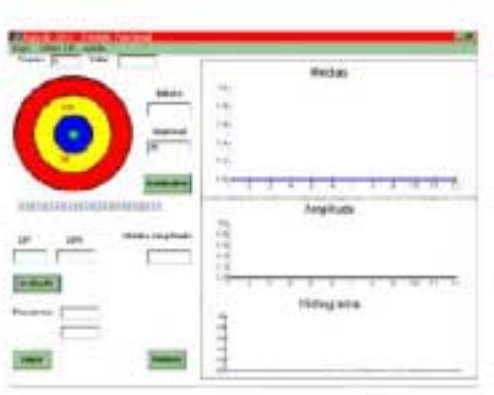


FIGURA 1.2 – TELA DO *JOGO DO ALVO* DESENOLVIDO PELO NIED - UNICAMP NO PROJETO DE FORMAÇÃO DE RECURSOS HUMANOS PARA A EMPRESA ENXUTA

Correio eletrônico, vídeo conferência e a WWW têm oferecido novas mídias para comunicação. Bibliotecas digitais de imagens têm se expandido em aplicações que vão da medicina (Figura 1.3) até a exploração do espaço.

Figura 1.3 – IMAGENS OBTIDAS DO *SITE* VISIBLE MAN QUE POSSUI UMA VASTA BIBLIOTECA DE REFERÊNCIAS MÉDICAS

Visualização científica e simuladores remotos permitem experimentos seguros e treinamento a baixo custo. Acesso público e educacional a *site*s de museus (Figura 1.4), bibliotecas ou fontes de informação governamentais estão se ampliando (Figuras 1.5,1.6).

FIGURA 1.4 – *SITE* DO MUSEU DE ARTE DE SÃO PAULO – PÁGINA DE ENTRADA E PÁGINA PRINCIPAL DA EXPOSIÇÃO VIRTUAL DO MICHELANGELO

FIGURA 1.5 – *SITE*S DA BIBLIOTECA NACIONAL E BIBLIOTECA DA ESCOLA DO FUTURO DA USP

FIGURA 1.6 – FONTES DE INFORMAÇÃO DO GOVERNO BRASILEIRO SOBRE DADOS ESTATÍSTICOS (IBGE) E AIDS

Ferramentas computacionais específicas e ambientes de programação permitem construir protótipos rápidos como as ferramentas de auxilio ao design de produtos industriais. Muitos de nós usam vários produtos eletrônicos, como os gravadores de vídeo cassete, fornos de microondas, telefones, etc. Arte, música, esportes e entretenimento são atualmente auxiliadas e suportadas por sistemas computacionais.

Profissionais das mais diferentes áreas têm contribuido significativamente para todo esse desenvolvimento, e dentre estes podemos destacar:

- **Designers de software** têm explorado maneiras melhores de organizar informação graficamente. Eles têm desenvolvido linguagens de consulta e facilidades visuais para entrada, busca e saída de informação. Têm usado sons (música e voz), representações tridimensionais, animação e vídeo para melhorar o conteúdo e a expressão das interfaces. Técnicas como manipulação direta, tele-presença, e realidade virtual mudam a maneira de interagir e de pensar sobre computadores.

- **Desenvolvedores de hardware** têm oferecido novos design de teclados e dispositivos de apontamento, além dos *displays* de alta resolução. Eles têm projetado sistemas com resposta rápida para complexas manipulações tridimensionais. Tecnologias que permitem entrada e saída por voz, entrada por gestos, telas de toque, em muito têm aumentado a facilidade de uso dos computadores.

- **Desenvolvedores na área de tecnologia educacional** estão criando tutorais *online*, e materiais de treinamento e explorando novas abordagens de discussões em grupo, ensino a distância, apresentações de vídeo etc. **Designers gráficos** estão fortemente envolvidos com o *layout* visual, seleção de cores e animação. **Sociólogos, antropólogos, filósofos, administradores** estão tratando do

impacto organizacional, ansiedade computacional, treinamento, grupos de trabalho distribuídos, suporte computacional ao trabalho cooperativo, e mudanças sociais em geral.

Portanto, estamos vivendo um momento vital e estratégico para os desenvolvedores de interfaces. Pode-se dizer que a tecnologia está pronta. Temos portanto as pontes e túneis construídos e agora as estradas precisam ser pavimentadas e as sinalizações pintadas para tornar possível o pesado tráfico da grande leva de usuários (Schneiderman, 1998).

Neste capítulo estaremos apresentando a área de Interfaces Humano-Computador de modo a poder situar e apresentar ao leitor os problemas, a terminologia e conceitos envolvidos nessa área de estudo e aplicação. Vale lembrar que muitos dos aspectos aqui apresentados serão aprofundados em capítulos subsequentes.

INTERFACE HUMANO-COMPUTADOR

Quando o conceito de interface surgiu, ela era geralmente entendida como o hardware e o software com o qual homem e computador podiam se comunicar. A evolução do conceito levou à inclusão dos aspectos cognitivos e emocionais do usuário durante a comunicação.

Muito embora algumas pessoas ainda possam se lembrar dos antigos teletipos, é comum hoje em dia pensarem na interface como a tela e o que nela é mostrado. O nome interface é tomado como algo discreto e tangível, uma coisa que se pode desenhar, mapear, projetar e implementar, "encaixando-a" posteriormente a um conjunto já definido de funcionalidades. Um dos objetivos deste livro é acabar com essa idéia substituindo-a por outra que ajude os construtores de interfaces a irem na "direção correta".

De acordo com Brenda Laurel (1990) a "direção correta" é aquela que leva o usuário a ter mais poder. Por exemplo, uma nova versão de um editor de textos comumente oferece o dobro de opções que a versão anterior. E com isso se espera que o usuário possa customizar melhor seu uso e conseguir atingir objetivos mais complexos. Este objetivo nem sempre é conseguido, pois o enorme conjunto de funções a as convenções de interface que deverão ser aprendidas de modo a se poder usufruir as pretensas novas qualidades, na maioria dos casos, deixam o usuário atônito e cansado. Certamente as melhoras acrescentadas ao produto oferecem ao usuário mais poder e qualidade ao produto final, oferecendo mais graus de liberdade na sua concepção. Mas tudo isso se perde quando o custo para o usuário é muito alto. O que acontece é que a nova versão é adotada, muitas vezes por problemas de compatibilidade entre diferentes versões de um produto, mas toda melhoria é deixada de lado e o usuário continua usando o mesmo domínio de funções que ele já

conhecia. Concluindo, para que o usuário tenha mais poder, é preciso sim, que mais funcionalidade seja oferecida mas é fundamental a facilidade de uso.

UMA PRIMEIRA DEFINIÇÃO DE INTERFACES

Primariamente, como já dissemos, se visualiza uma interface como um lugar onde o contato entre duas entidades ocorre (por exemplo, a tela de um computador). O mundo está repleto de exemplos de interfaces: a maçaneta de uma porta, uma torneira, a direção de um carro, etc.

A forma das interfaces reflete as qualidades físicas das partes na interação. A maçaneta de uma porta é projetada para se adequar à natureza da mão que irá usá-la, o mesmo acontece com o câmbio de um carro (observe que a localização do câmbio dentro do carro sugere o uso por uma pessoa destra). Existem tesouras de dois tipos uma para pessoas destras e outra para pessoas canhotas.

O que muitas vezes é esquecido é que a forma da interface também reflete o que pode ser feito com ela. Tomando o exemplo da maçaneta, podemos ver que no mundo existem diversos formatos de maçaneta e de acordo com o formato sabemos como deve ser aberta uma porta: girando a maçaneta no sentido anti-horário, empurrando a porta, puxando a porta, etc. (Norman, 1988). O mesmo acontece com a forma das torneiras onde se deve girar ou empurrar ou levantar uma alavanca, etc.

Nos exemplos anteriores da porta e da torneira que foram feitas para serem abertas por um humano podemos dizer que o humano é o agente e a porta (ou torneira) são os pacientes dessa ação. Mas, temos também as portas, ou torneiras, que abrem automaticamente quando identificam através de um sensor ou uma câmera a presença de alguém (mesmo que esse alguém não queira abrir a porta). Nesse caso o sentimento que temos de quem está controlando a interação é bastante diferente. Em muitos banheiros públicos existem instalados aqueles secadores automáticos de ar quente para mãos e muitas vezes, mesmo não querendo usá-los eles se ligam porque nos encostamos próximos a eles ou sem querer passamos a mão perto do sensor. E as torneiras que sempre se fecham antes de acabarmos de lavar as mãos? Nesses casos, não é mais o humano que está no controle da interação.

Portanto, podemos ter como uma definição de base, que uma interface é uma superfície de contato que reflete as propriedades físicas das partes que interagem, as funções a serem executadas e o balanço entre poder e controle (Laurel, 1993).

EVOLUÇÃO DE INTERFACES E SUA CONCEITUAÇÃO

Interface tornou-se uma tendência (ou moda, como nomeiam os mais incrédulos) como um importante conceito a ser explorado nos últimos anos, e isso é largamente atribuído a introdução dos computadores Macintosh da Apple. Certamente, quando se pensa hoje em dia em Interfaces Humano-Computador (IHC) imediatamente se visualiza ícones, menus, barras de rolagem ou talvez, linhas de comando e cursores piscando. Mas certamente interface não é só isso.

Podemos fazer um histórico analisando a geração de interfaces, da mesma forma com que analisamos gerações de computadores, ou seja, fazendo um forte paralelo com os componentes de hardware que as suportam (Tesler, 1991). Nielsen(1993) apresenta uma
tabela onde ele faz esse relacionamento e também qualifica a categoria de usuários de computadores em cada geração, o que é de absoluta relevância para o desenvolvimento de interfaces (Tabela1.1).

GERAÇÃO	TECNOLOGIA DE HARDWARE	MODO DE OPERAÇÃO	LINGUAGENS DE PROGRAMAÇÃO	TECNOLOGIA TERMINAL	TIPO DE USUÁRIOS	IMAGEM COMERCIAL	PARADIGMA DE INTERFACE DE USUÁRIO
-1945 pré-histórica	Mecânia e eletromecânica	Usado somente para cálculoes	Movimento de cabos e chaves	Leitura de luzes que piscam e cartões perfurados	Os próprios inventores	Nenhuma (computadores não sairam dos laboratórios)	Nenhum
1945-1955 pioneira	Válvulas, máquinas enormes e com alta ocorrência de falha	Um usuário a cada tempo usa a máquina (por um tempo bastante limitado)	Linguagem de máquina 001100111101	TTY. Usados apenas nos centros de computação	Especialistas e pioneiros	Computador como uma máquina para cálculos	Programação, batch
1955-1965 histórica	Transistores, mais confiáveis. Computadores começam a ser usados fora de laboratórios	Batch (computador central não acessado diretamente)	Assembler ADD A,B	Terminais de linha glass TTY	Tecnocratas, profissionais de computação	Computador como um processador de Informação	Linguagens de Comando
1965-1980 tradicional	Circuito integrado. relação custo-benefício justifica a compra de computadores para muitas necessidades	Time-sharing	Linguagens de alto nível (Fortran, Pascal, C)	Terminais full screen, caracteres alfa-numéricos. Acesso remoto bastante comum	Grupos especializados sem conhecimento computacional (caixas automáticos, p.ex.)	Mecanização das atividades repetitivas e não criativas	Menus hierárquicos e preenchimento de formulários
1980-1995 moderna	VLSI. Pessoas podem comprar seu computador.	Computador pessoal para um único usuário	Linguagens orientadas a problemas/objetos (planilhas de cálculo)	Displays gráficos. estações de trabalho, portáveis	Profissionais de todo tipo e curiosos	Computador como uma ferramenta	WIMP (Window,Icons,Menus, e Point devices)
1995- futura	Integração de alta-escala. Pessoas podem comprar diversos computadores	Usuários conectados em rede e sistemas embutidos	Não imperativas, provavelmente gráficas	Dynabook, E/S multimídia, portabilidade simples, modem celular	Todas as pessoas	Computador como um aparelho eletrônico	Interfaces não baseadas em comando.

TABELA 1.1 - GERAÇÃO DE COMPUTADORES E DE INTERFACES DE USUÁRIOS (ADAPTADO DE NIELSEN,1993, P.50

De modo semelhante Walkers (1990) faz uma análise histórica da evolução de interfaces sob o aspecto do tipo de interação entre o usuário e o computador. No início havia um relacionamento um a um entre uma pessoa e o computador através de chaves e mostradores das primeiras máquinas como o ENIAC ou UNIVAC, EDVAC (foto ao lado)[1].

O advento dos cartões perfurados e do processamento em *batch* substituiu essa interação direta entre o homem e o computador por uma transação mediada pelo operador do computador. *Time sharing* e o uso dos teletipos trouxeram novamente o contato direto e conduziram o desenvolvimento das interfaces de linhas de comando e orientadas por menu (Figura1.7). O estilo de diálogo é bastante simples, onde uma pessoa faz alguma coisa e o computador responde

FIGURA 1.7 – TELAS DE UM PROGRAMA COM ESTRUTURA DE DIÁLOGO POR MENU E LINEAR

Essa noção simplista de uma conversação levou ao desenvolvimento de um modelo de interação que trata o humano e o computador como duas entidades diferentes que conversam intermediadas por uma tela.

[1] Extraída de http://ei.cs.vt.edu/~history/index.html em fev./2000

Avanços da Lingüística têm demonstrado que diálogo não é linear, ou seja, quando dizemos alguma coisa, você pensa sobre o que dissemos e aí dá uma resposta, nós vamos pensar sobre a resposta e aí …., e assim por diante. Portanto, para que o diálogo efetivamente ocorra é necessária a existência, ou a construção, de um meio comum de significados.

As atuais interfaces gráficas explicitamente representam o que vem a ser esse meio de significados comum, pela aparência e comportamento dos objetos na tela. Este conceito dá suporte a idéia de que uma interface é um contexto compartilhado de ação no qual tanto o computador como o humano são agentes (Laurel, 1993). Enganos, resultados inesperados e mensagens de erro são evidência típica de uma quebra na conversação, onde o pretenso meio de significados comum torna-se uma seara de desentendimentos.

A noção de **metáforas de interfaces** (Carroll *et al*., 1988; Wozny, 1989) foi introduzida para prover às pessoas um esquema do funcionamento da interface que prevenisse tais desentendimentos, ou seja, facilitassem a criação desse contexto compartilhado. Por que metáforas?

METÁFORAS DE INTERFACE

Metáforas são parte integrante de nosso pensamento e linguagem. Elas aparecem não somente na poesia ou literatura, mas em nossa linguagem cotidiana. E geralmente as pessoas não se dão conta de que estão usando metáforas, elas são invisíveis. Alguns exemplos bastante comuns: *gastar* dinheiro, *atacar, defender e destruir* um argumento; *tratar* superficialmente um assunto; trânsito *engarrafado*; etc. As metáforas funcionam como modelos naturais, nos permitindo usar conhecimento familiar de objetos concretos e experiências para dar estrutura a conceitos mais abstratos. As características de metáforas em nossa linguagem são as mesmas que governam o funcionamento de metáforas de interfaces. Da mesma forma que metáforas invisíveis permeiam nossa linguagem cotidiana elas o fazem nas interfaces que usamos e projetamos (Erickson, 1990). Por exemplo, um usuário quando arrasta um documento de um diretório (ou *pasta*) para outro nos sistemas gerenciadores de arquivos de ambientes Windows, ele efetivamente acredita que está mudando o documento de lugar e o que efetivamente ocorre é que o *apontador* para o arquivo mudou (*apontador* também é uma metáfora).

Como as metáforas são usadas como modelos, uma metáfora de interface que sugira um modelo incorreto pode causar dificuldades para o usuário. Por exemplo, o clássico caso das funções *cortar* e *colar* dos editores atuais - quando se corta algum objeto ele fica guardado em um buffer (usuários principiantes acham que sumiu) e quando se cola em outra parte o objeto não cola como no real, ele "empurra" (para fazer a real função de colar é preciso marcar e depois colar).

E também, mesmo boas metáforas, não funcionam em sua totalidade. Por exemplo, considerando-se a metáfora da mesa de trabalho (*desktop metaphor*) dificilmente as pessoas conseguem explicar satisfatoriamente o funcionamento de uma função como a de busca, por exemplo, pois ela diverge significativamente da pretendida referência ao mundo real. Nesses casos, metáforas servem como auxiliares ao entendimento atuando como mediadores cognitivos cujos rótulos são menos técnicos que os do jargão computacional.

Mesmo não funcionando sempre, o seu uso crescente, especialmente em interfaces gráficas, favoreceu (ou forçou) a expansão do domínio da área de design de interfaces, com contribuições mais que relevantes de outras especialidades como design gráfico e industrial, lingüística, Psicologia e Educação dentre outras. Portanto, uma importante contribuição da abordagem metafórica foi ter tornado o design e estudo de interfaces uma preocupação inter(multi)(trans)disciplinar. Nas próximas seções deste capítulo, e no decorrer do livro como um todo, estaremos clarificando essa natureza inter(multi)(trans)disciplinar tanto no design como na avaliação de interfaces. No Capítulo 3 voltaremos a discutir sobre o entendimento e uso de metáforas no design de interfaces.

Concluindo, o que vimos nessas primeiras seções é que não se consegue ter um conceito simplista de interface como *os* aspectos do sistema com os quais o usuário tem contato, ou ainda a um pouco mais elaborada linguagem de entrada para o usuário, linguagem de saída para a máquina e um protocolo de interação (ACM CHI'85). Não se pode pensar em interfaces sem considerar o ser humano que vai usá-la, e portanto **interface e interação são conceitos que não podem ser estabelecidos ou analisados independentemente**. E no decorrer deste livro ao nos referirmos a interfaces estaremos focando a interação, o que para nós dá a amplitude desejada ao termo interface, pois pensar somente na "interface" é pensar muito pequeno. As preocupações usuais dos designers de interfaces - criar tipos mais legíveis, melhores barras de rolagem, integrar cor, som e voz - são todas importantes, mas são secundárias. A preocupação primeira deve ser a de melhorar o modo como as pessoas podem usar o computador para pensar e comunicar, observar e decidir, calcular e simular, discutir e projetar.

INTERAÇÃO HUMANO-COMPUTADOR (IHC)

Para que os computadores se tornem amplamente aceitos e efetivamente usados eles precisam ser bem projetados. Isso de maneira alguma quer dizer que o design deve ser adequado a todas as pessoas, mas os computadores devem ser projetados para as necessidades e capacidades de um grupo alvo. Certamente, usuários em geral não devem ser obrigados a pensar sobre como o computador funciona, da mesma forma que o funcionamento mecânico de um carro não é preocupação da maioria das pessoas. Entretanto, a posição dos pedais, direção e câmbio têm muito impacto sobre

o motorista, como também o design de sistemas computacionais têm efeito sobre seus usuários.

Empresas produtoras de software têm despertado para idéia de que a melhora no aspecto físico da interface do usuário proporciona maiores chances de sucesso de mercado. Para explorar essa nova dimensão do produto surgiu um termo amplamente usado - interface amigável ou sistema amigável (*user-friendly)*. Na prática o significado do amigável está associado somente a uma interface, ou melhor, aos elementos na tela serem esteticamente mais agradáveis ou bonitos. Muito embora tenha implicado num avanço com relação às antigas interfaces, muitas empresas usaram o termo simplesmente como um atrativo de mercado. A maioria dos sistemas continua não atendendo às necessidades de seus usuários que tem que lidar com interfaces que mais parecem inimigas. E um outro aspecto, é o de quão pouco adequado é esse termo: primeiro, é desnecessariamente antropomórfico, usuários não precisam de máquinas para serem amigas, eles precisam de máquinas que lhes facilitem na execução de suas tarefas; segundo, significa que as necessidades dos usuários podem ser descritas em apenas uma dimensão, mais ou menos amigável - diferentes usuários têm diferentes necessidades e o que é amigável para um pode ser muito tedioso para outro.

Por outro lado, pesquisadores estavam preocupados em como o uso de computadores pode efetivamente enriquecer o trabalho e a vida das pessoas. Em particular, eles estavam analisando as capacidades e limitações humanas, ou seja, estudando o lado humano da interação com sistemas computacionais. Isso implicava em procurar entender os processos psicológicos das pessoas quando interagem com computadores. Entretanto, com o desenvolvimento da área, em paralelo com avanços tecnológicos, tornou-se claro que outros aspectos ligados ao usuário e ao uso dos computadores precisavam ser incluídos: treinamento; práticas de trabalho; estrutura administrativa e organizacional; relações sociais; saúde; e todos os demais fatores importantes para o sucesso ou fracasso no uso de computadores.

O termo Interação Humano-Computador (IHC) foi adotado em meados dos anos 80 como um meio de descrever esse novo campo de estudo. E como já dissemos, o termo emerge da necessidade de mostrar que o foco de interesse é mais amplo que somente o design de interfaces e abrange todos os aspectos relacionados com a interação entre usuários e computadores.

Muito embora, ainda não exista uma definição estabelecida para IHC, acreditamos que a seguinte definição incorpora o espírito da área no momento: **IHC é a disciplina preocupada com o design, avaliação e implementação de sistemas computacionais interativos para uso humano e com o estudo dos principais fenômenos ao redor deles** A Figura 1.8 tenta expressar o conjunto de componentes contidos nessa definição.

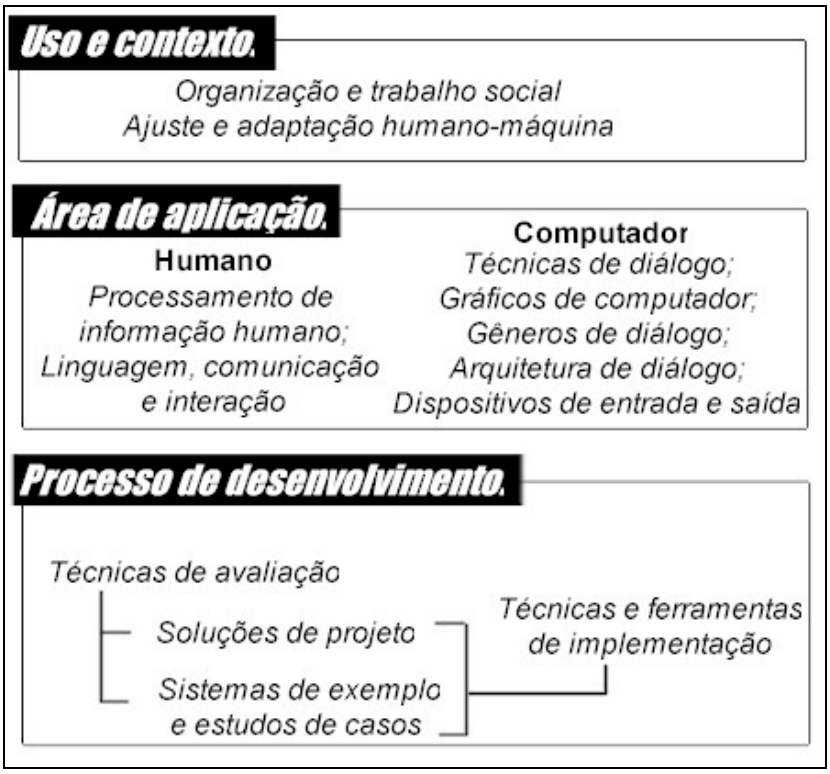


FIGURA 1.8 - INTERAÇÃO HUMANO-COMPUTADOR ADAPTADA DA DESCRIÇÃO DO COMITÊ SIGCHI 1992

Concluindo, IHC trata do design de sistemas computacionais que auxiliem as pessoas de forma a que possam executar suas atividades produtivamente e com segurança. IHC tem, portanto, papel no desenvolvimento de todo tipo de sistema, variando dos sistemas de controle de tráfego aéreo onde segurança é extremamente importante, até sistemas de escritório onde produtividade e satisfação são os parâmetros mais relevantes, até jogos, onde o envolvimento dos usuários é o requisito básico.

DESAFIOS DE IHC

Dado o rápido desenvolvimento da tecnologia, mais os conflitos e compromissos dos objetivos de um design e mais as diferentes componentes (e áreas de estudo) que caracterizam IHC, sem dúvida alguma ela é uma área com ricos desafios.

O desenvolvimento de máquinas mais rápidas e com maior poder de processamento, em conjunto com melhorias de tecnologias de hardware e software não pára, e abre inúmeras possibilidades para IHC. Dispositivos especiais possibilitam ao usuário "pegar" objetos dentro de um espaço virtual, e mesmo movimentar-se através de um espaço de realidade virtual. Aplicações multimídia, onde som, gráficos estáticos e dinâmicos, vídeo e texto são interligados são comuns hoje em dia. Desenvolvimentos recentes em telecomunicações têm possibilitado que grandes quantidades de diferentes tipos de informação possam ser enviadas através de redes. Imagens, vídeo, som e texto podem ser transmitidos com perda mínima de eficiência e qualidade. Informações de bancos de dados existentes em todo o mundo podem ser obtidas pelas pessoas de suas próprias casas. Essas mudanças trazem dois importantes desafios aos designers de IHC (Preece *et al*, 1994):

- Como dar conta da rápida evolução tecnológica?
- Como garantir que os design ofereçam uma boa IHC ao mesmo tempo que exploram o potencial e funcionalidade da nova tecnologia?

Um exemplo clássico desses problemas são os aparelhos de vídeo cassete. Enquanto a maioria das pessoas não tem problema algum em colocar uma fita, iniciar uma gravação ou dar um *play*, adiantar ou atrasar a fita, elas freqüentemente não acham assim tão fácil acertar o *timer* de forma a gravar um programa em um tempo futuro. Para a maioria dos gravadores de vídeo não é óbvio, a partir da interface entre a pessoa e a máquina, como a informação deve ser especificada para o sistema, e muito menos a resposta do sistema (quando existe) é clara. Certamente, se descobre que as coisas não funcionaram bem quando já é muito tarde. Tentando minorar essa dificuldade, muitos aparelhos de vídeo atualmente provêem um display das funções no televisor, mas mesmo assim as dificuldades permanecem (minoradas sem dúvida alguma).

O mesmo acontece com relação aos atuais aparelhos telefônicos. Enquanto as funcionalidades estavam restritas ao suporte de uma conversação tudo ia muito bem. Mas atualmente, a tecnologia permite conversas entre mais que duas pessoas; o sinal de que não está ocupado já não significa mais isso, pois os telefones tem *bip* que permite a interrupção, sem desligar, de uma conversa para atender outra; podemos transferir nossas ligações para outro número; etc. E aí, a mesma interface para dar conta de todas essas novas funcionalidades ficou complexa e não é mais óbvia. As pessoas têm problemas quando tentam operar essas funções e muitas desistem.

Não existe como negar que muitos sistemas computacionais foram projetados com interfaces extremamente pobres. O ponto que precisa ser entendido é que aumentar a funcionalidade não pode ser uma desculpa para um design pobre. Deve ser possível projetar boas interfaces cujos controles têm operações e efeitos relativamente óbvios e que também provêem um feedback imediato e útil.

Um bom exemplo é o dado por Norman (1988) com relação aos carros. Observando os controles dos painéis dos carros atuais podemos ver que eles têm cerca de 100 controles ou mais - dez ou mais para o equipamento de som, 5 ou mais para o sistema de ventilação, outros tantos para as janelas, limpadores de pára-brisa, luzes, para abrir e fechar portas, para dirigir o carro, etc. A maioria das pessoas, com pouca tentativa e erro (quase sempre enquanto dirige) ou após uma rápida olhada no manual, tem poucos problemas em lidar com todo o domínio de funções. Por que isso acontece, se não existe termo de comparação entre o número de funções e controles de um carro e de um gravador de vídeo? O que torna a interface do carro tão boa e a do vídeo tão pobre? Uma das razões é que o feedback nos carros é imediato e óbvio. Também, as pessoas que já dirigiram qualquer carro sabem o que esperar pois, muito embora, os carros sejam diferentes, a posição da maioria dos controles é a mesma ou similar, e símbolos similares são usados para indicar suas funções.

Portanto, os desafios de IHC são evidentes e a procura de soluções estabelece os objetivos da área que ao serem centrados no humano e não na tecnologia são sempre atuais.

OBJETIVOS DE IHC

Os objetivos de IHC são o de produzir sistemas usáveis, seguros e funcionais. Esses objetivos podem ser resumidos como desenvolver ou melhorar a segurança, utilidade, efetividade e usabilidade de sistemas que incluem computadores. Nesse contexto o termo sistemas se refere não somente ao hardware e o software mas a todo o ambiente que usa ou é afetado pelo uso da tecnologia computacional.

Nielsen (1993) engloba esses objetivos em um conceito mais amplo que ele denomina **aceitabilidade de um sistema** (Figura 1.9).

A **aceitabilidade geral** de um sistema é a combinação de sua aceitabilidade social e sua aceitabilidade prática. Como um exemplo de **aceitabilidade social**, podemos mencionar os sistemas atuais de controle das portas de entrada em bancos. Apesar de serem benéficos socialmente pois tentam impedir situações de assalto onde os usuários dos bancos ficam em sério risco, não são aceitos socialmente pois levam a que qualquer pessoa que queira entrar no banco tenha que esbarrar na porta trancada por inúmeras vezes até se desfazer de todo e qualquer objeto suspeito (o problema é que não se sabe quais os objetos que impedem a entrada).

A **aceitabilidade prática** trata dos tradicionais parâmetros de custo, confiabilidade, compatibilidade com sistemas existentes, etc., como também da categoria denominada ***"usefulness"***

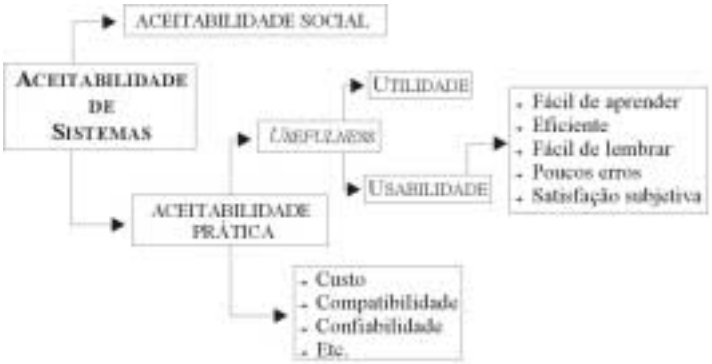


FIGURA 1.9 - ATRIBUTOS DE ACEITABILIDADE DE SISTEMAS (ADAPTADO DE NIELSEN, 1993)

"Usefulness" refere-se ao sistema poder ser usado para atingir um determinado objetivo. Novamente essa categoria é uma combinação de duas outras: utilidade e usabilidade. **Utilidade** deve verificar se a funcionalidade do sistema faz o que deve ser feito, ou seja, se um jogo efetivamente diverte e um software educacional auxilia o aprendizado. **Usabilidade** é a questão relacionada a quão bem os usuários podem usar a funcionalidade definida e este é um conceito chave em IHC, que trataremos mais detalhadamente ainda neste capítulo.

Portanto, a aceitabilidade de um sistema tem muitos componentes (daí a complexidade da tarefa), e IHC tem, de certa forma, que atender aos compromissos de todas essas categorias. Mas, como temos afirmado e reafirmado, a pesquisa de IHC é fundada na crença de que o centro e ponto básico de análise são as pessoas usando um sistema computacional. Suas necessidades, capacidades e preferências para executar diversas tarefas devem informar os meios como os sistemas devem ser projetados e implementados. As pessoas não devem ter que mudar radicalmente para se adequar ao sistema, o sistema sim deve ser projetado para se adequar aos seus requisitos.

A MULTI(INTER)(TRANS)DISCIPLINARIDADE EM IHC

Estabelecidos os objetivos de IHC tem-se a parte mais difícil que é a de que forma conseguir estes objetivos. Isso envolve uma perspectiva multidisciplinar, ou seja, resolver os problemas de IHC analisando diferentes perspectivas em seus multifacetados fatores: segurança, eficiência e produtividade, aspectos sociais e organizacionais, etc.

Um resumo dos principais fatores que devem ser levados em conta pode ser visto na Tabela 1.2 (Preece, 1994). Primeiramente, tem-se os fatores relacionados com o usuário como o conforto, saúde, ambiente de trabalho ou ergonomia do equipamento

a ser utilizado. Analisar esses fatores é tarefa bastante complexa pois eles não são independentes, interagem fortemente uns com os outros.

Outro ponto que em muito aumenta a complexidade da análise dos fatores ligados ao usuário, é que eles não são homogêneos em termos de requisitos e características pessoais. Humanos compartilham muitas características físicas e psicológicas, mas são bastante heterogêneos em termos de qualidades como habilidades cognitivas e motivação. Essas diferenças individuais têm importância fundamental no design da interface de um sistema computacional.

<table>
<tr><td colspan="2">FATORES ORGANIZACIONAIS
TREINAMENTO, POLÍTICAS, ORGANIZAÇÃO DO TRABALHO, ETC.</td><td>FATORES AMBIENTAIS
BARULHO, AQUECIMENTO, VENTILAÇÃO, LUMINOSIDADE, ETC.</td></tr>
<tr><td>SAÚDE E SEGURANÇA
estresse, dores de cabeça, perturbações musculares, etc.</td><td>capacidades e processos cognitivos
O USUÁRIO
motivação, satisfação, personalidade, experiência, etc.</td><td>CONFORTO
posição física, layout do equipamento, etc.</td></tr>
<tr><td colspan="3">INTERFACE DO USUÁRIO
dispositivos de entrada e saída, estrutura do diálogo, uso de cores, ícones, comandos, gráficos, linguagem natural, 3-D, materiais de suporte ao usuário, multimídia, etc.</td></tr>
<tr><td colspan="3">TAREFA
fácil, complexa, nova,
alocação de tarefas, repetitiva,
monitoramento, habilidades, componentes, etc.</td></tr>
<tr><td colspan="3">RESTRIÇÕES
custos, orçamentos, equipe,
equipamento, estrutura do local de trabalho, etc.</td></tr>
<tr><td colspan="3">FUNCIONALIDADE DO SISTEMA
hardware, software, aplicação</td></tr>
<tr><td colspan="3">PRODUTIVIDADE
aumento da qualidade, diminuição de custos, diminuição de erros,
diminuição de trabalho, diminuição do tempo de produção,
aumento da criatividade , oportunidades para idéias criativas em direção a novos produtos, etc.</td></tr>
</table>

TABELA 1.2 - FATORES EM IHC (ADAPTADO DE PREECE, 1994, P.31)

Voltemos ao exemplo dos carros, considerando agora os bancos. Se todos tivessem o mesmo formato, nenhuma dificuldade haveria para o designer projetar sempre o banco ideal. Comparado com diferenças psicológias, as diferenças físicas podem ser

consideradas triviais de lidar. Um modo de tratar essa diversidade é projetar sistemas flexíveis que possam ser "customizados" de forma a se adequar às necessidades individuais. Isto, de certa forma, está sendo feito no design dos bancos nos carros atuais, onde um bom número de modos para ajuste são disponíveis. Também, sistemas computacionais, como editores de texto por exemplo, oferecem atualmente uma série de opções para se adequar à experiência e preferência de usuários.

Portanto, na análise dos fatores humanos envolvidos em IHC diversas disciplinas são necessárias(Figura 1.10).

FIGURA 1.10 - DISCIPLINAS QUE CONTRIBUEM EM IHC (ADAPTADO DE PREECE, 1994, P. 38)

Temos as principais: Psicologia Cognitiva, Psicologia Social e Organizacional, Ergonomia (termo europeu) ou Fatores Humanos (termo americano) e a Ciência da Computação. Outras áreas de estudo que tem tido uma crescente influência em IHC incluem: Inteligência Artificial, Linguística, Psicologia, Filosofia, Sociologia, Antropologia, Engenharia e Design.

CIÊNCIA DA COMPUTAÇÃO contribui provendo conhecimento sobre as possibilidades da tecnologia e oferecendo idéias sobre como explorar todo o seu potencial. Também os profissionais de computação têm se preocupado em desenvolver ferramentas de software auxiliares ao design, implementação e manutenção de sistemas: linguagens de programação, ferramentas de prototipação, sistemas de gerenciamento de interfaces de usuário (UIMS), ambientes de design de interfaces de usuário (UIDE), ferramentas de *debugging* e teste, etc. Alguns esforços têm sido feitos no sentido de prover métodos rigorosos de analisar a forma como IHC é projetada e incorporada em sistemas, que incluem arquiteturas de sistemas, abstrações e notações. Conceitos de reuso e de engenharia reversa também são utilizados em IHC. Em particular, tem havido a preocupação de prover meios para que designers iniciantes possam reusar trabalhos de colegas mais experientes, como bibliotecas de código, por exemplo. Os sofisticados sistemas gráficos usados em visualização e em realidade virtual também são resultados da ciência da computação.

PSICOLOGIA COGNITIVA. A preocupação principal da Psicologia é entender o comportamento humano e os processos mentais subjacentes. A Psicologia Cognitiva adotou a noção de processamento de informação como modelo para o comportamento humano e tenta colocar tudo que vemos, sentimos, tocamos, cheiramos, etc. , em termos desse modelo. Como poderá ser visto no Capítulo 2, importantes tópicos de IHC são o estudo da percepção, atenção, memória, aprendizagem, solução de problemas, etc. O objetivo da Psicologia Cognitiva tem sido o de caracterizar esses processos em termos de suas capacidades e limitações. Por exemplo, uma das principais preocupações da área nos anos 60 e 70 era identificar a quantidade de informação que podia ser processada e lembrada de uma só vez. Recentemente, existe a preocupação em caracterizar o modo como as pessoas trabalham entre si e com vários artefatos, entre eles o computador. Um dos principais resultados desses estudos é a cognição distribuída. Psicólogos cognitivistas têm se preocupado em aplicar princípios psicológicos em IHC usando uma variedade de métodos: desenvolvimento de *guidelines*, uso de modelos para predizer a performance humana no uso de computadores, métodos empíricos para testar sistemas computacionais, etc.

PSICOLOGIA SOCIAL tem como preocupação estudar a natureza e causas do comportamento humano no contexto social. Pode-se resumir as preocupações básicas da Psicologia Social em quatro pontos (Vaske e Grantam, 1990):

- a influência de um indivíduo nas atitudes e comportamentos de outra pessoa
- impacto de um grupo sobre o comportamento e as atitudes de seus membros
- impacto de um membro nas atividades e estrutura de um grupo
- relacionamento entre estrutura e atividades de diferentes grupos

E a tecnologia desempenha um papel importante em todos esses aspectos.

A **PSICOLOGIA ORGANIZACIONAL** dá aos designers o conhecimento sobre estruturas organizacionais e sociais e sobre como a introdução de computadores influencia práticas de trabalho. Em grandes organizações, por exemplo, o computador serve tanto como meio de comunicação, quanto para fazer a folha de pagamento e contabilidade em geral, para controlar entrada e saída de pessoas, etc. Isso envolve entender a estrutura e funcionamento de organizações em termos de autoridade e poder, tamanho e complexidade, eficiência, fluxo de informação, tecnologia, práticas de trabalho, ambiente de trabalho e contexto social. Modelos de mudanças organizacionais com a inclusão da tecnologia são bastante úteis a esse entendimento.

FATORES HUMANOS, OU ERGONOMIA, teve um grande desenvolvimento a partir da segunda grande guerra, atendendo a demanda de diversas disciplinas. Seu objetivo é conceber e fazer o design de diversas ferramentas e artefatos para diferentes ambientes de trabalho, domésticos e de diversão, adequados às capacidades e necessidades de usuários. O objetivo é maximizar a segurança, eficiência e confiabilidade da performance do usuário, tornando as tarefas mais fáceis e aumentando os sentimentos de conforto e satisfação. As primeiras contribuições dos

especialistas em fatores humanos para IHC foram no design do hardware (teclados mais ergonômicos, posições do vídeo, etc.) e nos aspectos de software que poderiam resultar em efeitos fisiológicos adversos nos humanos, como a forma da apresentação de informação na tela do vídeo.

LINGÜÍSTICA é o estudo científico da linguagem (Lyons, 1970). Muita atenção tem sido dada atualmente aos resultados da lingüística como fontes de conhecimento importantes para IHC. O uso imediato e mais tradicional é o de explorar a estrutura da linguagem natural na concepção de interfaces, principalmente para facilitar o acesso e consulta a bases de dados. Também na concepção de linguagens de programação mais fáceis de serem aprendidas resultados da lingüística estão presentes (por exemplo, na linguagem de programação Logo, voltada para a Educação). Estudos derivados, que consideram o estudo da linguagem enquanto forma de comunicação, não apenas textual, têm tido muita relevância hoje em dia em IHC (Semiótica e Engenharia Semiótica, que serão tratadas no Capítulo 3, são um exemplo). Também na internacionalização de interfaces e localização de software a lingüística tem tido um papel cada vez mais importante. Internacionalização é a preocupação em isolar os fatores culturais de um produto (por exemplo, textos, ícones, datas etc.) de outros que podem ser considerados genéricos culturalmente. Localização é exatamente o processo de colocar os aspectos culturais em um produto previamente internacionalizado (Russo e Boor, 1993).

INTELIGÊNCIA ARTIFICIAL (IA) é um ramo da ciência da computação cujo objetivo é desenvolver sistemas computacionais que exibam características que nós associamos com inteligência no comportamento humano. A preocupação central é com o desenvolvimento de estruturas de representação do conhecimento que são utilizadas pelo ser humano no processo de solução de problemas. Métodos e técnicas de IA, tais como o uso de regras de produção, têm sido usados por IHC no desenvolvimento de sistemas especialistas e tutores com interfaces inteligentes. IA também se relaciona com IHC no processo de interação dos usuários com interfaces inteligentes no sentido do uso de linguagem natural (textual e falada), na necessidade do sistema ter que justificar uma recomendação, nos sistemas de ajuda contextualizados e que efetivamente atendam às necessidades dos usuários, etc. Atualmente, grande ênfase tem sido dada no desenvolvimento de agentes de interfaces inteligentes, que auxiliam os usuários na navegação, busca de informação, organização da informação, etc. O objetivo no uso desses agentes é o de reduzir a sobrecarga cognitiva que muitos usuários têm atualmente ao lidar com a quantidade de informação apresentada, na maioria das vezes, de forma hipertextual.

FILOSOFIA, SOCIOLOGIA E ANTROPOLOGIA são, das disciplinas que contribuem com IHC, as tradicionalmente denominadas *soft sciences* (Preece, 1994). Com isso, se está querendo dizer que elas não estão diretamente envolvidas com o design real de um sistema computacional do mesmo modo que as *hard sciences* que oferecem métodos, técnicas e implementações. Elas estão mais diretamente envolvidas com os

desenvolvimentos da tecnologia de informação e com a transferência de tecnologia. Isso na verdade vem mudando, pois atualmente métodos da Sociologia e Antropologia têm sido aplicados no design e avaliação de sistemas. Uma dessas técnicas é a etno-metodologia onde a premissa básica é não assumir um modelo *a priori* do que vai acontecer quando as pessoas usam o computador, ao invés disso, analisar o comportamento na observação do que acontece durante o uso em seu contexto real de uso. Portanto, a ênfase é em entender o que acontece quando as pessoas se comunicam entre si ou com as máquinas, enquanto e depois que isso acontece, e não modelar e predizer de antemão como o faz a Psicologia Cognitiva. A razão da aplicação desses métodos na análise de IHC é a de que uma descrição mais precisa da interação entre usuários, seu trabalho, a tecnologia em uso e no ambiente real de uso precisa ser obtida. Trabalho cooperativo auxiliado pelo computador (CSCW) o qual objetiva prover ferramentas de software que possibilitem a execução cooperativa (compartilhando software e hardware) de tarefas é uma área de aplicação e desenvolvimento que depende diretamente do resultado dessa descrição.

ENGENHARIA é uma ciência aplicada direcionada à construção e testes empíricos de modelos. Basicamente, a Engenharia usa os resultados da ciência em geral na produção de artefatos. Na maioria dos aspectos, a grande influência da engenharia em IHC tem sido via Engenharia de Software.

DESIGN tem oferecido a IHC conhecimento mais que evidente, como por exemplo na área de design gráfico. Muitos autores, afirmam que o envolvimento e o crescente interesse de designers gráficos no projeto de telas de sistemas computacionais consolidou IHC como uma área de estudo. O processo de design de IHC também foi influenciado pela prática de design gráfico. Por exemplo, a prática de gerar diversas alternativas para serem avaliadas em sessões de *brainstorming* entre colegas logo no início de um projeto tem sido adotada atualmente por um grande número de designers de interfaces. Com o advento da WEB, como uma nova mídia de comunicação, dificilmente se tem uma equipe de desenvolvimento que não tenha um designer gráfico.

A contribuição dessas disciplinas em IHC certamente é uma via de duas mãos, ou seja, certamente IHC alterou também a prática em cada uma dessas disciplinas. Por exemplo, hoje em dia não existe engenheiro ou designer que não faça uso de ferramentas de design tanto para produzir projetos arquitetônicos ou mecânicos quanto para fazer o projeto de uma nova cadeira. E de modo mais geral, em todas essas áreas, ferramentas de visualização, busca, compilação, análise de informação têm sido geradas e amplamente utilizadas.

PRINCÍPIOS DE DESIGN

Se pensamos na complexidade da maioria dos sistemas computacionais, vemos que o potencial de se ter uma precária IHC é bastante alto. Daí alguns autores definirem príncipios básicos que ajudem a garantir uma boa IHC.

PARTINDO DOS OBJETOS QUE NOS CERCAM

Norman (1988), partindo da experiência de observar e vivenciar as frustações que as pessoas experimentam com objetos do cotidiano que não conseguem saber como usar, com embalagens que parecem impossíveis de serem abertas, com portas que mais parecem uma armadilha, com máquinas de lavar e secadoras que têm se tornado cada vez mais poderosas e confusas, identifica alguns princípios básicos de um bom design, que segundo ele, constituem uma forma de Psicologia - a Psicologia de como as pessoas interagem com objetos. Os quatro princípios (visibilidade e *affordances*; bom modelo conceitual; bons mapeamentos e feedback), como veremos a seguir, são altamente interrelacinados e difíceis de serem tratados e estudados isoladamente.

- **Visibilidade e *Affordances***
  O usuário necessita ajuda. Apenas as coisas necessárias têm que estar visíveis: para indicar quais as partes podem ser operadas e como, para indicar como o usuário interage com um dispositivo. Visibilidade indica o mapeamento entre ações pretendidas e as ações reais. Indica também distinções importantes - por exemplo, diferenciar a vasilha do sal da do açúcar.

  A visibilidade do efeito das operações indica se a operação foi feita como pretendida, como por exemplo, se as luzes foram acesas corretamente, se a temperatura de um forno foi ajustada corretamente, etc. A falta de visibilidade é que torna muitos dispositivos controlados por computadores tão difíceis de serem operados.

  A título de ilustração, vamos considerar um exemplo muito simples já mencionado anteriormente, o das portas. Quantos de nós já experimentamos frustrações no uso de portas cuja funcionalidade é extremamente simples - abrir e fechar, e nada mais. Muitas vezes empurramos portas que deveríamos puxar, abrimos para a direita quando deveria ser para a esquerda, empurramos portas que deveriam ser deslizadas em alguma direção, que também muitas vezes não descobrimos qual é. As partes corretas deveriam estar visíveis. Designers deveriam prover sinais que claramente indicassem que uma porta deveria ser empurrada, simplesmente colocando a barra de empurrar em um dos lados da porta e nada no outro. Os pilares de suporte deveriam estar visíveis e tudo isso

sem nenhum prejuízo da estética tão largamente procurada. A barra horizontal para empurrar e os pilares são sinais naturais e portanto naturalmente interpretados, conduzindo ao que se denomina de design natural (Norman, 1988).

Outro exemplo clássico de falta de visibilidade, que também já mencionamos, é o dos modernos telefones, com múltiplas funções e com uma interface de uso que não as deixa visíveis. E um exemplo favorável é o dos carros, que oferecem uma boa visibilidade na maioria de suas inúmeras funções.

A mente humana é extraordinária no processo de dar sentido ao mundo. Considere os objetos - livros, rádios, eletrodomésticos, máquinas de escritório, etc. - que fazem parte de nossa vida. Objetos com um bom design são fáceis de interpretar e entender. Eles contém "dicas" visíveis de sua operação. Objetos com design pobre são difíceis e frustantes de usar. Eles não provêem indicações ou o que é muito pior, provêem, muitas vezes, falsas "dicas".

Affordance é o termo definido para se referir às propriedades percebidas e propriedades reais de um objeto, que deveriam determinar como ele pode ser usado. Uma cadeira é para sentar e também pode ser carregada. Vidro é para dar transparência, e aparenta fragilidade. Madeira dá solidez, opacidade, suporte, e possibilidade de escavar. Botões são para girar, teclas para pressionar, tesouras para cortar, etc. Quando se tem a predominância da *affordance* o usuário sabe o que fazer somente olhando, não sendo preciso figuras, rótulos ou instruções. Objetos complexos podem requerer explicações, mas objetos simples não. Quando estes necessitam rótulos ou instruções é porque o design não está bom.

- **Bom modelo conceitual**
  Um bom modelo conceitual permite prever o efeito de ações. Sem um bom modelo conceitual opera-se sob comando, cegamente. Efetua-se as operações receitadas, sem saber que efeitos esperar ou, o que fazer se as coisas não derem certo. Conforme as coisas vão dando certo, aprende-se a operar. Agora, quando as coisas dão errado ou quando se depara com situações novas necessita-se de um maior entendimento, de um bom modelo.

  Consideremos o exemplo de uma tesoura. Mesmo que nunca tenhamos visto uma anteriormente, é claro o seu limitado número de funções possíveis. Os buracos deixam claro que algo deve ser colocado neles, e a única coisa lógica de se colocar e que pode encaixar são os dedos. Os buracos têm *affordances*, que possibilitam os dedos serem inseridos. O tamanho dos buracos provêem restrições que limitam quais dedos podem ser usados: o buraco maior sugere diversos dedos e o menor apenas um. O mapeamento entre os buracos e os dedos é então sugerido e restringido pelos buracos. Entretanto, a operação não é limitada à colocação dos dedos corretos. A tesoura irá funcionar com qualquer

dedo. Consegue-se entender a tesoura e seu funcionamento porque suas partes são visíveis e as implicações claras. O **modelo conceitual** é portanto claro, e até óbvio, e existe um efetivo uso de *affordances*.

Um contra exemplo, é o de um relógio digital simples, com dois ou até quatro botões no mostrador. Para que servem esses botões? Como descobrir se são de puxar e empurrar e não girar? Como acertar a hora, a data? Não existe um relacionamento evidente entre os controles e suas funções, nenhum mapeamento aparente.

Para objetos do dia a dia, modelos conceituais podem ser bastante simples, mas quando consideramos, a complexidade de sistemas computacionais a relevância de um bom modelo é mais que óbvia.

- **Bons mapeamentos**
  Mapeamento é o termo técnico para denotar o relacionamento entre duas entidades. No caso de interfaces, indica o relacionamento entre os controles e seus movimentos e os resultados no mundo. Vamos novamente recorrer ao exemplo dos carros e os mapeamentos envolvidos em dirigir um carro. Quando queremos ir para a direita, devemos virar o volante também para a direita (sentido horário). O usuário identifica dois mapeamentos: o controle que afeta a direção e que o volante precisa ser virado em uma de duas direções. Ambos são arbitrários, mas a roda e o sentido horário são escolhas naturais: visíveis, muito relacionada ao resultado esperado, e provêem um feedback imediato. O mapeamento é facilmente aprendido e sempre lembrado.

  Mapeamentos naturais, aqueles que aproveitam analogias físicas e padrões culturais, levam ao entendimento imediato. Por exemplo, é comum designers utilizarem analogias especiais: para mover um objeto para cima, move-se o controle também para cima (Norman, 1988).

  Problemas de mapeamento são muitos e uma das principais causas das dificuldades que os usuários encontram no uso de objetos. Também retomando, consideremos os telefones. Suponha que se deseja redirecionar as ligações de um número para outro. As instruções são geralmente do seguinte tipo: tecle #, em seguida o número 9 e finalmente o número para o qual deseja desviar suas chamadas. Uma descrição incompleta de um procedimento arbitrário. O que acontece se eu errar no meio do caminho o que faço? Por que #? Por que 9? E a ausência de feedback é outro aspecto. Como sei se deu certo ou não?

  Um objeto é fácil de ser usado quando existe um conjunto visível de ações possíveis, e os controles exploram mapeamentos naturais. O princípio é simples mas raramente incorporado aos design.

- **Feedback**
  Retornar ao usuário informação sobre as ações que foram feitas, quais os resultados obtidos, é um conceito conhecido da teoria da informação e controle. Imagine falar com uma pessoa sem ouvir sua própria voz (a famosa ausência de "retorno" que os músicos tanto reclamam nos palcos), ou desenhar com um lápis que não risca, ou seja, sem nenhum feedback. Acreditamos não ser necessário retomar os exemplos clássicos já amplamente discutidos: carros (bom feedback) e telefones (nenhum feedback para algumas funções mais avançadas). Exemplificando com interfaces computacionais, quantos de nós mandamos imprimir documentos em impressoras de rede e sempre nos deparamos com problemas clássicos: qual foi mesmo a impressora? a impressão já terminou ou não? o documento foi mesmo impresso ou houve algum problema? acabou o papel da impressora (que está em outra sala muitas vezes bem distante) antes do término da impressão?

Um aspecto interessante e que pode ser depreendido desses exemplos que mencionamos é o que Norman (1988) chama de **paradoxo da tecnologia**. A tecnologia oferece potencial para tornar nossa vida mais simples e agradável, e cada nova tecnologia traz mais benefícios. E ao mesmo tempo adiciona tamanha complexidade que faz aumentar nossa dificuldade e frustação. O relógio é um bom exemplo disso. Ninguém tinha problemas com os clássicos relógios cuja única função era marcar as horas e com um único botão conseguiamos fazer todos os ajustes necessários. Os relógios digitais atuais ampliaram em muito as funcionalidades básicas: eles marcam data, dão alarme, são cronômetros, mostram hora no mundo todo e os mais modernos mostram inclusive o horário mundial da WEB. Mas adicionar todas essas funções causa problemas. Como fazer o design de um relógio com tantas funções e ao mesmo tempo limitar seu tamanho, custo e complexidade de uso? Quantos botões deveriam ter os relógios para torná-los fáceis de aprender e operar? Não existem respostas simples. Sempre que o número de funções excede o número de controles, o design torna-se arbitrário e não natural, e complicado. A mesma tecnologia que simplifica a vida provendo um maior número de funcionalidades em um objeto, também a complica tornando muito mais difícil aprender, e usar. Esse é o paradoxo da tecnologia e o grande desafio dos designers é minimizar esses efeitos.

USABILIDADE DE SISTEMAS COMPUTACIONAIS

Nielsen (1993) já mais direcionado para sistemas computacionais explora o design e propõe princípios que levem a um aumento da usabilidade, que como vimos é um dos critérios que definem a aceitabilidade de um sistema (Figura 1.9). Nielsen explicita seus princípios de design a partir de alguns *slogans*, que ele define como *slogans de usabilidade*. A seguir apresentamos alguns deles:

- **Sua melhor tentativa não é boa o suficiente**
  É impossível fazer o design de uma interface ótima simplesmente baseado em nossas melhores idéias. Usuários tem um potencial infinito para mal interpretar elementos de interface e para fazer suas tarefas de modo diferente do que imaginamos. Portanto, o design é sempre melhor se trabalhamos baseados no entendimento do usuário e de
  suas tarefas. Sempre temos que nos preocupar em validar ou avaliar nossos design usando as diferentes formas de avaliação existentes (o Capítulo 4 trata especificamente de avaliação de interfaces) e estar abertos para efetuar um redesign a partir dos resultados dessas avaliações.

- **Usuário está sempre certo**
  A atitude do designer quando verifica que o usuário tem problemas de interação com um determinado aspecto da interface, não deve ser a de julgar que o usuário é ignorante ou então, que ele não tentou o suficiente ou ainda, deixar passar que um dia o usuário aprende. Por exemplo, quando escrevemos um determinado procedimento de operação e verificamos que o usuário sempre erra em uma determinada parte do procedimento, certamente a solução não é colocar um aviso em destaque do tipo LEIA CUIDADOSAMENTE ESSE TRECHO. O que deve ser feito é aceitar que o texto está mal escrito e que precisa ser reformulado. Portanto, o designer de interfaces deve adquirir uma certa humildade e aceitar a necessidade de modificar uma "grande idéia" de forma a resolver problemas dos usuários.

- **Usuário não está sempre certo**
  Também não se deve ir ao extremo de construir uma interface somente a partir do que os usuários gostariam. Usuários freqüentemente não sabem o que é bom para eles. Qualquer um de nós teria dificuldades em prever como gostaríamos de interagir com um sistema em potencial com o qual não temos nenhuma experiência. Temos a tendência de rejeitar *a priori* qualquer grande inovação em objetos com os quais estamos familiarizados e que atendem satisfatoriamente nossas necessidades.

- **Usuários não são designers**
  Uma solução simples para atender a diversidade de usuários seria a de prover interfaces flexíveis que pudessem ser amplamente customizadas e aí cada usuário teria exatamente a interface que melhor lhe satisfizesse (análogo aos bancos de carros modernos mencionados anteriormente). Estudos demonstram que usuários novatos não customizam suas interfaces, mesmo quando essas facilidades estão disponíveis (Jorgensen e Sauer, 1990). Mas existem alguns outros bons motivos para não se dar à customização uma importância indevida: primeiro, customização é fácil somente se puder produzir um design coerente a partir do conjunto de opções disponíveis; segundo, o processo de customização

também vai exigir uma interface e portanto adiciona complexidade; terceiro, muita customização leva a que cada usuário tenha uma interface muito diferente de outro usuário (isso dificulta, por exemplo, o pedido a ajuda entre colegas que é um dos principais métodos de *help* usado tanto por novatos como por especialistas (Sellen e Nicol, 1990)); e quarto, usuários nem sempre adotam as decisões de design mais apropriadas.

- **Designers não são usuários**
  Designers são humanos e certamente usam computadores, mas são diferentes de usuários em diversos aspectos básicos: a experiência computacional e o conhecimento dos fundamentos conceituais do design do sistema. Conseqüentemente o designer olha uma determinada tela ou uma determinada mensagem e acredita que são perfeitamente claras e adequadas, mesmo que sejam incompreensíveis para quem não conhece o sistema. Conhecer sobre um sistema é uma via de mão única, impossível voltar e fazer o papel de um novato.

- **Menos é mais (*less is more*)**
  Uma das frequentes soluções de design que têm sido adotadas é colocar no sistema todas as opções e características imagináveis, pois se tudo está disponível então todos ficarão satisfeitos. Essa tendência é verificada nos softwares, como editores de texto por exemplo, que a cada nova versão tem dobrado de tamanho gerando o fenômeno denominado '*fatware'* (Perratore *et al*, 1993). Cada elemento em uma interface acarreta uma sobrecarga ao usuário que tem que considerar se o usa ou não. Ter poucas opções, as necessárias à tarefa, geralmente significa uma melhor usabilidade, pois o usuário pode se concentrar em entender essas poucas opções.

- ***Help* não ajuda (*help doesn´t*)**
  Muitas vezes, senão na maioria delas, vemos usuários perdidos tentando encontrar informação na enorme quantidade de material de *help* que acompanha um sistema, e quando a encontra não consegue entendê-la. Também, a existência de *helps* acrescenta mais complexidade à interface e na maioria das vezes sem grande efetividade. Em qualquer situação, deve-se ter claro que a existência de um *help* não pode ser usada como desculpa para um design ruim. Sempre é melhor o usuário poder operar um sistema sem ter que usar um *help* e o design deve usar isso como um requisito básico.

Usabilidade é definida em função de múltiplos componentes e é tradicionalmente associada com cinco atributos de usabilidade (Nielsen, 1993):

- **Facilidade de aprendizagem (*learnability*)**
  O sistema precisa ser fácil de aprender de forma que o usuário possa rapidamente começar a interagir. Segundo Nielsen, é o mais importante atributo de usabilidade, por ser a primeira experiência que qualquer usuário tem com um

sistema. Certamente existem sistemas para aplicações altamente especializadas e complexas onde se prevê um extenso trabalho de treinamento em seu uso, mas na maioria dos casos um sistema deve ser fácil de aprender.

Quando se analisa a facilidade de aprendizagem, é preciso ter em mente que geralmente o usuário não aprende toda uma interface antes de começar a usá-la. Pelo contrário, o aprendizado ocorre do uso. Portanto, esse fator é avaliado em função do tempo que o usuário demora para atingir um suficiente grau de proficiência na execução de suas tarefas.

- **Eficiência**
  O sistema precisa ser eficiente no uso, de forma que uma vez aprendido o usuário tenha um elevado nível de produtividade. Portanto, eficiência refere-se a usuários experientes, após um certo tempo de uso. Um modo típico de avaliar esse atributo é definir de alguma forma o que significa um usuário experiente e avaliar um grupo desses executando tarefas típicas de um sistema.

- **Facilidade de relembrar (*memorability*)**
  O sistema precisa ser facilmente relembrado, de forma que o usuário ao voltar a usá-lo depois de um certo tempo não tenha novamente que aprendê-lo. Esse atributo tanto se refere a usuários casuais (que é uma categoria com um número grande de usuários na maioria dos sistemas) como para aqueles sistemas utilitários que são inerentemente usados em periodos específicos como os sistemas para confecção de relatórios de atividades trienais, de imposto de renda, etc. Certamente, aumentar a facilidade de aprendizagem também torna a interface mais fácil de ser relembrada, mas tipicamente usuários que retornam a um sistema são diferentes dos usuários principiantes.

  Raramente se avalia esse item, mas é notável a preocupação com ele nas modernas interfaces gráficas onde tudo que for possível é visível. Usuários desses sistemas não precisam lembrar o que está disponível, pois o sistema sempre o relembra quando necessário.

- **Erros**
  Neste contexto, erro é definido como uma ação que não leva ao resultado esperado, um "engano" portanto. O sistema precisa ter uma pequena taxa de erros, ou seja, o usuário não pode cometer muitos erros durante o seu uso e, em errando, deve ser fácil a recuperaração, sem perda de trabalho. Erros catastróficos (o usuário perder seu trabalho, não perceber que errou, etc.) não podem ocorrer.

- **Satisfação subjetiva**
  Os usuários devem gostar do sistema, ou seja, deve ser agrádável de forma que o usuário fique satisfeito ao usá-lo. Esse atributo é muito relevante quando se

considera sistemas usados fora do ambiente de trabaho, tais como jogos, sistemas domésticos em geral, etc. Para esses sistemas o entretenimento e envolvimento são muitas vezes, valores mais importantes que velocidade de processamento (claro, quando esta não compromete o resultado).

Nielsen (1993) ressalta que satisfação subjetiva como atributo de usabilidade é diferente dos estudos que avaliam atitudes gerais das pessoas com relação a computadores. Esses estudos são efetuados no contexto da aceitabilidade social de computadores (LaLomia e Sidowski, 1991) e não como atributo de usabilidade. Entretanto, certamente os sentimentos que as pessoas têm com relação a computadores em muito afeta sua interação com um determinado sistema. Pouco se conhece sobre a relação entre atributos de um determinado sistema e a atitude geral de uma pessoa com relação a computadores. Um dado importante é que usuários que percebem que têm um alto grau de controle sobre os computadores têm atitudes positivas com relação a ele (Kay, 1989). E dar controle ao usuário não é das soluções atuais de design mais freqüentes, muito pelo contrário.

Satisfação subjetiva pode ser medida simplesmente perguntando ao usuário sobre suas opiniões subjetivas. Para um único usuário o resultado desse questionamento é subjetivo, mas quando se considera muitos usuários, a média das respostas passa a ser uma medida objetiva. Isso é o que é feito na maioria dos estudos de usabilidade. Tipo de questionários usados e formas de avaliar satisfação subjetiva são tratados no Capitulo 4.

O que se pode depreender dos princípios de usabilidade é que eles tratam basicamente de dois aspectos: a tarefa e as características individuais dos usuários. Portanto, mais uma vez, conhecer o usuário é fundamental para se fazer o design de um sistema usável. Entender os principais modos de classificar usuários ajuda a fazer um bom design que atenda a maior diversidade desses.

Na análise do usuário sua experiência é um fator relevante, e essa experiência deve ser analisada em três dimensões: com relação ao uso do sistema, com relação ao uso de computadores em geral e com relação ao domínio da aplicação. Nielsen (1993) apresenta essa diferenciação em um gráfico que ele denomina de *cubo do usuário* (Figura 1.11).

O importante é ressaltar que o uso do sistema altera, e tem como um de seus objetivos, a categoria do usuário (um novato não é um eterno novato) e isso tem importantes implicações no design. Algumas interfaces são e devem ser projetadas com ênfase apenas em novatos - sistemas de informação sobre museus ou quiosques em parques e exposições, sistemas que necessariamente são alterados anualmente, etc. - que são categorias de sistemas onde a facilidade de aprendizagem é o requisito mestre. Mas a maioria das interfaces é projetada tendo em vista tanto os expertos como os novatos e portanto precisa acomodar ambos os estilos (não esquecendo que

novatos são futuros especialistas). Freqüentemente, interfaces que são boas para novatos também o são para especialistas, mas sempre é possível prover múltiplos estilos de interação, de tal forma que os usuários iniciem aprendendo um estilo mais fácil e depois migrem para outro mais eficiente.

FIGURA 1.11 - TRÊS DIMENSÕES DAS DIFERENTES EXPERIÊNCIAS DE USUÁRIOS (ADAPTADO DE NIELSEN, 1993, P.44)

Um modo típico de permitir essa evolução natural do aprendizado inicial para um uso mais eficiente é o uso de aceleradores na interface. Aceleradores são elementos de interface que permitem que usuários realizem tarefas freqüentes de forma mais rápida. Exemplos são as teclas de função, abreviação de nome de comandos, uso de duplo-clique para ativar objetos, etc. Muitos sistemas também provêem dois conjuntos de menu, os menus curtos para novatos e os longos para usuários mais experientes. Isto permite ao sistema oferecer um maior conjunto de opções para os usuários mais avançados sem confundir os principiantes. Importante é estar ciente de que ter ambos os estilos de interação aumenta a complexidade da interface e pode vir a ser um problema. Portanto é importante, fazer o design da interface de tal modo que o principiante não esteja exposto ao modo especialista. Por exemplo, sistemas que possibilitam o uso de comandos em sua forma abreviada devem ser cuidadosos no sentido de prover também a forma extensa em mensagens de erro e *helps*, por exemplo.

A **experiência com computadores** também tem um impacto no design da interface. Usuários experientes em um amplo conjunto de aplicações têm mais idéia de que características procurar e de como o computador normalmente trata várias situações. Especificamente, usuários com experiência em programação são aptos a usar as macro-linguagens e outros meios mais complexos de combinar comandos em aplicações simples, não orientadas a programação, como editores de texto, por exemplo.

Finalmente, a terceira dimensão que é **a experiência no domínio da tarefa** tem importância fundamental. Interfaces projetadas para especialistas podem fazer uso de terminologia e jargão específico de uma área de especialidade. Usuários com pouca experiência terão que ter mais explicação sobre o que o sistema faz e sobre o que as diferentes opções significam, e a terminologia usada não pode ser tão abreviada e densa quanto a dirigida para especialistas. Por exemplo, consideremos um sistema auxiliar a escolha de investimentos financeiros. Caso o sistema seja dirigido a não especialistas muito cuidado deve ser tomado dada a complexidade do domínio e a especificidade de sua terminologia. Poucos de nós sabe diferenciar, por exemplo, uma ação ordinária de uma preferencial, no mercado de aplicação em ações.

Diferenças entre usuários tem outras dimensões além da experiência. Alguns fatores são fáceis de serem verificados, como a idade (Czaja, 1988) e sexo (Fowler e Murray, 1987). Outros são menos óbvios como diferenças em habilidades de raciocínio (Gomez *et al*, 1986), estilos de aprendizagem (Sein e Bostron, 1989), etc. Diferenças culturais também são extremamente relevantes e serão tratadas quando discutirmos interfaces internacionais.

Parece complexo, e realmente o é, obter ótimos graus de usabilidade em todos os atributos simultaneamente. Compromissos são inerentes ao processo de design. Por exemplo, o desejo de evitar erros catastróficos pode levar a se ter uma interface menos eficiente de usar, no estilo das interfaces que a cada ação solicita ao usuário a confirmação e a re-confirmação antes da ação ser executada. Então, é importante estabelecer os objetivos de usabilidade a serem atingidos, quais os atributos a serem priorizados e isso é definido pelo contexto específico ao qual é dirigido um projeto.

Em **aplicações de escritório**, **domésticas e de entretenimento** - processadores de texto, jogos, softwares educacionais, etc. - facilidade de aprendizagem, baixa taxa de erros e satisfação subjetiva são fundamentais e devem ser maximizados. A escolha das funcionalidades adequadas nesse tipo de sistemas é muito difícil, pois toda gama de usuários deve ser alvo e certamente é mais que desejável uma evolução agradável do nível principiante para o especialista.

Sistemas críticos - controle de tráfego aéreo, reatores nucleares, operações militares, etc. - são sistemas de alto custo onde espera-se alta confiabilidade e efetividade. Também são sistemas altamente complexos onde um treinamento é aceitável de forma a obter rapidamente ausência de erro, mesmo sob estresse. Satisfação subjetiva é menos valorizada pois os usuários são bem motivados. E como são sistemas de uso freqüente a memorização é naturalmente obtida. Eficiência portanto é o atributo base do design de sistemas dessa categoria.

Sistemas de uso comercial e industrial - banco, seguros, reserva aérea, aluguel de carros, gerenciamento de cartão de crédito, etc. - que são de uso amplo, exigem um elevado custo de treinamento. Então facilidade de aprendizagem é fundamental de

forma a se ter performance rápida para muitos a custos razoáveis. Satisfação subjetiva tem importância modesta pois os usuários são motivados ao uso e a memorização é obtida a partir do uso freqüente.

Os sistemas exploratórios, **cooperativos e criativos** - enciclopédias eletrônicas, escrita cooperativa, tomada de decisão, simulação científica, sistemas para composição musical, diagnóstico médico, auxiliares de projetos de arquitetura, etc. - são sistemas geralmente direcionados a usuários peritos no domínio da tarefa, mas não experientes com computadores. São usuários altamente motivados e a preocupação central do design é a de deixar o computador transparente de forma a que o usuário somente se preocupe com a tarefa. Então, baixa taxa de erros e facilidade de aprendizagem devem ser os atributos mais relevantes.

Certamente, não podemos nos esquecer que existem outras considerações que não a usabilidade que levam a design que viola princípios de usabilidade. Por exemplo, considerações quanto a segurança freqüentemente requerem acesso a controles que não "passam" em nenhuma avaliação de usabilidade. Certamente, mensagens de erro construtivas são difíceis em resposta a senhas erradas. Outros sistemas, têm funções escondidas do usuário comum, como por exemplo, funções especias de *boot* ou de utilização apenas pela administração do sistema, por exemplo.

USABILIDADE NA WEB

Dados disponíveis apontam que em 1998 cerca de três bilhões de dólares deixaram de ser ganhos na WEB norte-americana por causa de design mal feito de páginas, que dificultava a compra em vez de facilitar. Essa estimativa dá conta de um debate que ganha cada vez mais espaço: como equilibrar o uso de recursos visuais capazes de atrair a atenção do usuário e ao mesmo tempo tornar os *site*s fáceis de entender e usar? A questão pode ser parafraseada: Como fazer uso da tecnologia disponível e ao mesmo tempo aumentar a usabilidade de *sites* da Web?

Com cerca de 10 milhões de *sites* na Web em Janeiro de 2000 (com a previsão de cerca de 25 milhões até o final do ano e 100 milhões em 2002), usuários têm mais escolhas que nunca. Por que então eles iriam gastar seu tempo em algo que é confuso, lento, e que não satisfaz suas necessidades?

Como dissemos no início deste capítulo, o número de pessoas que usa a Internet está crescendo sem parar. O crescimento trouxe mudanças no perfil do usuário. No começo predominavam os especialistas e agora predominam os novatos, que mal sabem ligar o computador e que algumas vezes tem rejeição a ele. Assim, deslumbrar-se com a tecnologia não tem mais razão de ser. Com a enorme oferta de alternativas, usuários da Web tem uma notável impaciência e insistência em

gratificação imediata. Se não conseguem entender como usar um *Website* em poucos minutos, eles concluem que não vale a pena perder seu tempo. E então o abandonam.

Usabilidade assumiu uma importância na economia da Internet como nunca teve antes (Nielsen, 1999). No desenvolvimento tradicional de produtos, usuários não experimentam a usabilidade do produto até que o tenham comprado. Por exemplo, somente quando se compra um VCR é que se descobre, o que é bastante comum, o quanto é difícil programá-lo. Mas isso não importa mais para o fabricante, em um primeiro momento, pois a compra já foi faturada. A indústria de software já tem um pouco mais de preocupação com a usabilidade de seus produtos, dado o suporte que é preciso ser dado ao usuário e que tem um custo altamente significativo no produto.

A Web reverteu esse cenário. Agora o usuário experimenta a usabilidade de um *site* antes de se comprometer a usá-lo e antes de ter gasto qualquer dinheiro com potenciais compras e a equação é bastante simples:

- no design de produtos e de softwares tradicionais, usuários pagam antes e experimentam a usabilidade depois
- na Web usuários experimentam a usabilidade antes e pagam depois.

Portanto, é clara a extrema importância da usabilidade no design para a Web.

Um exemplo muito citado é o da IBM dos Estados Unidos (Figura 1.12). A empresa constatou que o recurso mais popular em seu *site* era a função de busca, porque as pessoas não conseguiam descobrir como navegar, e o segundo mais popular era o botão de ajuda. A solução foi um amplo processo de redesign, envolvendo centenas de pessoas e milhões de dólares. Resultado: na primeira semana depois do redesign, em fevereiro de 1999, o uso do botão de ajuda caiu 84% enquanto as vendas aumentaram 400%.

FIGURA 1.12 – PÁGINAS DO *SITE* DA IBM AMERICANA

Outro exemplo, também bastante conhecido, é o *site* da Amazon Books (Figura 1.13). No início ele era exuberante, mas ultimamente ele tem mudado muito, perdendo os "fogos de artifício" em favor da funcionalidade e rapidez e, principalmente, do relacionamento com o usuário. Certamente o que impulsionou a Web não foi a multimídia, mas a capacidade de relacionamento que nenhuma outra mídia ofereceu até então, e hoje os melhores *site*s são os que estão aprendendo a se relacionar com o usuário.

FIGURA 1.13 – PÁGINA DE ENTRADA DO *SITE* DA AMAZON BOOKS EM FEVEREIRO DE 2000

Um paralelo esclarecedor pode ser feito com uma agência bancária - *imagine se você fosse a uma agência bancária para fazer um simples depósito e lhe obrigassem a sentar na recepção, ouvir uma propaganda sobre o elenco de serviços que aquele*

banco oferece, passar para outra sala, ouvir mais coisas, esperar, e só então ser transferido para a fila do caixa. É assim que muitos sites são construídos (Pedro Mozart, Gazeta Mercantil, 20 de outubro de 1999, p. C-8)

No design para a Web existem basicamente duas abordagens: uma artística onde o designer se expressa e outra dirigida a resolver o problema do usuário. Certamente existe a necessidade da arte, da diversão e do prazer na Web, mas acreditamos que o principal objetivo dos projetos para a Web deva ser o de tornar fácil para os usuários executarem tarefas úteis. Nada diferente do que advogamos em todo este livro.

Para garantir usabilidade em design para a Web podemos estabelecer alguns princípios básicos (Nielsen, 1999):

- **Clareza na arquitetura da informação**
  É essencial que o usuário consiga discernir o que é prioritário e o que é secundário no *site*. Ou seja, antes de mais nada é preciso chegar a um bom arranjo da informação. Os usuários sempre terão dificuldades em encontrar o que procuram, então devem ser ajudados provendo-se um senso de como a informação está estruturada e localizada. Para se conseguir isso, uma das alternativas adotadas em alguns *site*s, é prover um mapa do *site*, de forma que os usuários saibam onde estão e para onde podem ir.

- **Facilidade de navegação**
  Uma máxima é que o usuário deveria conseguir acessar a informação desejada no máximo em três cliques. E conseguir organizar a informação dentro disso já é um bom princípio.

- **Simplicidade**
  Quem navega quer encontrar o mais rapidamente possível o objetivo da busca. Portanto, a pirotecnia deve ser evitada, dando ao usuário paz e tranquilidade para que possa analisar a informação. Cuidados devem ser tomados para que a simplicidade não signifique ausência de informação. Por exemplo, ao se entrar em uma homepage do *site* de uma instituição ou projeto o usuário precisa que duas perguntas básicas sejam respondidas: Onde eu estou? O que posso obter nesse *site*?

FIGURA 1.14 – PÁGINA DE ENTRADA DO NÚCLEO DE INFORMÁTICA APLICADA Ã EDUCAÇÃO – NIED – DA UNICAMP – SIMPLES MAS SEM INFORMAÇÃO

Muitos exemplos podem ser encontrados na rede de páginas que de tão simples (Figura 1.14) omitem informações básicas como essas. Sempre temos que ter em mente que o que importa na Web é a informação e ela não pode ser omitida em função de uma pretensa simplicidade.

- **A relevância do conteúdo**
  Se nas revistas ou na televisão, por exemplo, a sedução passa muito pela beleza das imagens, na Web o conteúdo é o que mais importa para atrair e prender a atenção do usuário. Sempre que questionados sobre *site*s, usuários se referem a qualidade e relevância do conteúdo. Um bom texto para essa mídia tem que ser o mais conciso e objetivo possível, não promocional ou publicitário, como impera hoje, com perda de credibilidade. É preciso alterar o estilo de escrita, de forma a ser otimizado para leitores *online* que freqüentemente imprimem textos e que necessitam páginas bem curtas com a informação secundária deixada para páginas de suporte.

- **Manter a consistência**
  Assim, como para qualquer outro tipo de software, a consistência é um poderoso princípio de usabilidade na Web. Quando as coisas acontecem sempre do mesmo jeito, os usuários não precisam se preocupar a respeito do que irá acontecer. Ao contrário, eles sabem o que vai acontecer baseados numa experiência anterior. Isso leva a adoção de procedimentos padrões, como por exemplo, o uso de cores. *Layouts* ambiciosos devem ser abandonados. As fontes a serem usadas devem ser as mais comuns, pois o designer não sabe as fontes que o usuário tem instaladas.

  Outro aspecto bastante verificado e que transparece no design, é o de se gerenciar um projeto para a Web da mesma forma que qualquer outro projeto corporativo tradicional. Isso conduz a um design com uma interface inconsistente. Ao invés disso, um *Website* deve ser gerenciado como um projeto único de interface com o usuário.

- **Tempo suportável**
  O tempo de carga das páginas deve ser necessariamente curto. Estudos indicam que 10 segundos é o máximo de tempo antes que as pessoas percam o interesse. Mas na Web os usuários já têm uma baixa expectativa, então esse limite pode aumentar para 15 segundos e mesmo assim ser aceitável

- **Foco nos usuários**
  Novamente, todos os princípios podem ser sumarizados em um só: o foco deve estar nas atividades dos usuários. Deixar-se embevecer pelas últimas tecnologias

da Web irá atrair uns poucos interessados somente na tecnologia. Como cada vez há um número maior de páginas, as pessoas estão se tornando impacientes com *site*s não usáveis e não tem pudor algum em mudar - afinal, há atualmente outros dez milhões de *site*s para ir e nada impede a livre navegação.

Resumindo, pessoas são extremamente dirigidas a um objetivo quando usam a Web. Elas têm alguma coisa específica que querem fazer e não toleram nada que dificulte atingir esse objetivo. Portanto, o princípio mestre do design para a Web é "sair do caminho" de forma a que o usuário possa fazer o que quer da maneira mais rápida possível.

INTERFACES INTERNACIONAIS

Exportar software é vital para qualquer indústria produtora de software. Interfaces internacionais são aquelas projetadas para serem usadas em mais de um país. Fato mais que definitivo é que não basta um produto traduzido em muitas outras línguas. Fazer o design de uma interface internacional pode ou não envolver tradução de linguagem, mas certamente deve envolver conhecimento sobre as necessidades e cultura de outros países. Usuários desejam um produto que seja adequado às suas características culturais e práticas de trabalho e, algumas vezes, isso não implica necessariamente em uma tradução. Equipes de design de software têm então pela frente o desafio de garantir a usabilidade de seus produtos para todo o mercado global.

Do ponto de vista dos usuários, temos que bem mais da metade de usuários de software usam atualmente interfaces cujo design foi feito em um país estrangeiro. Usabilidade para esse grande número de usuários irá depender da maior consciência da necessidade de se ter design direcionado à internacionalização (Nielsen, 1993).

FIGURA 1.15 – PÁGINA DE ENTRADA DO *SITE* DO YAHOO AMERICANO EM FEVEREIRO DE 2000

FIGURA 1.16 – LOCALIZAÇÃO DO YAHOO PARA O BRASIL COM LINKS PARA FUTEBOL

Como já mencionamos anteriormente, o desenvolvimento tradicional de software distingue entre **Internacionalização** e **Localização**. Internacionalização faz referência a se ter um único design que possa ser usado em qualquer parte do mundo, e localização é o processo de adaptar uma versão do design para um local específico. Internacionalização envolve usar uma linguagem simples que possa ser entendida por pessoas não nativas, enquanto localização frequentemente envolve a tradução (Figuras 1.15, 1.16)

Mesmo não sendo um fenômeno novo, garantir a usabilidade internacional ganha absoluta relevância nos dias de hoje graças à fantástica expansão da WWW (literalmente rede MUNDIAL). No contexto da Web, inicialmente faz mais sentido o processo de internacionalização ao invés de localização, pois em muitos países o número de usuários não justifica uma localização.

À primeira vista, pode parecer que a tendência atual do uso de interfaces gráficas e o uso de elementos gráficos ao invés de palavras resolve grande parte dos problemas. Mas isso não é bem verdade, pois símbolos gráficos e padrões de cores não são necessariamente universais. Podemos classificar os símbolos gráficos em três categorias distintas (Rogers, 1989):

- **Símbolos de semelhança ou ícones**: retratam o objeto que representam. Por exemplo, a figura de uma impressora para indicar a função imprimir, a de uma tesoura para a cortar, etc.

- **Símbolos de referência**: retratam algum objeto que por referência ou analogia pode representar o conceito que o símbolo está querendo representar. Por exemplo, o uso de uma lupa indicando que se pode ver o conteúdo de um arquivo (ou um *preview* de um documento), o uso de um semáforo indicando as funções de parar, prossegir e esperar de um depurador, etc.

- **Símbolos arbitrários:** formas arbitrárias que somente tem significado por convenção. Por exemplo, todos os sinais de trânsito (que têm sido fonte de inspiração para muitos designers dada a sua característica de padrão internacional). Certamente, essa é a categoria de simbolos mais difícil de aprender para o usuário, a menos que sejam tão largamente uilizados que a convenção deixa de ser menos importante. Por exemplo, poucos usuários se dão conta que o uso do símbolo "?" para indicar dúvida é absolutamente arbitrário (Nielsen, 1993)

Símbolos por semelhança geralmente são comuns a muitos países, desde que não sejam de natureza muito específica. Por exemplo, símbolos de esportes onde em alguns países o esporte não é nem conhecido - o caso do *squash* na Hungria (Brugger, 1990). Portanto o uso de símbolos por semelhança, é preferível para a internacionalização (e para interfaces sem esse objetivo também).

Outro exemplo, de problema interessante em interfaces gráficas é o uso dos marcadores em *check boxes*. Em muitas interfaces o uso do sinal X teve que ser substituído pelo V (sinal de visto) pois em testes no Japão, o X era entendido como exclusão e não seleção da opção.

Conceitualmente os sinais gráficos presentes em interfaces têm sua origem e entendimento na Semiótica, que será tratada no Capítulo 3.

Além desses problemas gráficos, tem-se o problema da absoluta falta de tradição no processo de tradução da linguagem. Cada software e cada designer tem seus próprios termos. O pior é que nem em produtos de um mesmo fabricante a consistência é garantida.

Portanto, interfaces gráficas não possuem uma garantia inerente de serem internacionais. O design precisa efetivamente ser orientado à internacionalização desde o início do processo. E os princípios de IHC tratados ao longo deste livro são os mesmos, embora tratados em algumas partes com um maior grau de profundidade. Em primeiro lugar os usuários e suas tarefas, localizados em um contexto.

REFERÊNCIAS:

Brigger, C. (1990) Advances in the international standardization of public information symbols. *Information Design Journal* 6, 1, 79-88

Carroll, J. M. e Thomas, J. C. (1988) Fun. ACM SIGCHI Bulletin 19,3 (Januay), 21-24

Czaja, S. J., Hammond, K., Blascovitch, J. J., e Swede, H. (1989) Age related differences in learning to use text-editing system. *Behaviour & Information Technology* 8, 4, 309-319.

Erickson, T.D. (1990) Working with Interface Metaphors. Em B. Laurel (ed.) (1990) *The Art of Human-Computer Interface Design.* Reading, Mass.: Addison-Wesley

Fernandes, T.(1995) *Global Interfaces Design: A Guide to Design International User Interfaces.* Academic Press Professional, Boston, MA

Fowler, C. J. H. e Murray, D. (1987) Gender and cognitive style differences at the human-computer interface. *Proc. IFIP INTERACT´87 Second Intl. Conf. Human-Computer Interaction* (Stuttgart, Germany, 1-4 September), 709-714

Gomez, L. M., Egan, D. E., e Bowers, C. (1986) Learning to use a text-editor : Some learner characteristics that predict success. *Human-Computer Interaction*, 2, 1, 1-23

Jorgensen, A. H. (1990) The personal touch: A study of users customization practice. *Proc. IFIP INTERACT'90 Third Intl. Conf. Human Computer Interaction* (Cambridge, U.K., 27-31 August), 549-554

Kay, R.H. (1989) A practical and theoretical approach to assessing computer attitudes: The computer attitude measure (CAM). *Journal on Research on Computing in Education*, 456-463

LaLomia, M. J. e Sidowski, J. B. (1991) Measurements of computer attitudes: A review. *Intl. J. Human-Computer Interaction*, 3,2,171-197

Laurel, B. ed. (1990) *The Art of Human-Computer Interface Design.* Reading, Mass.: Addison-Wesley

Laurel, B. (1993) *Computer as a Theatre.* Reading, Mass.: Addison-Wesley

Lyons, J. (1970) *New Horizons in Linguistics.* Hamondsworth:Penguin

Nielsen, J.(1990) *Design User Interfaces for International Use.* Elsevier Science Pub., Amsterdam, The Hetherlands

Nielsen, J. (1993) *Usability Engineering.* Academic Press, Cambridge, MA

Nielsen, J. (1999) *Design Web Usability.* New Riders Publish., Indianapolis, Indiana USA

Norman, D. A. (1988) *The Psychology of Everyday Things.* Basic Books, New York

Perratore, E., Thompson, T., Udell, J., e Malloy, R. (1993) Fighting fatware. *Byte* 18, 4(April), 98-108

Preece, J. (1993) A Guide to Usability: Human Factors in Computing. Reading, Mass.: Addison-Wesley

Preece, J. *et al*. (1994), *Human-Computer Interaction.* Reading, Mass.: Addison-Wesley

Proceedings ACM CHI´85 Conference: Human Factors in Computing Systems. Lorraine Borman and Bill Curtis (eds.), San Francisco (April 14-18, 1985)

Proceedings ACM CHI´92 Conference: Human Factors in Computing Systems. Penny Bauersfeld, John Bennett and Gene Lynch(ed.), Monterey, CA(May 03-07,1992)

Rogers, Y. (1989) Icons at the Interface: Their usefulness. *Interacting with Computers*, 1,1(April), 105-117

Russo, P. e Boor, S. (1993) How fluent is your interface? Design for international users, in Bridges Between Worlds. *INTERCHI´93 Conferencing Proceedings*, Reading, Mass.: Addison-Wesley

Sein, M. K. e Bostron, R. P. (1989) Individual differences and consceptual models in training novice users. *Human-Computer Interaction*, 4, 3, 197-229

Sellen, A. e Nicol, A. (1990) Building user-centered On-line Help. Em B. Laurel (ed.) *The Art of Human-Computer Interface Design.* Reading, Mass.: Addison-Wesley

Schneiderman, B. (1980) *Software Psychology: Human factors in Computer and Information Systems.* Little, Brown, Boston (1980)

Schneiderman, B. (1998) *Design the User Interface - 3rd Edition.* Reading, Mass.: Addison-Wesley

Tesler, L.G. (1991) Networked Computing in the 1990s. *Scientific American* 265, 3(September), 86-93

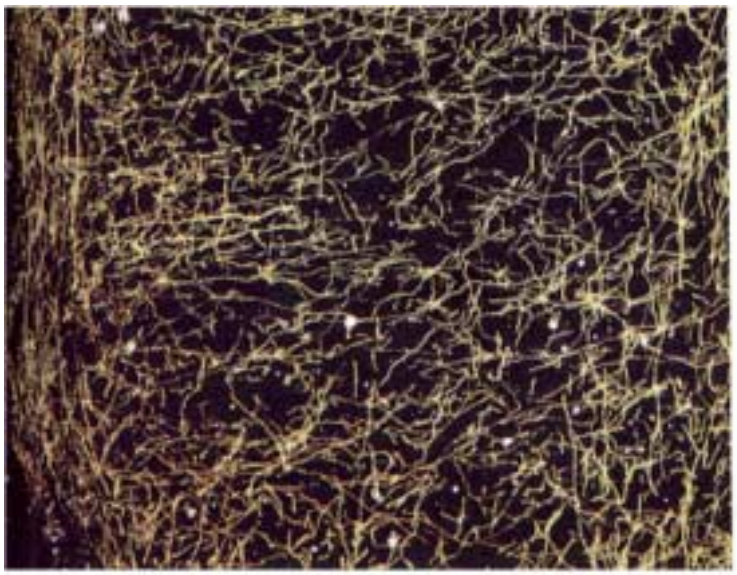
Vaske, J. J. e Grantan, C.E. (1990) *Socializing the Human-Computer Environment.* Norwood, NJ:Ablex

Walkers, J. (1990) Through the Looking Glass. Em B. Laurel (ed.) *The Art of Human-Computer Interface Design.* Reading, Mass.: Addison-Wesley

Wozny, L. A. (1989) The application of metaphor, analogy, ans conceptual models in computer systems. *Interacting with Computers* 1, 3, 273-283

CAPÍTULO 2

FUNDAMENTOS DE FATORES HUMANOS EM IHC



Cortex cerebral, *Scientific American*, July/August 1995 p.50

INTRODUÇÃO

Como apresentado no capítulo anterior, o Estudo da Interação Humano-Computador envolve conhecimento sobre o Humano por um lado, sobre a tecnologia por outro e sobre as maneiras como um influencia e é influenciado pelo outro. Neste capítulo estudaremos as capacidades físicas e cognitivas do Humano, como fatores que influenciam o design de interfaces de sistemas computacionais. Esse entendimento está fortemente motivado pela busca de melhoria da qualidade da interação entre pessoas e computadores.

O comportamento humano e os processos mentais subjacentes têm sido estudados pela Psicologia Cognitiva que adotou o modelo de processamento de informação para estudar esse comportamento. A referência clássica para esse modelo são os trabalhos de Card, Moran e Newell (1993). O modelo do Processador de Informação Humano será utilizado como uma aproximação inicial para se entender e analisar o uso de interfaces em relação ao processamento motor, viso-motor, perceptual e cognitivo do sistema humano. Será apresentado e discutido o modelo GOMS (*Goals, Operations, Methods and Selection Rules*), desde sua formulação básica até versões estendidas mais atuais apresentadas em literatura recente. O modelo GOMS é uma abstração para uma família de modelos que tentam caracterizar os vários processos cognitivos subjacentes à realização de determinada tarefa. O modelo possibilita várias predições qualitativas e quantitativas a respeito da performance humana em interação com computadores.

Considerando que esse modelo, como qualquer outro, é apenas uma aproximação e não reflete precisamente a complexidade e sofisticação da mente humana, alguns mecanismos fisiológicos importantes como os mecanismos da percepção e circuitos neurais da memória serão apresentados em maiores detalhes, com base em Lindsay e Norman (1972), outra referência clássica na literatura em questão.

Embora muitos desses fatores humanos sejam relacionados a questões ergonômicas do ambiente de trabalho como, por exemplo, os efeitos de design de determinado periférico sobre a sua eficiência de uso, estendemos nossa apresentação a questões relacionadas a ergonomia cognitiva principalmente, isto é, à adequação de interfaces aos mecanismos e modelos mentais humanos.

O tópico sobre modelos mentais está relacionado diretamente ao entendimento sobre o uso de metáforas em interfaces e têm implicações diretas em questões de usabilidade dessas interfaces. A literatura nesse assunto é mais recente e estaremos utilizando além de referências básicas como Johnson-Laird (1989), Carrol e Olson (1988 a,b), Lakoff e Johnson (1980), publicações científicas na área.

A PSICOLOGIA DA INTERAÇÃO HUMANO-COMPUTADOR

O conceito de interface tem evoluído na mesma proporção em que se conhece mais sobre a tecnologia dos computadores por um lado e sobre a natureza humana, por outro. Na verdade há uma relação dialética entre o nosso conhecimento sobre o Homem e os artefatos que ele cria, em especial os tecnológicos. O design de ambientes baseados no computador, portanto, reflete e ao mesmo tempo é influenciado pelo conhecimento científico sobre a natureza humana. Nossa própria relação com o computador tem sido nomeada diferentemente ao longo da história ainda recente desse artefato de nossa cultura: da relação de operação da máquina computador até conceituações mais recentes de comunicação com e/ou através de computadores. O próprio uso da palavra "interação" já tem sido contestado (Suchman, 1999), como será discutido no Capítulo 5.

O modelo de processador de informação humano dominante nos anos setenta, refletia um paralelismo com a arquitetura de computadores da época, e consistia de uma estrutura lógica composta de muitos registradores (boxes) cada um com seus parâmetros de memória conectados por um conjunto de caminhos de transferência. O Modelo do Processador de Informação Humano, que será apresentado na próxima seção, substituiu registradores separados, por um aninhamento que usa sub-registradores. A noção de memória como tendo força e podendo fortalecer-se pela repetição passa à noção de memória como um conjunto de *chunks* discretos na memória de longa duração, que são ativados com base em estratégias de acesso.

O conhecimento sobre o ser humano enquanto sistema tem alimentado teorias em várias áreas do conhecimento; ao mesmo tempo usamos da analogia para refletir e construir conhecimento sobre o Homem. Só para citar algumas áreas de ciência e tecnologia em que essa relação dialética se estabelece, a Inteligência Artificial, as Redes Neurais, a Cibernética, a Teoria da Informação, a Engenharia Genética, são exemplos contundentes.

Independentemente da discussão sobre que disciplinas influenciam ou são influenciadas por quais outras, ao falar de interação humano-computador estamos sobre uma região de fronteira que intercepta no mínimo a Ciência da Computação e a Psicologia. As características intrínsecas desse artefato tecnológico particular nos impelem a conhecer mais sobre como e porque "interagimos", nos "comunicamos", ou "imergimos" em ambientes baseados no computador. Card, Moran e Newell (1983) foram os primeiros autores a desenvolver um Modelo do Usuário de Computadores, com base no estudo do seu funcionamento psicológico, para entender como características intrínsecas ao ser humano afetam a maneira como ele interage com computadores. O Modelo do Processador de Informação Humano, que será apresentado na próxima seção, forma as bases para as abordagens cognitivas ao design de sistemas computacionais, que serão tratadas no Capítulo 3, e à avaliação de tais sistemas, tratada no Capítulo 4.

UMA TEORIA CLÁSSICA PARA O PROCESSAMENTO DE INFORMAÇÃO NO HOMEM

A facilidade com que palavras da linguagem de interface podem ser lembradas, como o tipo de fontes de caracteres afetam a legibilidade, e a velocidade com que lemos informação na tela, são exemplos simples de como nossa interação com computadores pode ser afetada pelo funcionamento de nossos mecanismos perceptuais, motores e de memória.

Assim como o engenheiro de computação descreve um sistema de processamento de informações em termos de memórias, processadores, seus parâmetros e interconexões, Card *et al* (1983) propõem o Modelo do Processador de Informação Humano (MPIH), como uma descrição aproximada para ajudar a prever a interação usuário-computador, com relação a comportamentos. O modelo é constituído por um conjunto de memórias e processadores e um conjunto de princípios de operação. Três subsistemas fazem parte e interagem no MPIH: o Sistema Perceptual (SP), o Sistema Motor (SM) e o Sistema Cognitivo (SC). A Figura 2.1, adaptada de Card *et al* (1983), ilustra o modelo proposto.

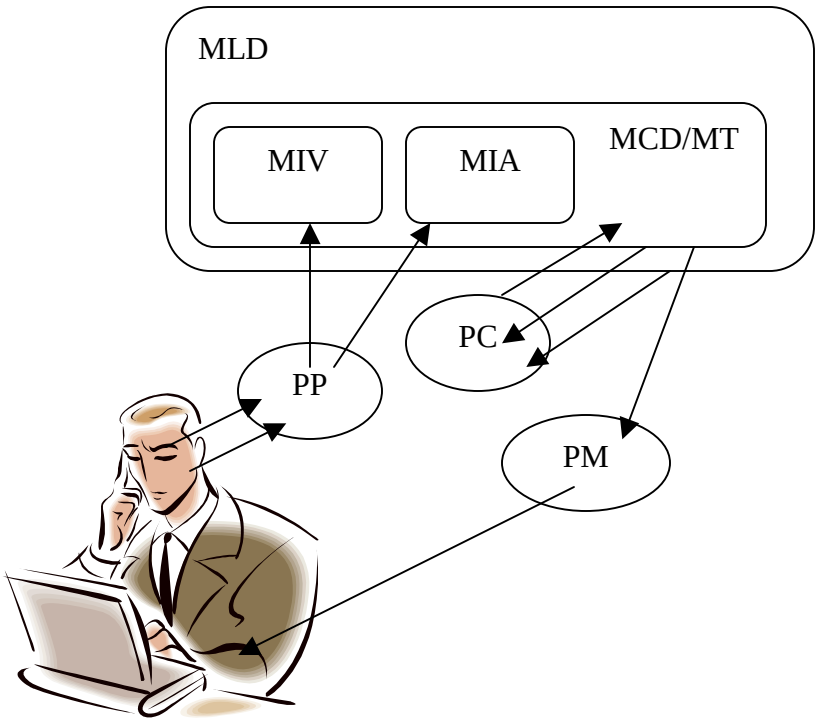


FIGURA 2.1 O MPIH E SEUS COMPONENTES PRINCIPAIS

Conforme pode ser observado na Figura 2.1, informação sensorial captada pelos órgãos dos sentidos – no caso específico pela visão e audição, flui para a Memória de Trabalho (MT), também chamada Memória de Curta Duração (MCD), através do Processador Perceptual (PP). A Memória de Trabalho consiste da ativação de partes da Memória de Longa Duração (MLD), que os autores chamam de *chunks*. O princípio básico de operação do MPIH é o ciclo Reconhece-Age do Processador Cognitivo (PC). O Processador Motor (PM) é acionado pela ativação de certos *chunks* da Memória de Trabalho, colocando em ação conjuntos de músculos que concretizam fisicamente determinada ação.

O Sistema Perceptual (SP) possui sensores e *buffers* associados, chamados Memória da Imagem Visual (MIV) e Memória da Imagem Auditiva (MIA), que guardam a saída do sistema sensorial enquanto ela está sendo codificada simbolicamente. O Sistema Cognitivo recebe informação codificada simbolicamente na MCD e usa informação armazenada previamente na MLD para tomar decisões de como responder. O Sistema Motor viabiliza a resposta.

Memórias e Processadores do modelo são descritos por parâmetros. Os parâmetros principais da memória são sua capacidade de armazenamento em itens (u), o tempo de desbotamento de um item (d) e o tipo do código utilizado na gravação - físico, acústico, visual, semântico (k). O parâmetro principal do processador é o tempo de ciclo (t). A seguir apresentaremos de forma resumida as principais propriedades de cada um dos subsistemas do MIPH.

O SISTEMA PERCEPTUAL

O Sistema Perceptual transporta sensações do mundo físico, detectadas por sistemas sensoriais do corpo e os transforma em representações internas. O sistema visual humano é um exemplo fantástico de vários subsistemas – visão central, visão periférica, movimentação do olho, movimentação da cabeça – operando de forma integrada para prover uma representação contínua da cena visual de interesse do observador. A retina é sensível à luz e registra sua intensidade, comprimento de onda e distribuição espacial. Embora o olho tome a cena visual em quase meio hemisfério, detalhe da cena é obtido somente em uma região estreita, de dois graus, chamada fóvea. O restante da retina provê visão periférica, necessária para orientação, conforme veremos mais adiante. O olho fica em contínuo movimento em uma seqüência de "sacadas" (viagem + fixação). Sempre que o alvo está a mais de 30^o da fóvea, é necessário, também, movimento da cabeça para reduzir a distância angular. Card *et al*. (1983), baseados em dados experimentais, colocam como medida típica para a duração total do movimento do olho (tempo de viagem + tempo de fixação), 230 ms, considerando um intervalo para tempo de fixação que varia nas pessoas de 70 a 700 ms. Essa variação é devida à complexidade da tarefa e à

habilidade do observador. Para leitura, por exemplo, o movimento do olho em uma criança em seu primeiro ano de leitura é de 660 ms.
Muitos fenômenos perceptuais acontecem em uma área tão grande que a fóvea do olho deve ser movida para vê-los. Quando movimentos do olho estão envolvidos, eles dominam o tempo requerido para a tarefa. A rapidez com que uma pessoa pode ler um texto, por exemplo, depende de quanto ela "capta" em cada fixação e isso é função da habilidade do leitor e da dificuldade do material. Considerando o tempo de sacada de 230ms, se nesse tempo ele capta uma letra, a sua média de leitura seria de 52 palavras por minuto (considerando uma média de 5 letras por palavra). Se em uma sacada ele capta uma palavra, sua média de leitura será de 261 palavras/min. Se em uma sacada o leitor consegue captar uma frase (média de 2.5 palavras), seu tempo de leitura será de 652 palavras/min. Isso significa que com tempos de leitura muito superiores a esse, o leitor estaria "pulando" partes do texto em sua leitura. Se o material sendo lido é difícil, então o tempo do Processador Cognitivo pode ser o limitante do tempo de processamento.

Logo após a apresentação de um estímulo visual, uma representação do estímulo aparece na MIV; se o estímulo é auditivo, na MIA. Essas memórias sensoriais guardam informação codificada fisicamente: um análogo não simbólico ao estímulo externo. Esse código é afetado pelas propriedades físicas do estímulo, como, por exemplo, intensidade. Card *et al*. (1983, p.28) exemplificam que *a representação do número 2 contém características de curvatura e comprimento (ou padrões de freqüência espacial equivalentes) em oposição ao dígito reconhecido.*

Logo após a apresentação física de um estímulo nas memórias perceptuais, uma representação de pelo menos parte do conteúdo da memória perceptual ocorre na Memória de Trabalho. O **tempo de desbotamento (*d*)** das memórias perceptuais é definido como o tempo depois do qual a probabilidade de recuperação da informação é menor do que 50%. O MPIH estabelece um tempo de 200ms como parâmetro para o *dmiv*; resultados de dados experimentais variam em um intervalo entre 90 e 1000 ms. Como parâmetro do *dmia* é estabelecido um tempo de 1500ms, tomado de um intervalo que varia de 900 a 3500 ms. As **capacidades das memórias perceptuais (*u*)**, ainda que difíceis de serem fixadas, para efeitos do uso no modelo são definidas como *umiv*=17 letras (tomadas de um intervalo entre 7 e 17 letras) e *umia*=5 letras (tomadas de um intervalo de 4.4 e 6.2 letras), parâmetros esses obtidos de resultados experimentais.

Conforme dito anteriormente o parâmetro principal do processador perceptual é seu **tempo de ciclo (*tp)***, sua unidade de resposta a impulso. Se um estímulo é fornecido à retina no tempo *t=0*, no final do tempo *t=tp* a imagem estará disponível na memória de imagem visual MIV e o Homem diz que a vê. Por exemplo, o tempo de resposta do sistema visual para um breve pulso de luz, obtido de dados empíricos é dado por *tp=100* ms tomado de um intervalo que varia de 50 a 200 ms. Uma propriedade importante do processador perceptual é que seu tempo de ciclo não é constante; o *tp* é mais curto para estímulos mais intensos (de luz ou som, por

exemplo). Daí se origina o primeiro princípio do Modelo do Processador de Informação Humano:

> ***Princípio n. 1:*** *O tempo do ciclo do Processador Perceptual varia inversamente com a intensidade do estímulo.*

Eventos perceptuais que ocorrem dentro de um único ciclo, são combinados em um único *perceptum*, impressão mental percebida pelos sentidos, se forem suficientemente similares. Por exemplo, duas luzes ocorrendo em posições diferentes dentro do intervalo de tempo de 60 a 100 ms, nos dão a impressão de uma única luz em movimento. Um outro exemplo importante na literatura a respeito da percepção humana dá conta de que um breve pulso de luz que dura *t* ms com intensidade *i* tem a mesma aparência de um longo pulso de menor intensidade, se ambos os pulsos duram menos de 100 ms. Essa propriedade é conhecida como Lei de *Bloch*: $i.t = k$, $t<tp$.

Card *et al.* (1983), mostram os resultados de um experimento clássico[1] no qual uma explosão de sons contendo um número desconhecido de cliques (estalidos), em intervalos uniformes de 10/seg, 15/seg e 30/seg foram apresentados a sujeitos do experimento. O gráfico da Figura 2.2 mostra o número de cliques reportados pelos sujeitos como uma função do número apresentado.

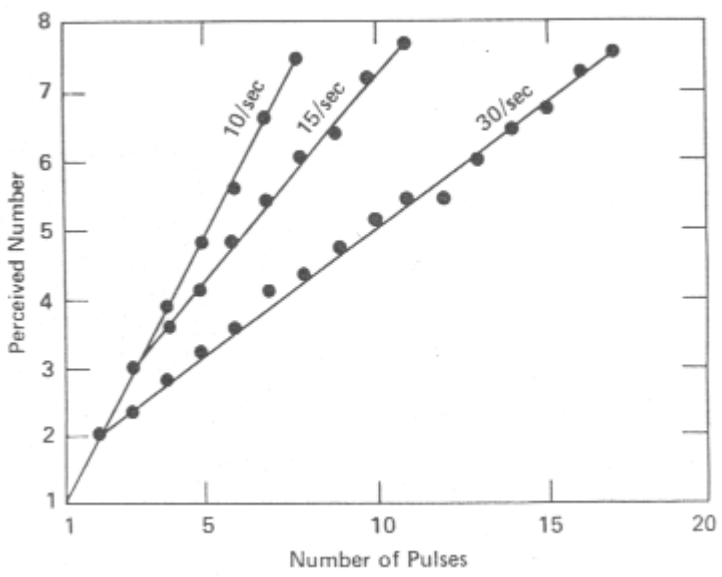


FIGURA 2.2 – EXPERIMENTO DA FUSÃO DE CLIQUES EM 100MS (CARD *ET AL*,1983, P.33)

[1] Cheatham e Whote (1954)

Os resultados mostram que os sujeitos ouvem o número correto quando os cliques são apresentados na seqüência de 10 cliques por segundo, mas perdem progressivamente mais cliques no caso das seqüências de 15 e 30 cliques por segundo. No primeiro caso, como há um clique para cada *tp* (100 ms) o sujeito ouve (percebe) cada som. No último caso, quando há 3 cliques para cada *tp*, os 3 cliques em cada 100 ms são fundidos em 1 *perceptum* (talvez parecendo um som mais alto em volume), e o sujeito percebe (ouve) somente 1 clique em vez de 3.

O mesmo efeito pode ser observado em relação a estímulos visuais. Imagens parecidas colocadas mais próximas no tempo do que *tp* (tempo de ciclo do Processador Perceptual), são "fundidas" em uma única imagem. Assim, por exemplo, para produzir animação, a média de quadros de imagens apresentadas deve ser maior que 1 quadro a cada 100 ms, para possibilitar a percepção de movimento contínuo. As câmeras de filmagem usam em geral 20 quadros/seg o que corresponde à medida de *tp* do extremo do intervalo (*tp* = 50 ms).

O SISTEMA MOTOR

Conforme descrição do ciclo de operações no MPIH, após processamento perceptual e cognitivo, pensamento é finalmente traduzido em ação pela ativação de padrões de músculos voluntários que são arranjados em pares antagônicos disparados um após o outro em seqüência. Para usuários de computador, os sistemas braço-mão-dedo e cabeça-olho são exemplos de conjuntos desses músculos capazes de responder a impulso nervoso.

O movimento não é contínuo como parece, mas uma série de micro-movimentos discretos, cada um requerendo um ciclo do processador motor definido no MPIH como *tm* = 70 ms, tomado de dados experimentais num intervalo que varia de 30 a 100 ms. Um experimento interessante citado em Card *et al.* (1983) mostra resultados experimentais de atividade motora observada e medida. Este é um experimento bem simples, fácil de ser reproduzido: um sujeito deve usar a caneta para fazer movimentos entre duas linhas paralelas, indo e voltando o mais rapidamente possível, durante 5 segundos. A Figura 2.3 mostra as marcas feitas por um sujeito nesse experimento.

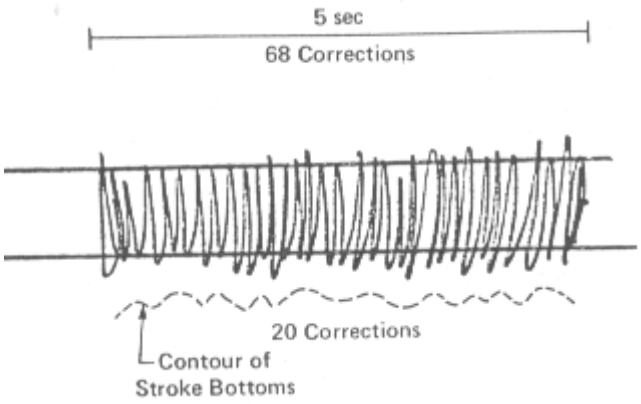


FIGURA 2.3 - MÉDIA DE SAÍDA DO PROCESSADOR MOTOR (CARD *ET AL* 1983, P. 35)

Como resultado extraído da Figura 2.3, 68 inversões da caneta foram feitas em 5 segundos o que significa um tempo de 74 ms por inversão. Em 5 segundos o sujeito faz 20 movimentos, resultando em 250 ms por movimento, o que se aproxima bastante do resultado teórico do modelo.

Observando a performance de usuários ao teclado, existem dados experimentais que registram uma média de 1000 ms para o novato e 60 ms para o experto. Usando o MPIH, quão rápido um usuário poderia pressionar repetitivamente com o mesmo dedo, uma determinada tecla? Estariam envolvidos na tarefa, 2 tempos do processador motor, um para pressionar e outro para soltar a tecla, o que daria 140 ms por toque. É claro que, numa tarefa real usando o teclado outros custos, incluindo os perceptuais e cognitivos (texto difícil, falta de experiência), estariam também envolvidos e diminuiriam a média.

O MPIH pode ser usado para se medir a velocidade relativa de usuários expertos em dois tipos diferentes de teclado. Experimento reportado em Card *et al.* (1983, p. 63) mostra que a datilografia no teclado alfabético é 8% mais lenta do que no teclado *qwerty* (o teclado convencional de Sholes), para texto escrito na língua inglesa e considerando freqüências com que combinações de 2 letras aparecem no inglês.

O SISTEMA COGNITIVO

Nas tarefas mais simples, o Sistema Cognitivo (SC) serve meramente para conectar entradas do Sistema Perceptual para saídas corretas do Sistema Motor. Entretanto, a maioria das tarefas realizadas pelo humano envolve de forma complexa aprendizado, recuperação de fatos e resolução de problemas. Existem duas memórias associadas ao SC no MPIH, que formam as bases para o entendimento de estratégias e teorias em IHC, conforme veremos no Capítulo 3; são elas a Memória de Trabalho, também

chamada Memória de Curta-Duração (MCD), e a Memória de Longa Duração (MLD). Grosseiramente a MCD é usada para armazenar informação sob consideração no momento de determinada atividade e a MLD é usada para armazenar informação a ser acessada em longo prazo.

A Memória de Curta Duração, ou Memória de Trabalho armazena os produtos intermediários do pensamento e as representações produzidas pelo Sistema Perceptual. Estruturalmente consiste de um subconjunto de elementos da Memória de Longa Duração que se tornaram ativados. Funcionalmente é onde as operações mentais obtém seus operandos, e deixam seus resultados intermediários. O tipo predominante de código é o simbólico, diferentemente das MIV e MIA.

Conceitualmente a MCD é constituída de *chunks*: elementos ativados da MLD, que podem ser organizados em unidades maiores. O *chunk* é função tanto do usuário quanto da tarefa que ele tem para realizar, uma vez tratar-se de ativação de sua MLD. Por exemplo, a seqüência das letras a seguir H-I-C-S-A-U-I-W-M-P lidas sem qualquer diferença de entonação e de intervalo pode ser difícil para um ouvinte lembrar. Já a seqüência I-H-C-U-S-A-W-I-M-P, composta das mesmas letras em outra ordem poderão ser facilmente reproduzidas pelo ouvinte. Por que? Para uma certa população de ouvintes, a segunda seqüência representa apenas 3 *chunks* a serem lembrados (IHC, USA, WIMP) em vez de 10...

Chunks podem estar relacionados a outros *chunks*. Quando um *chunk* na MLD é ativado, a ativação se espalha aos *chunks* relacionados em vários níveis, conceitualmente como numa rede semântica. Há interferência de novos *chunks* com os antigos. Os *chunks* como os elementos da MIV e MIA, estão sujeitos ao desbotamento com o tempo. No modelo existe um parâmetro para o desbotamento do *chunk*, definido por *dmcd* = 7seg, tomado de um intervalo de 5 a 226 segundos com base em resultados experimentais. O tamanho do intervalo de variação é explicado em Card *et al.* (1983) pela dificuldade em analisar-se fenômenos de interferência entre *chunks* na MCD.

Um experimento bastante simples para verificarmos a capacidade da MCD é pedir que um sujeito recupere os dígitos que precedem uma seqüência que pára repentinamente. A capacidade observada da MCD é de *umcd* = 3 *chunks*, tomados em intervalo que varia de 2.5 a 4.1 *chunks*. Entretanto, dificilmente usamos a MCD isoladamente da MLD; quando se pede aos sujeitos para recuperarem a informação alguns segundos depois de ouvi-la, normalmente eles usam ambas MCD e MLD e, então, a capacidade efetiva da MCD estende-se para 7 *chunks*, de um intervalo de 5 a 9 *chunks*. Esse parâmetro é bastante utilizado em *guidelines* de design e avaliação de interfaces e será referenciado em outras partes deste livro.

Um experimento interessante[2] reportado em Card *et al.* (1983) mostra características de recuperação de informação na memória humana. Uma lista de palavras foi apresentada a um grupo de pessoas e foi pedido que elas as recuperassem em qualquer ordem. Os sujeitos foram impedidos de qualquer atividade de repetição (recitação mental ou física das palavras). Os resultados mostram que a probabilidade de relembrar uma palavra de uma lista de palavras é função da posição delas na lista e do tempo decorrido antes de iniciar a recuperação, conforme ilustra a Figura 2.4.

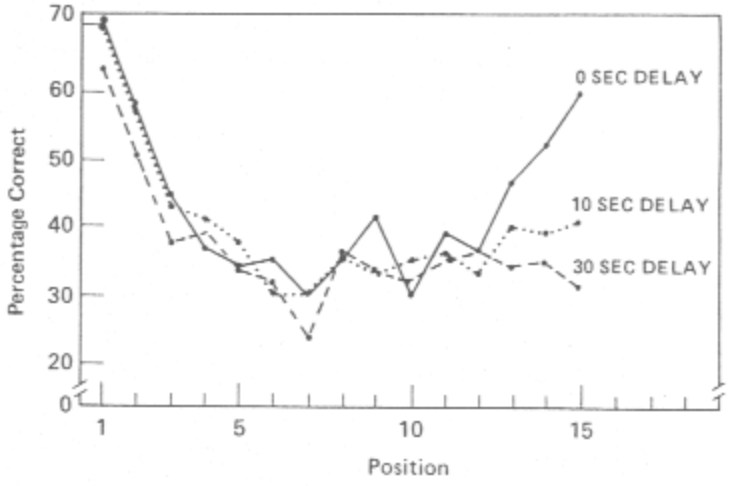


FIGURA 2.4 - LEMBRANDO PALAVRAS DE UMA LISTA (CARD *ET AL*, 1983, P.77)

Considerando a recuperação imediatamente após a apresentação da lista, a curva mostra que as palavras do início e do final da lista foram recuperadas mais facilmente do que as do meio. Quanto mais tempo decorre após a apresentação da lista, a recuperação das palavras do final da lista vai sendo diminuída. Isso se explica pela limitação da capacidade da MCD e pela conseqüente interferência de novos elementos na MCD.

A Memória de Longa Duração armazena a massa de conhecimento do usuário: fatos, procedimentos, história, etc. Conceitualmente pode ser entendida no modelo como uma rede de *chunks* acessados de forma associativa a partir da MCD ou Memória de Trabalho. O parâmetro de desbotamento tem valor infinito; teoricamente não há o "apagar" da MLD. Entretanto, a recuperação de um *chunk* da MLD pode falhar

[2] Ganzer e Cunitz (1966)

quando, por exemplo, associações não puderem ser encontradas, ou quando houver interferências entre associações de *chunks*.
O tipo de código predominante na MLD é o semântico. Quando a informação da MCD torna-se parte da MLD, a maneira como ela é codificada determina quais pistas serão efetivas na recuperação daquela informação mais tarde. Card *et al.* (1983) mostram um bom exemplo desse fenômeno: suponha que um usuário nomeie um arquivo de imagem de "*light*" significando o oposto a "*dark*"; se mais tarde ele percorre o diretório e pensa no "*light*" como oposto a "*heavy*", ele não reconhece o arquivo que está buscando, porque está usando um conjunto diferente de pistas para recuperação. Daí a formulação dos segundo e terceiro princípios associados ao MPIH:

> ***Princípio n. 2***: *Princípio da especificidade da codificação.*
> *Operações de codificação específicas realizadas sobre o que é percebido determinam o que é armazenado, e o que é armazenado determina que pistas de recuperação são efetivas em prover acesso ao que é armazenado*.
>
> ***Princípio n. 3:*** *Princípio da Discriminação*
> *A dificuldade da recuperação da memória é determinada pelos candidatos que existem na memória relativos às pistas para recuperação.*

O princípio n. 3 sugere que, embora presente fisicamente na memória, informação pode ser "perdida" funcionalmente. Algumas propriedades associadas à MLD, relevantes ao entendimento do MPIH podem ser observadas: quanto mais associações um item tiver, maior será a probabilidade de ser recuperado; itens de informação não são armazenados na MLD diretamente. A probabilidade de um item ser armazenado na MLD e associado de modo a ser recuperado aumenta com o seu tempo de residência na MCD.

Embora a recuperação de um item da MLD nem sempre seja bem sucedida, quando tempos grandes estão disponíveis para a busca, estratégias podem ser usadas para investigar a MLD. Experimento[3] reportado em Card *et al.* (1983) envolvia sujeitos que tinham que lembrar dos nomes de colegas de classe, depois de 7 anos de conclusão do 2^{o} grau. Os resultados mostraram que mesmo depois de 10 horas alguns sujeitos ainda estavam recuperando nomes. As estratégias usadas foram as mais variadas: desde a lembrança da posição espacial dos colegas na classe, lembrança de sub-grupos, até nomes em ordem alfabética, faces, etc. Muitos nomes foram também "fabricados", não fazendo parte da lista real, o que sugere o nível de interferência na MLD.

O Processador Cognitivo tem como unidade de medida o tempo de ciclo reconhece-age, definido pelo parâmetro *tc*=70 ms, tomado de um intervalo de 25 a 170 ms

[3] Williams e Hollan (1981)

obtidos experimentalmente. *Em cada ciclo, o conteúdo da MCD inicia ações de associação na MLD ("reconhece"), que por sua vez modifica o conteúdo da MCD ("age"), preparando para o próximo ciclo. Planos, procedimentos e outras formas de comportamento organizado são construídos a partir de um conjunto organizado de ciclos reconhece-age* (Card et al, 1983, p.41). Como exemplo de medição associada ao tempo de ciclo do processador cognitivo, podemos citar a contagem silenciosa. Experimente fazer uma contagem silenciosa 1, 2, 3, 4,... durante 5 segundos e verifique o tempo de seu processador cognitivo...Em dados experimentais[4] foi observada uma média de 167 ms por dígito (Card *et al*, 1983, p.43).

Como acontece com o Processador Perceptual, o *tc* não é constante, e será mais curto quanto maior for o esforço induzido, diminuindo com a prática. Essa propriedade dá origem ao quarto princípio do MPIH:

Princípio n. 4: *Princípio da variabilidade do ciclo do Processador Cognitivo*

O SC no MPIH é paralelo na fase de reconhecimento (pode-se estar consciente de muitas coisas ao mesmo tempo) e serial na fase de ação (não se consegue fazer deliberadamente mais do que uma coisa por vez). Isso explica a serialidade e o paralelismo que usamos em atividades do tipo dirigindo e conversando e lendo placas de trânsito, etc. "ao mesmo tempo". A serialidade ocorre no topo das atividades paralelas dos sistemas Perceptual e Motor.

Um exercício interessante para compreendermos a ação de nossos mecanismos perceptuais, motores e cognitivos: imagine que você está dirigindo em direção a determinada localidade e alguém pede para você ir explicando cada ação sua durante essa tarefa. A rota é conhecida e o tráfego está calmo. A partir de certo ponto, aparece uma interrupção na rota e você tem que desviar do caminho usual, buscando um caminho desconhecido. O tráfego agora está confuso e nervoso...como fica a sua tarefa? Enquanto você conhece a rota não precisa colocar muito esforço cognitivo no que está fazendo e, então, "falar sobre" é fácil. Quando a situação muda, você passa a ter que se concentrar mais para descobrir para onde ir – mais processamento cognitivo é necessário – e você pára de falar. Não é muito diferente a problemática do uso de telefone celular no trânsito. Há razão, também, para a polêmica nos dias de hoje, sobre o esforço da Ford em disponibilizar ao motorista a Internet dentro de seu carro - ainda que com interação por voz...

[4] Landauer (1962)

A Tabela 2.1 a seguir resume os subsistemas do MPIH, com seus respectivos parâmetros.

	Sistema Perceptual	Sistema Motor	Sistema Cognitivo
Memórias	dmiv=200[90~1000]ms dmia=1500[900~3500]ms umiv=17[7~17]letras umia=5[4.4~6.2]letras		dmcd=7[5~226]s dmld=infinito umcd=7[5~9]chunks umld= ?
Tipo de Código	kmiv=físico kmia=físico		kmcd=acústico/visual kmld=semântico
Processador	tp=100[50~200]ms	tm=70[30~100]ms	tc=70[25~170]ms

TABELA 2.1 - PARÂMETROS PRINCIPAIS DO MPIH

O Modelo do Processador de Informação Humano, obviamente como em qualquer modelo, não tem a pretensão de captar a complexidade e grandeza associados aos mecanismos humanos utilizados no nosso processo de perceber, pensar e agir. Entretanto, é uma aproximação bastante razoável para as tarefas relacionadas à avaliação e predição da performance humana ao interagir com computadores, especialmente nos aspectos ergonômicos envolvidos na interação. A seguir, mostraremos alguns exemplos extraídos da literatura em Fatores Humanos, que exemplificam e ao mesmo tempo comprovam a adequação do modelo para explicar aspectos de interação humano-computador.

EXEMPLO 1.

> Na simulação gráfica de um jogo de bilhar, existem ocasiões em que uma bola deve bater em outra bola, causando o movimento da segunda. Depois da colisão, em quanto tempo o movimento da segunda deve ter início para que o usuário "perceba" (tenha a ilusão de) uma causalidade?

De acordo com o MPIH, para que a colisão pareça causar o movimento da segunda bola, este deve ocorrer dentro de um ciclo do processador perceptual (Tabela 2.1). Resultados experimentais também comprovam essa medida. Experimento[5] reportado em Card *et al* (1983) mostra que a causalidade é percebida como uma função do tempo entre eventos associados aos movimentos das duas bolas, isto é, do tempo que decorre entre o final do movimento da primeira e o início do movimento da segunda bola. O gráfico da Figura 2.5 mostra 3 tipos de causalidade percebidas. A percepção da causalidade imediata termina em torno de 100 ms, isto é, quando o intervalo entre

[5] Michotte, (1963)

o final do primeiro e início do segundo movimento ocorre dentro de um ciclo do PC, e os eventos são considerados independentes em torno dos 180ms.

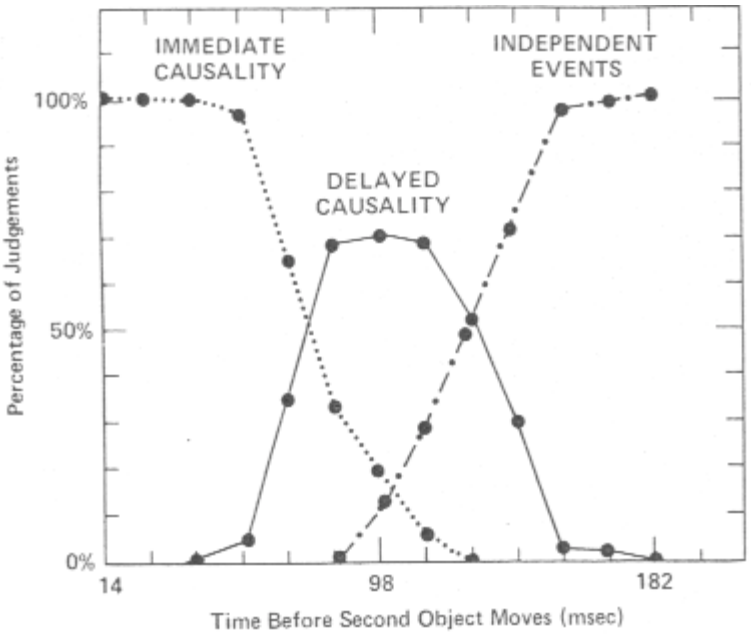


FIGURA 2.5- PERCEPÇÃO DE CAUSALIDADE EM MOVIMENTOS (CARD *ET AL*. 1983, P50)

EXEMPLO 2:

> Imagine um usuário em frente a um monitor de vídeo realizando a seguinte tarefa: são apresentados ao usuário dois símbolos, um de cada vez. Se o segundo for igual ao primeiro ele deve pressionar uma determinada tecla (X), caso contrário, outra tecla (Z). Qual será o tempo que decorrerá entre o sinal e a resposta para a tecla X?

De acordo com o MPIH, no tempo *t=0* o segundo símbolo aparece na tela. Ele é então transferido para a MIV e MCD. Há uma comparação do código visual dos dois símbolos. O PC, então, traduz o resultado (bem sucedido) da comparação em comando motor. O PM, então aciona a tecla X. O tempo necessário, será dado por $tp+2tc+tm$=310 ms, em um intervalo possível de 130 a 640 ms.

Embora essa tarefa seja bastante simples, ela sugere a forma como a complexidade de determinada tarefa pode comprometer o tempo gasto na interação. Quanto maior

a complexidade, maior o número de ciclos de processadores que estarão envolvidos e maior será o tempo gasto.

EXEMPLO 3:

> Suponha que a um conjunto de indicadores de erro em determinado sistema, devem ser atribuídos mnemônicos de 3 letras. Quando o sistema quebra, o operador deve escrever um conjunto de até 5 palavras. O que será mais importante usar para prevenir erros de transcrição: códigos similares no som ou no significado?

De acordo com o MPIH, na MCD o código utilizado é acústico ou visual. Resultados experimentais reportados em Card *et al.* (1983) dão conta de que a interferência na memória depende do tipo de representação mental do item, sendo a memória de curta duração mais suscetível à interferência acústica do que semântica, portanto. Isto é, itens que "soam" parecido são mais suscetíveis a interferências do que os que têm significado parecido. No caso do exemplo 3, considerando que os códigos devem ser escritos imediatamente após o evento, e que esses códigos ficam na MCD durante transcrição, deve-se pensar nas possíveis interferências na MCD. Códigos acústicos, isto é, códigos que soam de forma similar devem, portanto, ser evitados.

EXEMPLO 4:

> Considere a situação onde um usuário deve aprender a usar um novo editor de textos, orientado a comandos, idêntico a um usado anteriormente, exceto pelos nomes dos comandos (ERASE em vez de DELETE, por exemplo). Depois de algum tempo, qual será a facilidade ou dificuldade em retornar ao uso do editor antigo?

Trata-se de uma questão associada à recuperação de informação da MLD e os princípios 2 e 3 aplicam-se neste caso. De acordo com o modelo, a dificuldade em lembrar depende da interferência na MLD, isto é, de quais outros itens podem ser recuperados pelas mesmas pistas. Conforme o usuário acumula novos *chunks* na MLD, torna-se mais difícil recuperar *chunks* velhos que são semanticamente similares aos novos, uma vez que a codificação na MLD usa código semântico. O leitor que já experimentou várias ferramentas computacionais similares já deve ter experimentado essa dificuldade.

EXEMPLO 5:

> Consideremos o movimento da mão de um usuário em direção a determinado alvo, como normalmente se faz, por exemplo, para alcançar o

mouse a partir do teclado conforme ilustrado na figura 2.6. Quão rápido pode ser esse movimento?

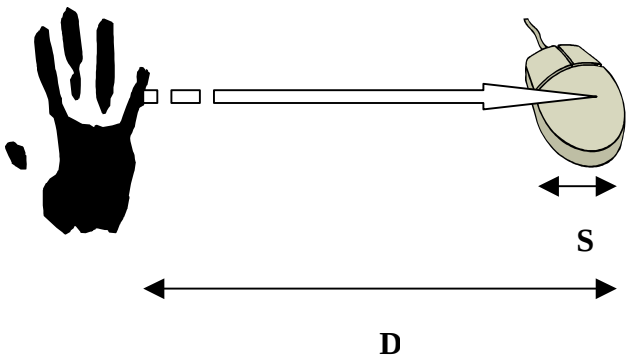


FIGURA 2.6 - MOVIMENTO DA MÃO EM DIREÇÃO A UM ALVO

Segundo o MPIH, o movimento da mão não é contínuo, mas é composto de uma série de movimentos discretos acompanhados de micro-correções. Para cada correção são necessários um ciclo do PP (para observar a mão), mais um ciclo do PC (para decidir sobre a correção) e mais um ciclo do PM (para fazer a correção). Portanto, o movimento total é dado pela expressão $n(tp+tc+tm)$, onde n é o número de intervalos de 240 ms necessários para alcançar o alvo.

Assumindo que a precisão relativa do movimento é constante, temos:

$x_i/x_{i-1}=e$, $x_1=ex_o=eD$, $x_2=ex1=e(eD)$ ... $x_n=e^nD$

onde:

$x_o=D$ (distância do ponto inicial ao alvo)

x_i é a distância ao alvo depois do i ésimo movimento corretivo

e $e<1$ (erro constante)

A mão pára de se mover quando está dentro do alvo:

$e^nD<=S/2$ e $n=-log_2(2D/S)/log_2e$

O tempo de movimento *Tpos* é dado por:

$Tpos=n(tp+tc+tm)$

$Tpos=Im\ log_2(2D/S)$ onde $Im=-(tp+tc+tm)/log_2e$

Em valores tomados experimentalmente, $e = 0.07$ e Im = 63 ms

Essa última equação, chamada de *Fitts's Law*, bastante utilizada na prática de ergonomia, define o quinto princípio do MPIH:

Princípio n. 5 *Fitts's Law*

O *tempo necessário para mover a mão para um alvo depende somente da precisão relativa requerida, isto é, a razão entre a distância ao alvo e seu tamanho.*

Esse princípio pode ser empregado, por exemplo, para determinar a melhor posição para determinadas teclas de função em interfaces, medindo o tempo que seria gasto nos movimentos da mão.

Mais 4 princípios de operação constituem o MPIH e serão apresentados resumidamente a seguir, para completeza do modelo.

Princípio n. 6 *Lei da Prática*

O tempo Tn necessário para realizar uma tarefa na n-ésima tentativa é dado por:
$T_n=T_1n^{-a}$ *onde* $a=0.4[0.2\sim0.6]$

Esse princípio estabelece que o tempo para fazer uma determinada tarefa decresce com a prática. Dados experimentais suportam esse princípio.

Princípio n. 7 *Princípio da Incerteza (Hick's Law)*

O tempo T de tomada de decisão aumenta com a incerteza sobre o julgamento da decisão a ser feita e é dado por:
T=IcH onde H é a entropia da decisão e Ic=150[0~157]ms/bit
para n alternativas igualmente prováveis $H=\log_2(n+1)$,
para alternativas com diferentes probabilidades pi de ocorrência,
H= somatória(em i) de $p_i(\log_2(1/p_i+1)$

Esse princípio pressupõe que a tarefa pode ser analisada como uma seqüência de decisões tomadas pelo Processador Cognitivo. A relação entre o tempo requerido e o número de alternativas não é linear porque as pessoas aparentemente podem organizar o processamento hierarquicamente.

Princípio n. 8 *Princípio da Racionalidade*

Uma pessoa age de forma a alcançar suas metas através de ação racional, determinada pela estrutura da tarefa e suas entradas de informação e limitada pelo seu conhecimento e habilidade de processamento:
Metas+Tarefa+Operadores+Entradas+Conhecimento+Limites de Processamento -> Comportamento

Esse princípio estabelece que muito da complexidade do comportamento humano deriva, não da complexidade do Humano em si, mas da complexidade da tarefa/ambiente no qual a busca da meta está acontecendo.

Princípio n. 9 *Princípio do Espaço do Problema*

> *A atividade racional na qual as pessoas se engajam para resolver um problema pode ser descrita em termos de (1) um conjunto de estados do conhecimento, (2) operadores para mudar um estado para outro, (3) restrições na aplicação desses operadores, (4) conhecimento para decidir que operador aplicar em seguida.*

O MPIH enquanto modelo é uma aproximação bastante boa para ser útil na análise e entendimento de opções de design e de periféricos envolvendo operações sensório motoras e cognitivas do usuário. Um experimento bastante interessante reportado em Card *et al* (1983) analisa, por exemplo, a performance de usuários na tarefa de seleção de texto em diferentes tipos de periféricos, mais especificamente *mouse, joystick, step keys e text keys*. Veremos mais adiante, ainda neste capítulo, o GOMS: um modelo usado no design e avaliação da interação H-C, derivado diretamente do MPIH através de seus componentes e princípios de operação.

Das muitas definições e conceituações para interface que abordamos neste livro, Laurel (1990) apresenta uma definição para o conceito de interface como uma *superfície de contato que reflete as* ***qualidades físicas*** *das partes que interagem entre si* (grifo nosso). Portanto, enquanto os modelos que a Psicologia nos oferece são importantes para entendermos nosso comportamento com relação aos artefatos criados pela nossa cultura, conhecer os mecanismos subjacentes ao processamento perceptual, cognitivo, motor, e a memória humana é fundamental ao estudo de interfaces. Nas próximas sessões estaremos estudando principalmente a percepção humana e as bases neurais da memória.

MECANISMOS DA PERCEPÇÃO HUMANA

O usuário deve "perceber" a informação apresentada na interface através dos sinais que a constituem. Principalmente quando consideramos sistemas computacionais baseados em multimídia ou em realidade virtual, torna-se clara a necessidade de entendimento de outras modalidades perceptuais, além do "ver" propriamente. Ficaremos impressionados se pensarmos no número de fenômenos que não somos capazes de perceber: a trajetória de uma bala atirada de uma arma, a luz infra-vermelha, o crescimento de uma planta, etc. Nosso objetivo com esta seção é explorar os mecanismos da percepção humana para entender sua influência no design de interfaces.

Várias teorias tentam explicar a maneira como percebemos; as construtivistas acreditam que nossa visão de mundo é construída de forma ativa por informação obtida do ambiente somada ao conhecimento previamente armazenado. Nas teorias construtivistas a informação que captamos é construída, envolve processos cognitivos, portanto. O paradigma construtivista explora a maneira como reconhecemos determinado objeto e fazemos sentido de determinada cena.

Em outra linha de teorias, denominadas ecologistas (Preece *et al.*, 1994), percepção é um processo direto que envolve a detecção de informação do ambiente e não requer quaisquer processos de construção ou elaboração. A noção de "*affordance*", que foi apresentada no Capítulo 1 e que ainda será bastante discutida neste livro, é derivada do entendimento da linha ecologista para a percepção: os objetos carregam certas características que dirigem nossa percepção sobre eles. As leis de Gestalt para a organização perceptual – proximidade, similaridade, fecho, continuidade, simetria - são exemplos de fatores que explicam a forma como características no sinal que nos é apresentado nos levam a perceber (ou deixar de perceber) determinada informação. A Figura 2.7 ilustra esse fenômeno.

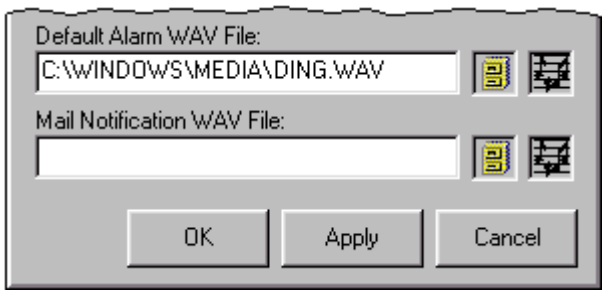


FIGURA 2.7 - PERCEPÇÃO E *AFFORDANCE*

A imagem da Figura 2.7 é um trecho de interface do software *Time & Chaos* (*Shame* 1999). Os pares de figuras colocadas à direita são botões, embora não pareçam. Note que botões têm a aparência dos 3 colocados abaixo, em seqüência (*OK, Apply, Cancel*). Se o usuário clicar sobre a imagem da direita antes de haver selecionado um arquivo, nenhum feedback acontece, sugerindo que o clique não teve efeito aparente, o que levará o usuário a acreditar que as imagens são realmente apenas decorativas. Os pares de figuras colocados à direita na tela não possuem o *affordance* de botões de comandos, como deveriam.

Entender os mecanismos da percepção humana envolve entender os processos psicológicos em operação e as redes neurais envolvidas. A primeira pergunta que se faz é *Como os sinais externos que chegam aos órgãos sensoriais são convertidos em experiências perceptuais significativas?*

O modelo mais simples para responder à pergunta é baseado na teoria do reconhecimento por casamento de padrões. Para a operação de casamento de padrões, deve existir alguma representação – um *template* – para cada um dos padrões a ser reconhecido. Se tivermos um padrão definido, por exemplo, a letra "L", para reconhecê-la basta procurar imagens que se "casem" com ela. A retina é composta de centenas de milhares de células nervosas sensíveis à luz, chamadas receptoras. Quando um padrão de luz estimula um certo conjunto de receptores, o detector do "L" responde. A Figura 2.8 ilustra esse processo.

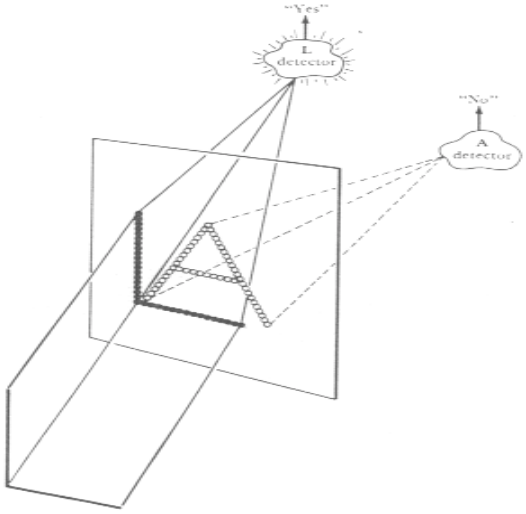


FIGURA 2.8 - RECONHECIMENTO POR CASAMENTO DE PADRÕES (LINDSAY E NORMAN 1972, P. 3)

Obviamente rejeita-se facilmente esse modelo para explicar o reconhecimento humano de padrões. Esse processo simples não funciona se a letra é apresentada com outra orientação ou com tamanho variável. Sistemas computacionais recorreriam a um pré-processamento. Um sistema mais poderoso e flexível é necessário para dar conta da capacidade humana de reconhecimento de padrões. Então, *como a informação que chega a nossos órgãos dos sentidos é interpretada?*

Nossos erros são reveladores ao mostrar pontos onde a nossa interpretação falha. Vários "truques" são usados por artistas da pintura e da fotografia, violando intencionalmente regras de construção da percepção, e são interessantes para nos revelar o fenômeno humano da percepção. Por exemplo, para que o processo seja aparente, a quantidade de informação disponível pode ser reduzida. Degrada-se a imagem completamente tornando a interpretação difícil. Façamos uma experiência simples, verificando como organizamos imagens degradadas. Olhe para as imagens da Figura 2.9.

(A) (B)

FIGURA 2.9 - ORGANIZANDO IMAGENS DEGRADADAS

Note que para "ver" o cachorro em (B)[6] ou a moça em (A)[7] nós adicionamos informações que não estão presentes na imagem. Se alguém pede a você que procure encontrar o cachorro ou a moça, fica mais fácil de vê-los. Além disso, uma vez ue se vê o cachorro ou a moça é muito difícil não vê-los mais. Isso explica um fenômeno interessante presente até no nosso dia a dia, e que não se restringe à percepção de imagens visuais: *Quando se olha para o que se quer ver é mais fácil "ver".*

Outro artifício, explorado genialmente por Salvador Dali e Mauritis Cornelis Escher, entre outros, consiste em colocar organizações competitivas na imagem, de forma a

[6] Foto de R.C. James (em Lindsay e Norman, 1972, p.8) ; [7] autor desconhecido

tornar possível o conflito de interpretações da mesma imagem. As Figuras 2.10 (A)-2.10(F) ilustram esse fenômeno.

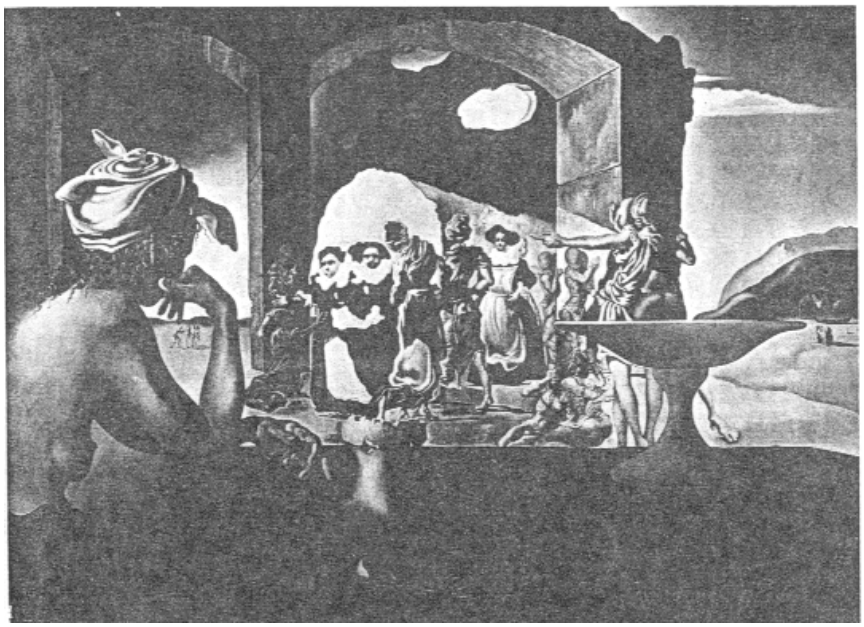


FIGURA 2.10 (A) - ORGANIZANDO IMAGENS QUE COMPETEM
Salvador Dalí, *The Slave Market with Disappearing Bust of Voltaire*

FIGURA 2.10 (B)[8] **-** Moça ou Velha?

[8] Autor desconhecido

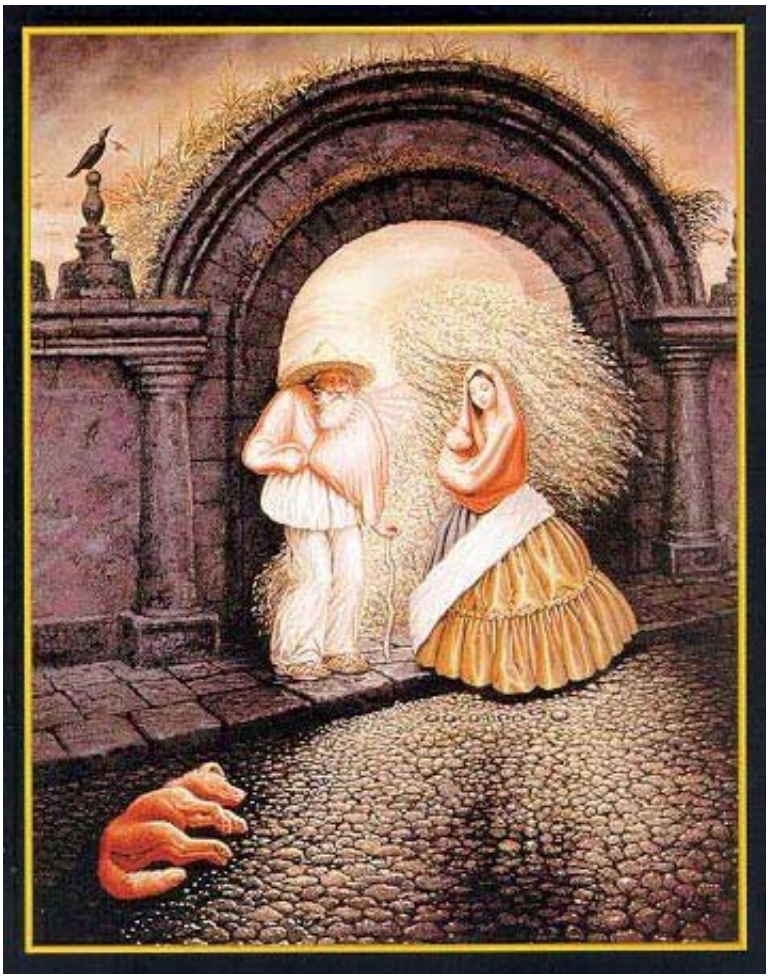


FIGURA 2.10 (C)[9]

Quantas faces estão presentes na figura ao lado?

FIGURA 2.10 (D)[9]
Pato ou coelho?

[9] Autor Desconhecidos

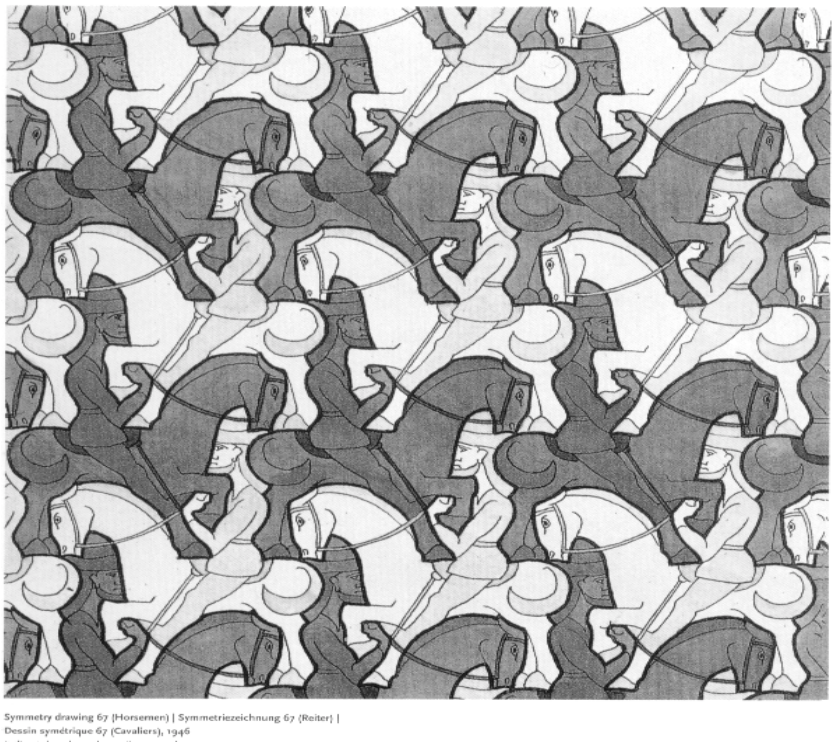


FIGURA 2.10 (E) - M.C. Escher, *Cavaleiros*, 1946

FIGURA 2.10 (F)[9] - Índio ou Esquimó?

[9] Autor desconhecido

Uma imagem pode ser ambígua por falta de informação relevante ou por excesso de informação irrelevante, como mostram estes últimos casos, revelando diferentes mecanismos de construção da informação. Note que temos dificuldade em interpretar a imagem de duas maneiras diferentes ao mesmo tempo: Na Figura 2.10 (A) ou vemos as freiras, ou vemos o busto do Voltaire, mas não os dois ao mesmo tempo. Em (C) ou vemos o olho do velho ou o homem. Em (E) ou vemos cavaleiros em uma direção ou na outra, em (B) a moça ou a velha, etc.

É interessante observar que esses fenômenos têm um análogo em nossa percepção auditiva. Por exemplo, não conseguimos "ouvir" duas conversas ao mesmo tempo e em festas, às vezes tentamos extrair uma conversa entre tantas de fundo. A música clássica é outro exemplo riquíssimo de expressões "figura-fundo" que conhecemos melhor do contexto de imagens visuais. A riqueza da música e a forma como ela nos agrada mais ou menos parece depender de organizações perceptuais que possam instigar nossa percepção auditiva.

Um terceiro artifício, para entendermos nosso processo de percepção consiste em colocar-se na imagem visual uma organização sem sentido para ver como a experiência passada afeta o processo. Olhe para a Figura 2.11.

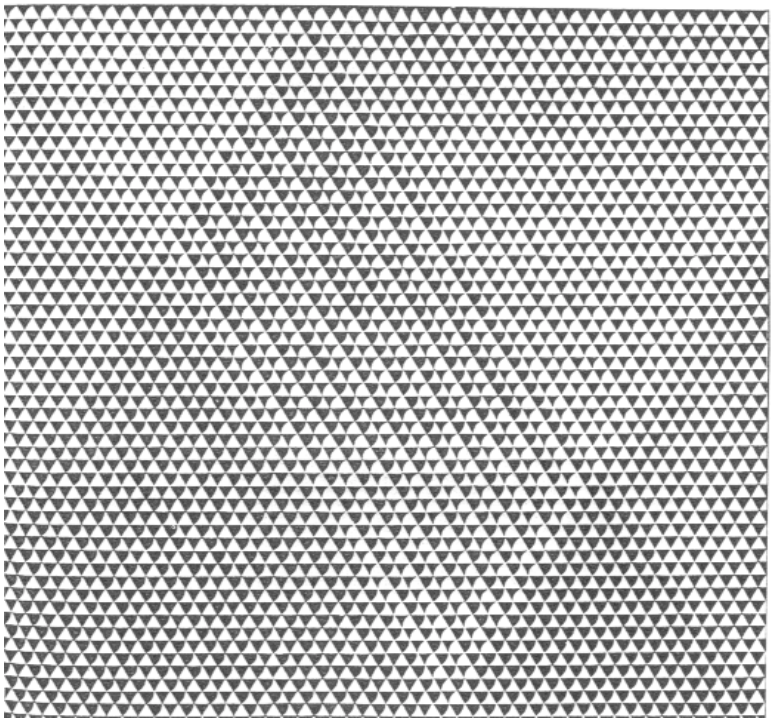


FIGURA 2.11(A)[10] ATRIBUINDO SENTIDO À IMAGEM

[10] Bridget Riley, *Tremor* (em Lindsay e Norman, 1972, p. 13)

Na figura 2.11(A), o que você vê? Escaninhos horizontais, verticais ou uma seqüência de triângulos? Há uma organização flutuante: uma forma ou outra pode ser observada. Este último exemplo mostra que os processos visuais e perceptuais impõem uma organização à imagem, mesmo que o artista tenha evitado deliberadamente colocar formas de organização. A interpretação da imagem é realizada pela segregação de grupos que tenham forma similar, que são tratados como unidades ou "pontos focais" (um tipo de quebra no padrão repetitivo).

Na figura 2.11(B), por sua vez, quantos animais você "vê"?

FIGURA 2.11(B)[11]
ATRIBUINDO SENTIDO À IMAGEM

[11] Autor Desconhecido

Outro fenômeno interessante é nossa percepção do espaço e profundidade. Assumindo que vivemos e nos movemos em um mundo 3D, faz sentido que nosso aparato visual tenha evoluído para colocar uma representação 3D nas imagens que vê. Olhe para a obra de Magritte[12], na Figura 2.12; você vê dois cones?

FIGURA 2.12 - ATRIBUINDO PROFUNDIDADE À IMAGEM

Sempre que um padrão visual no qual linhas e arestas convergem, há duas opções de interpretação: trata-se de objetos bidimensionais vistos diretamente – as linhas realmente convergem, ou objetos tridimensionais vistos em perspectiva – as linhas são paralelas. A escolha da interpretação parece baseada em análise das evidências disponíveis. No caso da Figura 2.12, em particular, há ainda a importância de informação do contexto. Não lidamos com as coisas isoladamente. Quando informação sensorial é colocada junto, uma imagem consistente do mundo deve ser produzida. Note como o cavalete parece suportar o quadro ao mesmo tempo em que a imagem sugere uma janela, provocando intencionalmente percepções diferentes da figura cônica.

Outro aspecto importante da organização de informação visual pode ser demonstrado por "ilusões". Lindsay e Norman (1972, p. 12,13), mostram uma experiência simples que pode ser feita com uma folha de papel dobrada, para

[12] René Magritte, *Les Promenades d'Euclide*, 1953

entendermos o fenômeno chamado “paralaxe do movimento”. A Figura 2.13 mostra como o cenário experimental deve ser montado.

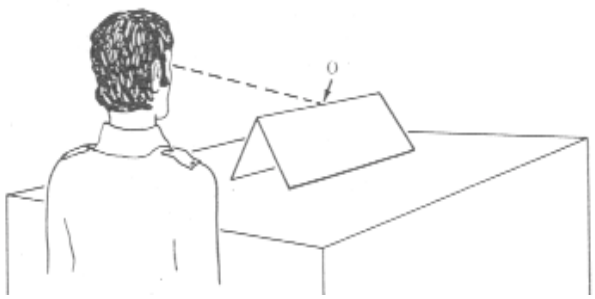


FIGURA 2.13 -EXPERIMENTANDO A ILUSÃO DE MOVIMENTO (LINDSAY E NORMAN 1972, P. 12-13)

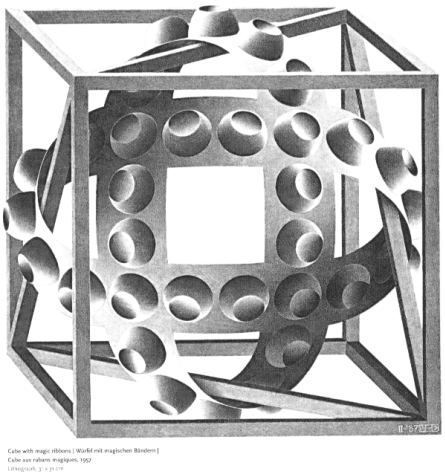
Observe o papel dobrado com um olho apenas, por alguns segundos, fixamente. Ele parecerá colocado tanto em pé (com sua aresta maior perpendicular à mesa) quanto deitado (com sua aresta paralela à mesa). Na primeira situação ele fica com sombras diferentes, parecendo luminoso. Se movermos a cabeça de um lado para outro, o papel parecerá se mover também, como se fosse de borracha (o que você “sabe” que não é). A imagem do ponto mais próximo do papel move-se, na retina, mais rápido do que a imagem do ponto mais distante. Para manter a consistência, o objeto (papel) “dança”. Ou seja, todos os dados sensoriais são usados para construir uma interpretação consistente do mundo visual. Esse fenômeno é conhecido como paralaxe do movimento.

As chamadas “ilusões de ótica” mostram que a nossa habilidade em ver coisas em profundidade não depende da familiaridade que temos com os objetos representados. Muitas dessas ilusões são decorrentes de um fenômeno conhecido como “gradiente da distância”. Considere um pedaço de papel retangular, com linhas horizontais igualmente espaçadas; quando inclinado em relação ao campo de visão, as linhas mais distantes da imagem da retina parecerão mais próximas e seu comprimento também parecerá menor. Ilusão de profundidade pode ser criada com o uso de perspectiva em padrões que se repetem indefinidamente. O contexto também exerce influência em imagens não necessariamente familiares. Na Figura 2.14 olhe para o desenho à esquerda e responda: qual das linhas intercepta a linha vertical em ângulo reto? E no desenho à direita? A Figura 2.14 mostra que quando informação de profundidade é adicionada, o ângulo reto parece obtuso e o obtuso parece reto.

FIGURA 2.14 - A INTERFERÊNCIA DA PROFUNDIDADE

Outra maneira de demonstrar a operação que fazemos de colocar objetos em 3D durante sua interpretação é olhar para figuras "impossíveis". As partes são compreendidas individualmente, mas são conflitantes na interpretação global da cena. Escher criou trabalhos fantásticos explorando esse fenômeno. Alguns de nossos preferidos são mostrados a seguir na Figura 2.15.

M.C. Escher, *Moebius Strip*, 1961



M.C. Escher, *Cube aux rubans magiques*, 1957

FIGURA 2.15 (A) E (B)
ORGANIZAÇÕES IMPOSSÍVEIS

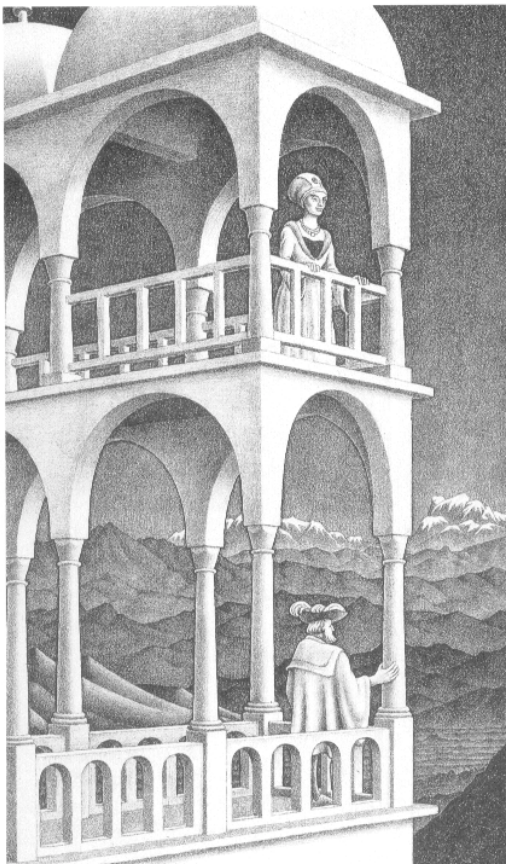


FIGURA 2.15(C)
ORGANIZAÇÕES
IMPOSSÍVEIS

M.C.Escher, *Belvedere* 1958 (detalhe)

Os exemplos mostrados ilustram características de nossos processos de percepção. *Como explica-los*? *Que informações o sistema nervoso extrai dos sinais chegando aos órgãos dos sentidos?*

Lindsay e Norman (1972) propõem que olhemos para as anomalias da percepção, para entender esse processo. Olhe para a imagem da Figura 2.16.

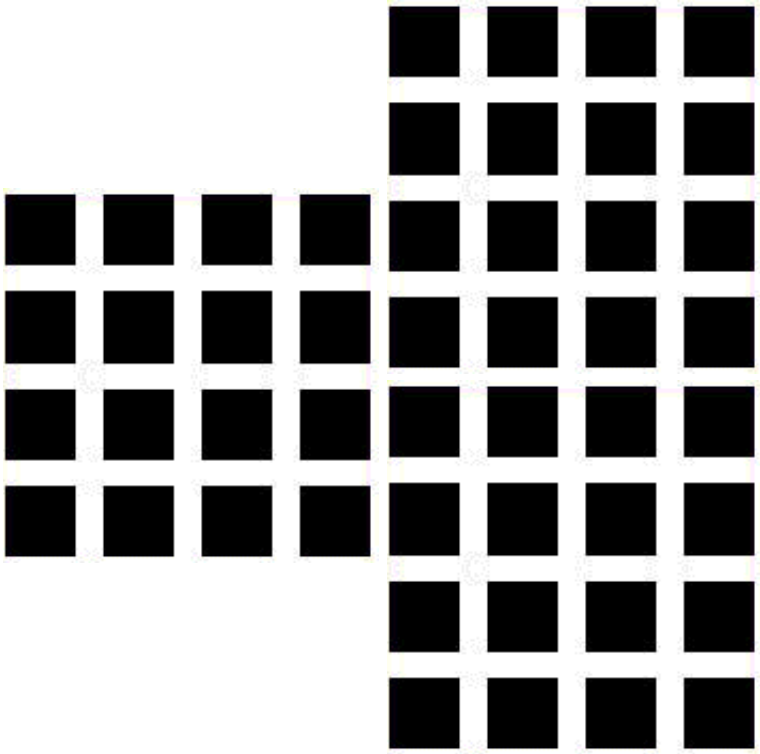


FIGURA 2.16 - A GRADE DE HERING

Pontos cinza são enxergados na intersecção dos quadrados escuros, mesmo não estando presentes, exceto naquela onde você fixa o olhar. Esse fenômeno é explicado pelo princípio da análise sensorial que estabelece que células neurais interagem umas com as outras. Receptores de uma parte da imagem visual são afetados pela operação de receptores para partes vizinhas. No único ponto do olho onde os receptores não interagem muito com os outros, área onde o olho está focando, o escurecimento da interseção não acontece.

O movimento do olho é outra fonte de informação sobre o mecanismo de extração de informação sensorial. Os olhos estão em movimento constante, conforme já discutido na apresentação do MPIH. Se o movimento pára, as imagens desaparecem. Movimentos da imagem visual sobre a superfície da retina podem ser parados através de técnicas e aparato de espelhos colocados no olho. Com calibração cuidadosa é possível fazer a imagem se mover no mesmo ângulo visual do olho, de

modo que a imagem vista na retina não muda sua posição apesar dos movimentos do olho. Quando a cena visual é vista através desse aparato, depois de alguns segundos os padrões começam a desaparecer. A Figura 2.17 ilustra esse resultado.

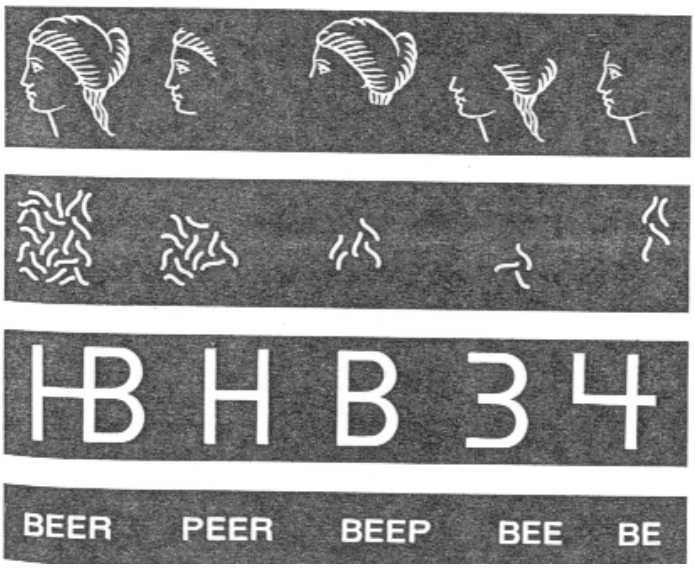


FIGURA 2.17 - O DESAPARECIMENTO DA CENA EM PARTES (LINDSAY E NORMAN 1972, P. 35)

Uma explicação de Lindsay e Norman (1972) ao fenômeno sugere que enquanto informação é fornecida pelos receptores neurais no olho, os detectores continuam a responder e o padrão é visto completamente. Quando o olho pára seus movimentos, os receptores cessam suas respostas.

A exemplo do olho, todos os sistemas sensoriais parecem requerer mudanças na estimulação para manter a percepção. Sistemas auditivos têm movimento embutido neles. Não existe sinal auditivo constante, por causa de padrões de pressão do ar. Em relação ao tato, depois de algum tempo não percebemos mais a pulseira do relógio, a menos que ela se mexa no braço. O mesmo ocorre com o olfato, um cheiro de perfume é notado menos por quem o está usando.

Outro efeito interessante para estudo da percepção é o chamado “efeito posterior”: visão prolongada de uma imagem deixa sua marca em percepções futuras; se observarmos prolongadamente determinado movimento, posteriormente o ambiente ao redor parecerá estar em movimento contrário. Esse efeito é muito observado também com cores; após intensa exposição a determinada cor - vermelho, por exemplo, olhando para o branco, enxergamos verde (cor complementar ao vermelho).

Embora pareça não haver dúvida sobre a existência no ser humano de detectores de padrões específicos, o tipo exato de operação desses detectores ainda não é bem conhecido. Por exemplo, Lindsay e Norman (1972) mostram que a interpretação de padrões pode ser feita pela própria pele. Situações de laboratório mostram resultados de pesquisa sobre visão através da pele. Uma câmera de vídeo acoplada a um conjunto de micro-vibradores são conectados às costas de um sujeito. Regiões de alta intensidade luminosa ativam os micro-vibradores correspondentes. Resultados mostram que os sujeitos conseguem distinguir linhas verticais de horizontais, formas geométricas, objetos comuns do dia a dia e mesmo posição relativa e profundidade relativa desses objetos em uma cena, quando eles próprios movimentam a câmera.

Como vimos nesta seção, perceber é muito mais do que ver. O conhecimento sobre os fenômenos da percepção humana é cada vez mais necessário ao designer de interfaces, mesmo sem considerarmos as propostas de interface 3D e realidade virtual. A expressão gráfica para interfaces ao mesmo tempo em que oferece ao usuário facilidades em relação à interface orientada a comandos, passa a exigir processamento perceptual cada vez maior. A demanda por memorização e a carga cognitiva exigida pelas interfaces baseadas em linguagem de comandos é deslocada para o processamento perceptual e viso motor nas interfaces gráficas. A ausência de conhecimento de designers nessa área tem mostrado seu reflexo em interfaces de todos os gêneros de software. Tente ler o texto na imagem a seguir, Figura 2.18, (*Shame*, 1999) e note o esforço requerido.

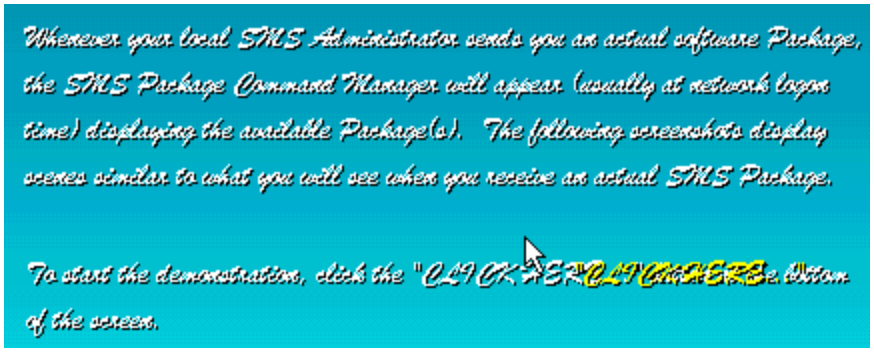


FIGURA 2.18 - TRECHO DE TELA PARA LEITURA (SHAME, 1999)

Essa imagem era parte de um tutorial realizado em uma grande organização, sobre como usar um novo sistema de software. A escolha da cor e tipo de fonte, cor de fundo, além de tornar o texto ilegível, mostra um desconhecimento total dos fenômenos relacionados ao esforço perceptual requerido do usuário.

Outro exemplo, Figura 2.19, (*Shame*, 1999) mostra trecho de interface onde o designer parece desconhecer que fenômenos de percepção levam o usuário de tal aplicação a tentar clicar sobre *"Subscriber"* ou *"Contact"*. O designer foi bastante infeliz ao escolher destacar os rótulos de seção, usando uma aparência de relevo que os tornam, em forma, semelhantes aos botões!

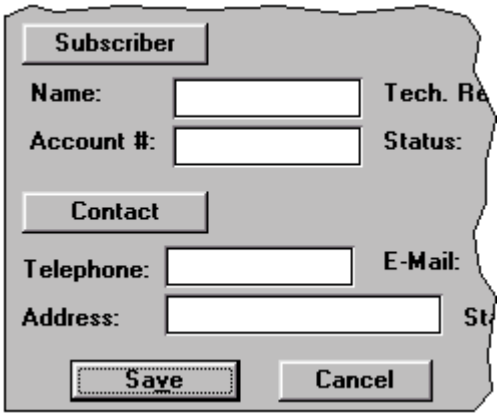


FIGURA 2.19 - PARTE DE INTERFACE PARA INTERAÇÃO (SHAME, 1999)

Assim como é necessário ao designer estar consciente das possibilidades e restrições do usuário em função de seus mecanismos perceptuais, a próxima seção estuda e discute os mecanismos humanos de memória. Nosso objetivo é possibilitar escolhas de design mais informadas pelo funcionamento de nossos mecanismos de memória.

AS BASES NEURAIS DA MEMÓRIA HUMANA

A literatura especializada da psicologia experimental define pelo menos três tipos diferentes de memória: a memória de informação sensorial (chamada MIV e MIA no MPIH), a memória de curta duração (chamada MCD ou MT no MPIH) e a memória de longa duração (MLD). O primeiro desses sistemas de memória mantém uma "fotografia" do mundo como ele é recebido pelos nossos órgãos dos sentidos. Balançando a caneta em frente aos olhos pode-se ver a imagem do rastro atrás do objeto em movimento. Esse rastro do movimento dura, aproximadamente, 250 ms e equivale ao conceito de tempo de desbotamento. Na memória de curta duração, a informação retida não é mais a imagem completa de eventos que acontecem ao nível

sensorial, mas a interpretação imediata desses eventos. Por exemplo, quando uma sentença é falada, não nos lembramos dos sons que formam a sentença, mas das próprias palavras. A informação pode ser mantida indefinidamente na memória de curta duração, através de repetição e sua capacidade é limitada. (5+-2 *chunks*). Na memória de longa duração não há limite prático para sua capacidade de armazenamento. Responda rapidamente: *O que você comeu no jantar do último sábado?* O cérebro tem aproximadamente 10 bilhões de neurônios capazes de armazenar informações.

Olhando para as estruturas cerebrais envolvidas no armazenamento e recuperação de informação, o cérebro humano é dividido em regiões que diferem anatomicamente e por isso possuem nomes diferentes. Dois hemisférios se destacam: o direito e o esquerdo, cada um composto de quatro lóbulos. A matéria cinza que compõe o cérebro é chamada de córtex cerebral, sendo responsável pelas suas funções mais sofisticadas – processamento de imagem visual, pensamento, linguagem, etc. A Figura 2.20 apresenta de forma diagramática as principais regiões do cérebro humano.

Mudanças estruturais e químicas devem ocorrer no cérebro, como resultado da aquisição de novo conhecimento. Descrição acurada e completa de como o sistema nervoso armazena informação não existe, ainda. Entretanto, é bem aceita a teoria de que as atividades correntes de pensamento, processos de consciência e memórias imediatas (armazenamento sensorial e MCD) são mediadas por atividades elétricas. O impulso elétrico carregado pelo neurônio viaja do corpo de uma célula para um outro corpo de célula, através do axônio. O local onde o axônio faz contato com o corpo da célula é chamado junção sináptica. Existem dois tipos dessas junções: as excitadoras e as inibitórias. As junções excitadoras tendem a fazer o novo neurônio "disparar"; isto é, responder com seu próprio impulso. As inibitórias tendem a prevenir (impedir) o disparo. No sistema nervoso, um grande número de impulsos chegando a conexões excitadoras pode ser requerido para fazer o corpo da célula "disparar" outro impulso. A Figura 2.21 ilustra esse processo.

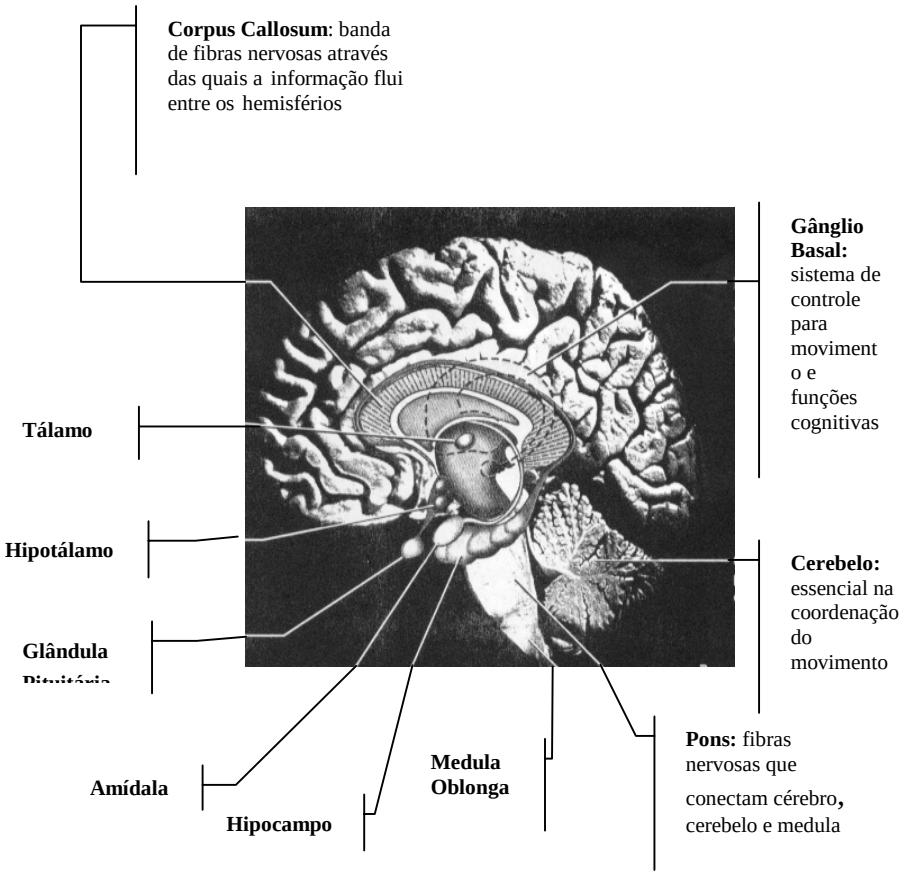


FIGURA 2.20 - ESTRUTURAS CEREBRAIS

Como a memória de uma entrada sensorial é mantida? Há pelo menos 3 teorias que explicam maneiras pelas quais o sistema nervoso poderia responder à presença da letra "A", por exemplo. Na primeira delas, uma única célula codifica a presença de cada item e responde quando o item é reconhecido. Na segunda teoria, a presença do "A" é apontada por uma configuração única de células neurais, que responde. Na terceira teoria, o "A" é apontado por um padrão especial de disparos neurais. A "lembrança" de que o "A" ocorreu seria mantida por circuitos "reverberatórios". O circuito mais simples que qualifica um circuito de memória é um *loop* fechado, como ilustra a Figura 2.22. Um sinal sensorial chegando inicia uma seqüência de eventos elétricos que persiste indefinidamente.

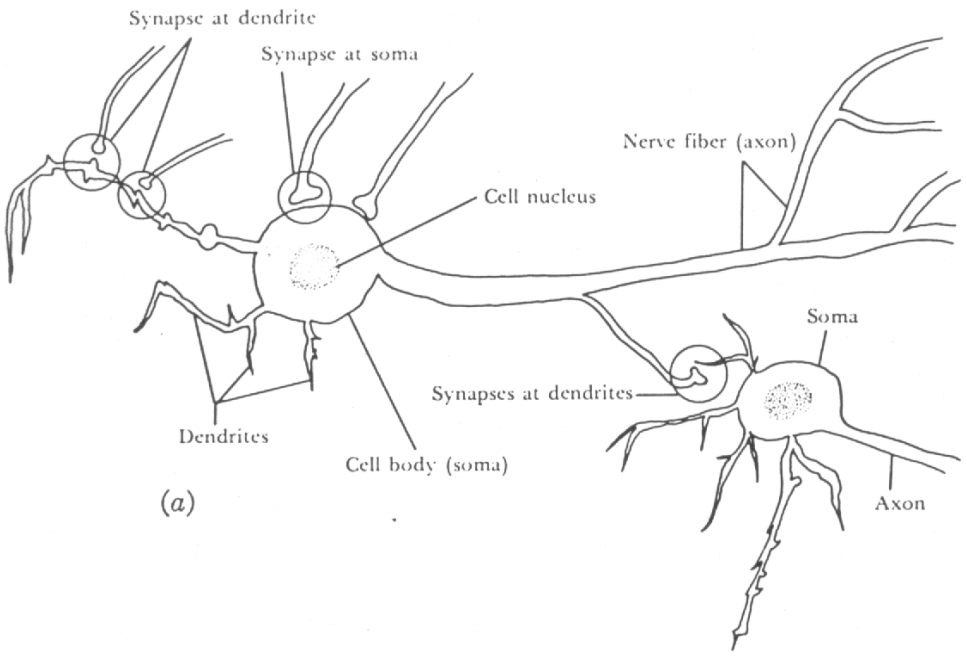


FIGURA 2.21 - ILUSTRAÇÃO DA BASE NEURAL (LINDSAY E NORMAN, 1972, P. 55)

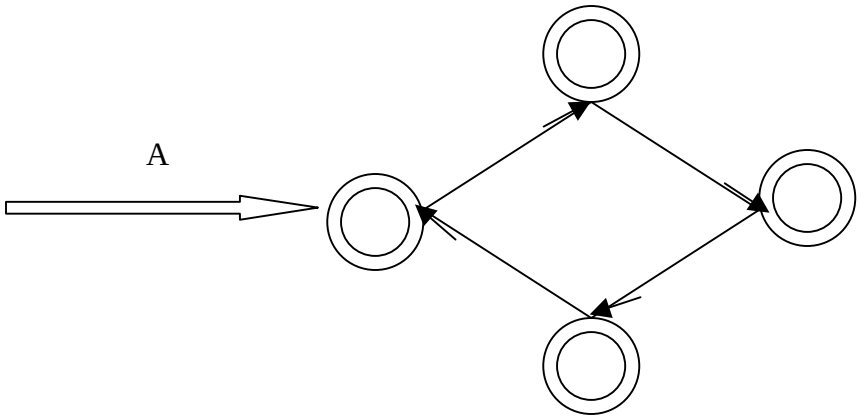


FIGURA 2.22 - MEMÓRIA PARA O EVENTO A

Mas, *o que pára a reverberação?* Entradas estranhas ao *loop* como, por exemplo: ocorrência de novos eventos, fadiga química nos neurônios ou nas sinapses, eventos anormais (pancada na cabeça, choque elétrico, etc.). A memória de curta duração consiste na ativação elétrica de um *loop* neural específico. É elétrica em sua operação, portanto.

E *como ela é representada?* Circuitos neurais específicos seriam construídos para memórias específicas. Atividade elétrica através desses circuitos representa uma ativação temporária deles.

A memória de longa duração é representada pela estrutura permanente dos circuitos neurais, que acontece através de um processo chamado consolidação. *Como aconteceria a consolidação?* Existem duas teorias principais: a primeira explica o processo de consolidação como uma codificação química na estrutura de moléculas de proteína em cada sinapse. Outra teoria explica a consolidação através do crescimento de novas junções sinápticas. Independentemente das teorias (mudança química ou crescimento neural), o resultado é que o efeito acontece nas sinapses.

Como seria a química da memória? As informações genéticas para cada organismo são armazenadas nas moléculas de DNA e transportadas por uma segunda molécula: o RNA. Portanto, há a possibilidade de processos químicos estarem envolvidos nos mecanismos da memória. Foi observado experimentalmente que ativação neural repetida e experiências de aprendizado têm efeitos mensuráveis na química do RNA. Experimento com ratos[13] reportado em Lindsay e Norman (1972) mostrou que após experiência de aprendizado, análise bioquímica do reservatório de RNA do sistema de orientação no cérebro do rato, mostrou concentração de RNA acima do normal. *Conteria essa alteração, informação sobre a tarefa?*

Um segundo experimento com planarian[14] (minhoca plana) – um animal com habilidade de regenerar dois corpos quando cortados na metade, reportado em Lindsay e Norman (1972) foi feito para investigar a hipótese de ser a memória codificada quimicamente ou ser retida em conexões neurais especiais. Foi ensinada determinada tarefa a uma planarian e logo após o animal foi cortado pela metade. Quando as duas metades estavam regeneradas, foi feito um teste sobre a tarefa aprendida. As duas responderam da mesma forma ao teste, indicando que o aprendizado foi retido. Esse resultado sugere que a memória (pelo menos desse animal) é codificada quimicamente, pois se fosse retida em conexões neurais

[13] Hyden e Egyhazi, (1964)
[14] McConnel, Jacobson e Kimble, (1959)

especiais na cabeça, o animal regenerado a partir da segunda metade (sem a cabeça) não conteria o conhecimento dos eventos aprendidos.

Os mesmos autores mostraram também que, no caso das planarian, há transferência de memória de um animal para outro. Esses animais são canibais. Uma planarian foi treinada em uma determinada tarefa, foi morta e dada como alimento a outra. Algum conhecimento sobre a tarefa que a primeira aprendeu parece ter sido transferido para a que a comeu. Forma similar de transferência foi reportada entre ratos e camundongos.

A memória de curta duração parece ser necessária para "segurar" a informação pelo período de tempo requerido para a consolidação. Durante o período de atividade elétrica que segue a ocorrência do evento, a memória para o evento torna-se consolidada em memória de longa duração. A aplicação de grande quantidade de correntes elétricas no cérebro interrompe a memória de curta duração. Desordens da memória após crises convulsivas ou acidentes com traumatismos cranianos ilustram e sustentam essa conceituação. Observações de pacientes pós-acidente relatam que *é como se houvesse uma linha de memória estendendo-se no tempo*. Os poucos minutos que antecedem imediatamente o acidente não são recuperados.

Algo mais do que a reverberação elétrica na memória de curta duração parece necessário para acontecer o armazenamento na memória de longa duração, entretanto. Casos de pacientes com anomalias pós-acidente mostram que não conseguem reter qualquer conhecimento novo, mesmo depois de repetição continuada da informação. Essas evidências sugerem que parece haver processos separados para retenção de memória antiga e aquisição de nova informação.

É bem aceita a hipótese de que memórias não estão em posições específicas, mas estão espalhadas através do cérebro. O cérebro é constituído de dois hemisférios que se comunicam por um conjunto de fibras nervosas chamado *corpus callosum* (ver figura 2.20). Cada órgão sensorial envia sua informação para ambas as metades do cérebro. Objetos à esquerda do ponto de fixação vão para a metade direita da retina e para a metade direita do cérebro. Analogamente acontece com os objetos à direita do ponto de fixação. Essa informação que chega nos dois hemisférios é coordenada para formar a percepção e memória correntes.

Nossos dois hemisférios parecem especializados. Evidências sugerem que o hemisfério esquerdo é especializado na linguagem, por exemplo. Observações reportadas em Lindsay e Norman (1972) dão conta de que em cirurgias onde o *corpus callosum* é secionado (para eliminar sintomas da epilepsia, por exemplo), os sujeitos mostram-se incapazes de verbalizar ou escrever sobre qualquer coisa que a metade direita do cérebro esteja monitorando. Embora seja incapaz de *produção* de linguagem, a metade direita do cérebro, é capaz de *reconhecimento* de linguagem.

Em resumo, o cérebro dos organismos superiores parece consistir de dois sistemas de processamento ligados por um conjunto de linhas de comunicação. Cada sistema recebe somente parte da informação sensorial que chega dos vários receptores sensoriais. As fibras de comunicação do *corpus callosum* parecem ser usadas para transferir as partes que faltam da mensagem, de modo que cada metade tenha uma representação completa do ambiente. Pelo menos algumas das funções superiores do cérebro humano são governadas por redes dedicadas de neurônios. O mapeamento de regiões especializadas continua e a descrição de sua operação interna continua desafiando os neuro-cientistas.

O entendimento de nossos sistemas perceptuais, motores, cognitivos e de memória continuam um grande desafio, ainda nos dias de hoje. Literatura mais recente - *Nature*, *Science*, etc. continuam mostrando casos inexplicáveis em geral associados a anomalias encontradas em pacientes após diferentes tipos de acidentes ou doenças do sistema nervoso central, que mostram quão ignorantes ainda somos, neste início de milênio, a respeito de como "funcionam" nossos maravilhosos sistemas. Uma publicação mais popular, mas não menos interessante é o livro *Um antropólogo em Marte*, de Oliver Sacks (1995), neurologista, que registra casos intrigantes ocorridos com alguns de seus pacientes.

O conhecimento da memória humana tem deixado seu rastro na história dos computadores, a começar das idéias iniciais de Vannevar Bush (1945):

> *Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it.*

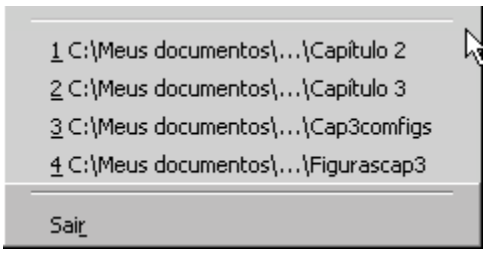
A seleção por associação em vez do uso de indexação é apenas uma das idéias de Bush derivadas da analogia, presentes nas interfaces atuais, que retomaremos no Capítulo 5.

O conhecimento das limitações de nossa memória, especialmente de curta duração, tem levado alguns designers (embora poucos) a criar soluções inteligentes para interfaces. Muitas vezes, ao entrarmos em determinado sistema com nosso *password*, somos surpreendidos por uma mensagem que não o aceita. Repetimos a operação achando que a digitação rápida pode ter falhado. Recebendo uma segunda negativa, somos neuroticamente inclinados a pensar que o pessoal da administração do sistema alterou alguma coisa, sem avisar...

O *Eudora Pro* para *Macintosh* criou uma solução simples para esse problema freqüente. É comum nos esquecermos o "Caps Lock" ligado e demoramos até nos darmos conta (lembrarmos) disso! Como a maioria dos *password* é sensível a escolha maiúscula, minúscula, a mensagem na Figura 2.23 (*Fame,* 1999) serve de "lembrança" dessa possibilidade.

FIGURA 2.23 - JANELA DO *EUDORA PRO* PARA *MACINTOSH*

Outra solução simples e interessante que nos ajuda a lidar com limitações de memória é o MRU (*Most Recently Used Files*), uma lista dos arquivos que acessamos mais recentemente (imagem ao lado). Esse é um recurso simples e interessante, presente em algumas aplicações, que nos poupa de atravessar a hierarquia de diretórios e arquivos, lembrando caminhos, nomes, etc.



A operação de inserção de figuras em algumas aplicações, vem acompanhada de *preview*, poupando-nos de lembrar que figura está associada a qual nome de arquivo, ou da demorada operação de carregar a figura para poder vê-la. A Figura 2.24 mostra esse recurso no *Microsoft Office2000*.

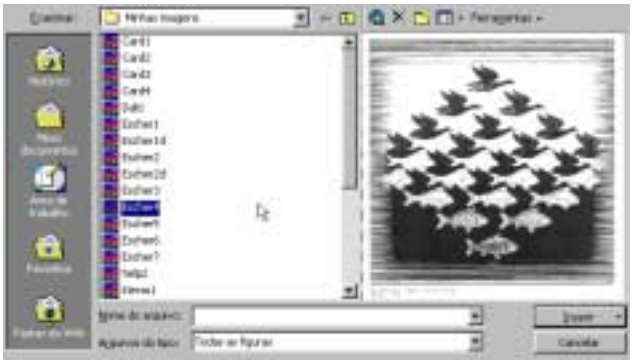


FIGURA 2.24 - JANELA NO *MICROSOFT OFFICE2000*

Conforme ilustramos, conhecendo os mecanismos humanos de memória, principalmente a capacidade de nossa memória de curta duração, pode-se criar recursos na interface que nos poupem de ter que lembrar detalhes necessários à interação, ao mesmo tempo em que nos liberem para ocupar a memória com informação relevante.

Estudado o modelo de processamento de informação humano (MPIH) e tendo olhado mais de perto os mecanismos da percepção e memória humanos, voltamos a apresentar um dos mais conhecidos conceitos em IHC, derivados do MPIH: o modelo GOMS.

O MODELO GOMS

A motivação para o GOMS foi fornecer um modelo de engenharia para a performance humana, capaz de produzir predições quantitativas a priori ou em um estágio anterior ao desenvolvimento de protótipos e teste com usuários. Ele prevê tempo de execução, tempo de aprendizado, erros, etc. identificando partes da interface associadas a essas previsões, de forma a orientar o re-design. *GOMS é atualmente o mais maduro dos modelos de engenharia em IHC [...] e pode ser realmente útil no desenvolvimento de sistemas no mundo real* (John e Kieras, 1996a, p.289).

GOMS baseia-se na premissa de que nosso entendimento sobre o desenvolvimento de sistemas pode ser melhorado se levarmos em conta as atividades cognitivas e de

processamento da informação do usuário. O modelo é conseqüência direta dos princípios números 8 e 9 do MPIH: o princípio da racionalidade e o princípio do espaço do problema. O acrônimo GOMS representa os componentes de um modelo GOMS: metas (G), operadores (O), métodos (M) e regras de seleção (S).
Com base na premissa de que "os usuários agem racionalmente para conseguirem alcançar as metas", quatro componentes básicos compõem, portanto, o modelo: (1) um conjunto de metas, (2) um conjunto de operadores, (3) um conjunto de métodos para alcançar as metas, (4) um conjunto de regras para seleção dos métodos.

Metas constituem uma estrutura simbólica que define o estado de coisas a serem alcançadas e determina o conjunto de métodos possíveis. A função dinâmica da meta é prover um "ponto de memória" para o qual o sistema pode retornar no caso de falha ou erro. Além disso, as metas carregam informação sobre o que é desejado, métodos disponíveis, o que já foi tentado, etc. Metas expressam o que o usuário deseja realizar com o software. Normalmente as metas formam uma hierarquia de submetas.

Operadores são atos elementares – perceptuais, cognitivos e motores - cuja execução é necessária para mudar aspectos do estado mental do usuário ou afetar o ambiente da tarefa. Operadores são as ações que o software possibilita ao usuário realizar. Embora possam ser definidos em diferentes níveis de abstração, os modelos GOMS os definem em termos concretos como o pressionar de um botão, o selecionar de um item de menu, etc.

Métodos são procedimentos necessários para conseguir realizar a meta. Relacionam-se à maneira como o usuário armazena conhecimento sobre a tarefa, e à seqüência condicional de submetas e operadores que usa na realização da tarefa; envolvem testes no conteúdo da memória de curta duração do usuário e no estado corrente do ambiente envolvido. Métodos são seqüências bem aprendidas de submetas e operadores que permitem realizar a tarefa.

Regras de seleção são requeridas quando há mais de um método disponível para a realização da mesma meta. Seleção refere-se à estrutura de controle usada no processo, em geral regras *se-então*. São as regras pessoais que o usuário escolhe para decidir que método usar.

Juntos, os componentes do modelo GOMS descrevem o conhecimento procedimental que um usuário requer para realização de determinada tarefa no computador. Mostraremos sucintamente o modelo GOMS aplicado à tarefa de edição de um texto marcado, ilustrado pela Figura 2.25.

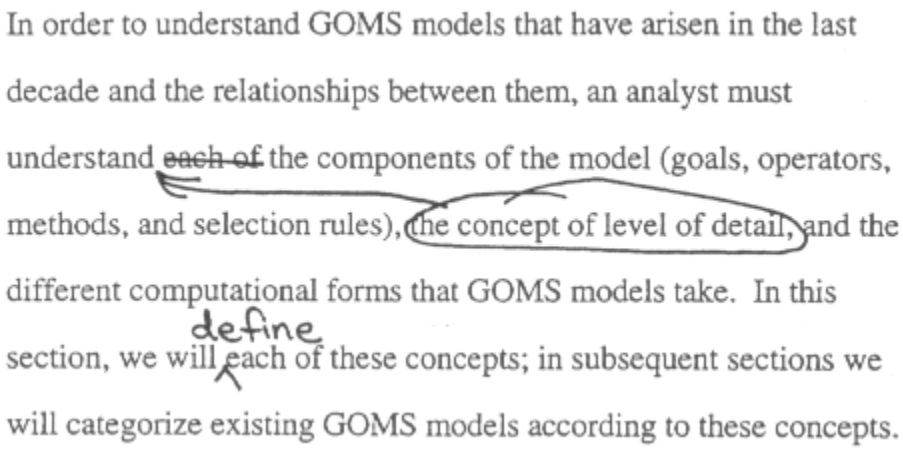

In order to understand GOMS models that have arisen in the last decade and the relationships between them, an analyst must understand ~~each of~~ the components of the model (goals, operators, methods, and selection rules), the concept of level of detail, and the different computational forms that GOMS models take. In this section, we will define each of these concepts; in subsequent sections we will categorize existing GOMS models according to these concepts.

FIGURA 2.25. - TEXTO MARCADO PARA EDIÇÃO (JOHN E KIERAS 1996B, P. 322)

Quando o usuário começa a edição tem uma meta geral: *editar-manuscrito*. Ele, então, segmenta a tarefa em unidades de tarefa, como por exemplo: *mover-texto*, *deletar-frase*, *inserir-palavra*, etc., como ilustrado a seguir:

Goal: editar-manuscrito

- *Goal: editar unidade de tarefa {repetir até que não haja mais unidades de tarefas}*
  - *Goal: adquirir unidade de tarefa {se tarefa não é lembrada}*
  - ...
  - *Goal: executar unidade de tarefa {se a unidade tarefa foi encontrada}*
    - *Goal: modificar-texto*
      [Select: Goal: mover-texto
      Goal: deletar-frase
      Goal: inserir-palavra]
      Verificar-edição

Em resumo, o modelo GOMS possui uma estrutura de pilha onde metas e sub-metas são "empilhadas" e "desempilhadas" quando completamente realizadas ou abandonadas. Um operador é uma ação realizada a serviço de uma meta. Operadores podem ser atos cognitivos, motores, perceptuais ou uma composição destes.

Operadores podem mudar um estado mental interno do usuário ou um estado do ambiente externo. Tempo de execução é um parâmetro importante dos operadores. Assim, a interação com o mundo físico aparece definida por um efeito específico e por uma duração específica. No exemplo da Figura 2.25, são operadores: *mover o mouse, clicar o botão do mouse, shift-clicar no botão do mouse* e *pressionar a tecla delete*. Operadores definem a granularidade da análise. Englobam uma mistura de mecanismos psicológicos básicos e comportamento organizado aprendido. Quanto mais fina a granularidade da análise, mais os operadores refletem os mecanismos psicológicos básicos.

Métodos são procedimentos já aprendidos; não são planos criados durante a realização da tarefa. Constituem a expressão da familiaridade e habilidade do usuário. Refletem a estrutura detalhada da tarefa no ambiente e o conhecimento da seqüência exata de passos requeridos pela ferramenta para a realização da tarefa. No exemplo citado, um método para a meta *deletar-frase* seria*: mover mouse para o inicio da frase, pressionar botão do mouse, mover mouse para o final da frase, soltar botão do mouse, pressionar tecla Del* (método marca e deleta).

A estrutura de controle no GOMS é a seleção. A essência do comportamento habilidoso pressupõe que as seleções acontecem suavemente, sem a problemática da busca que caracteriza comportamento de resolução de problemas. No exemplo o usuário poderia ter selecionado como método o *posicionar o mouse no início da frase e pressionar o delete tantas vezes quantas for o número de caracteres da frase a deletar* (método deleta caracteres). Seleção de métodos pelo usuário pode se dar pela experiência na tarefa ou por treinamento. O usuário poderia ter uma regra para o *deletar-frase* como a seguinte: se a frase tem mais de oito caracteres, usar método *marca e deleta*; caso contrário, usar método *deleta caracteres*.

Associando-se tempo a cada operador, tal modelo fornecerá previsão de tempo total para realização da tarefa. O modelo não é apropriado se erros ocorrem, uma vez que a detecção e correção de erros são rotineiras em comportamento habilidoso.

A Figura 2.26, extraída de John e Kieras (1996, p. 330), mostra um exemplo da expansão da meta *mover-texto*, para o exemplo citado anteriormente. O texto é movido através do uso do *cut* e *paste*. Para tal, o texto é primeiramente selecionado e então o *cut* é acionado. A seleção do texto pode ser feita de duas maneiras dependendo do tamanho do texto a ser selecionado. A meta *paste* requer posicionar o cursor no ponto de inserção e então acionar o paste.

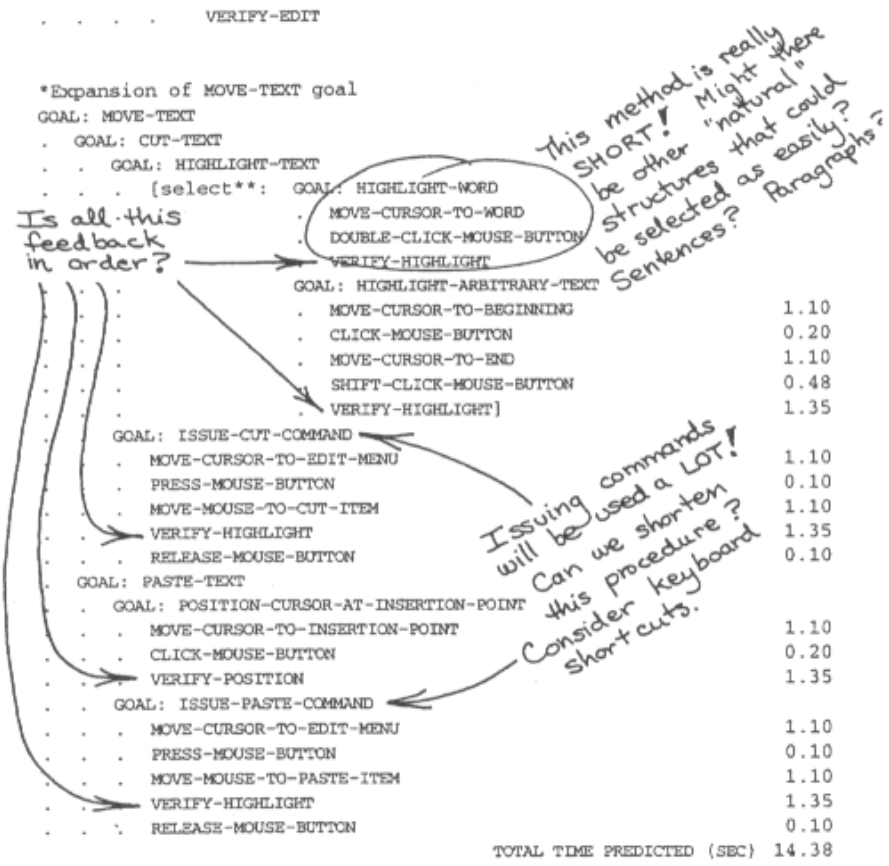


FIGURA 2.26 - ANÁLISE GOMS DA META *MOVER-TEXTO* (JOHN E KIERAS 1996B, P. 330)

Quantitativamente o método possibilita uma previsão da seqüência de operadores utilizados e tempo de execução requerido. Qualitativamente focaliza a atenção nos métodos necessários para a realização das metas. A Figura 2.26 mostra também, a título de ilustração, resultados apontados por um analista. É possível verificar, por exemplo, métodos similares, métodos muito grandes ou muito pequenos, pontos de feedback ao usuário, etc.

Atualmente existe na literatura uma família de modelos GOMS, definindo níveis de granularidade para a análise. Além da versão original, mais três variantes existem: o KLM (Keystroke Level Model), o NGOMSL (Natural GOMS Language) e o CPM GOMS (Cognitive Perceptual Motor GOMS).

KLM é uma simplificação do modelo original onde o analista lista a seqüência de operadores e o tempo de execução deles. São definidos 6 operadores diferentes: K para o pressionar de tecla ou botão, P para apontar um alvo na tela com o mouse, H para posicionar a mão sobre o periférico, D para desenhar segmento de reta, M para preparar mentalmente para uma ação e R para esperar tempo de resposta do sistema. Quantitativamente, o modelo fornece a previsão de tempo de execução para a tarefa. Qualitativamente o analista pode usar o modelo para identificar procedimentos recorrentes que poderiam ser combinados ou encurtados, por exemplo.

NGOMSL é uma notação em linguagem natural estruturada para representar modelos GOMS. O analista constrói o modelo realizando expansão *top-down* e em largura de metas, em métodos, até que os métodos contenham apenas operadores primitivos. Dessa forma a estrutura de metas fica explicitamente representada. Além de obter a seqüência de operadores e seu tempo de execução, o modelo também representa explicitamente o uso da MCD e MLD, estimando quanto de carga cognitiva o design exige e o tempo de aprendizado envolvido.

CPM GOMS não se restringe à seriação dos operadores, mas permite modelar algum grau de paralelismo na atividade do usuário. A análise chega ao nível em que operadores primitivos são atos perceptuais, cognitivos e motores, podendo ocorrer paralelismo de alguns. Usa diagramas PERT para representar operadores e suas dependências. A seqüência de operadores que produz o caminho mais longo é chamada de "caminho crítico". A soma dos operadores nesse caminho estima o tempo total da tarefa.

O modelo GOMS e seus derivados estão relacionados à abordagem geral de análise de tarefas. Como tal, enfatizam os procedimentos que o usuário deve aprender e realizar para usar bem o sistema. A descrição desses procedimentos possibilita quantificar o tempo a ser gasto na realização da tarefa e no seu aprendizado. Além disso, tal descrição possibilita antever implicações de escolhas de design. Ao contrário de serem radicalmente diferentes, os modelos GOMS ocupam diferentes espaços na análise de diferentes arquiteturas de sistemas computacionais, produzindo previsões quantitativas e qualitativas para como as pessoas usarão determinado sistema. Uma discussão bastante abrangente dos modelos GOMS e de sua aplicação em design e avaliação de interfaces pode ser encontrada em John e Kieras (1996 a, b).

A comparação entre alternativas de design é o uso mais óbvio para técnica baseada em GOMS. Documentação do sistema e *help online* também podem beneficiar-se diretamente de descrições do modelo GOMS. Kieras (1997) mostra, usando NGOMS, o resultado da comparação do PC-DOS com o Macintosh, para a tarefa de mover e deletar arquivos e diretórios. Os sistemas são comparados quanto ao número de métodos requeridos para o conjunto de metas, comprimento dos métodos e tipo de operações realizadas (perceptual, cognitiva, motora). Como resultado da análise o autor mostra que uma vantagem relevante do Mac é o tempo de

treinamento requerido comparado com linguagens de interfaces de comando. Enquanto o PC-DOS usa 12 métodos, com um comprimento total de 72 linhas, o Mac usa 3 métodos generalizados, com comprimento total de 15 linhas. O uso de poucos métodos generalizados para cobrir as metas mostra a consistência e o fator de facilidade de uso da interface do Mac.

Esse resultado mostra também a essência da manipulação direta. Enquanto o Mac usa essencialmente operadores perceptual-motores (localizar ícone é busca visual, mover cursor é movimento guiado visualmente), o PC-DOS usa operadores que demandam maior esforço cognitivo do usuário: recuperar informação da MLD para nomes de comandos, mantê-la na MCD; além disso, a estrutura dos métodos para entrar e executar comandos é complexa. Embora, a grosso modo, essa comparação seja justa, há esforço cognitivo envolvido também no "localizar ícone", pois, como veremos no próximo capítulo nem tudo que chamamos de ícone realmente o é; sinais gráficos codificados simbolicamente exigem esforço cognitivo de interpretação do usuário.

Enquanto os modelos GOMS e MPIH são aproximações razoáveis para modelar aspectos quantificáveis da interação com computadores, muito mais entendimento é necessário para captar os processos de pensamento humano subjacentes à interação com sistemas computacionais, conforme veremos na próxima seção.

MODELOS MENTAIS

"If the organism carries a "small-scale model" of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to emergencies which face it." (Craik, 1943, p.57, apud Preece *et al.* 1994)

O que é um modelo mental? Modelos mentais (MM) são explicados pela Psicologia Cognitiva com respeito a sua estrutura e função no raciocínio humano e no entendimento de linguagem. São representações analógicas, ou combinações de representações analógicas e proposicionais; são relacionados a imagens, embora diferentes destas. Ingenuamente poderia ser feita a seguinte analogia: enquanto uma imagem é uma tomada (um quadro) num filme, um modelo mental seria um pedaço desse filme. Staggers e Norcio (1993) acreditam que objetos nos modelos mentais são relacionados a entidades perceptuais. Norman (1983) considera o MM como uma representação dinâmica sobre qualquer sistema ou objeto, que evolui naturalmente na mente de um sujeito.

Modelos mentais são acionados quando nos é requerido fazer inferências ou previsões a respeito de determinado assunto. Tente responder *quantas janelas tem sua casa* e observe o processo que você usa para responder. Não parece provável que tenhamos um conhecimento específico armazenado, de quantas janelas tem nossa casa. O que normalmente fazemos é uma "execução" de um modelo mental, imaginando-nos em cada cômodo da casa, ou percorrendo a casa pelo lado de fora, contando as janelas...

Interagindo com o ambiente, com outros e com artefatos tecnológicos, as pessoas formam modelos mentais delas próprias e das coisas com as quais estão interagindo. As concepções espontâneas de fenômenos físicos, são também modelos mentais que as pessoas usam para explicar fenômenos da natureza. O entendimento das pessoas sobre os artefatos com os quais interagem é fraco, especificado imprecisamente e cheio de inconsistências, "buracos" e artimanhas idiossincráticas. Modelos mentais são incompletos. A habilidade das pessoas para "executar" seus modelos mentais é limitada pelos mecanismos perceptual e cognitivo. Modelos mentais são instáveis pelas próprias restrições e interferências da memória: as pessoas esquecem detalhes do sistema que estão usando, artefatos e operações similares são confundidos. Modelos mentais não são "científicos": as pessoas mantêm comportamento supersticioso em seus modelos e freqüentemente fazem operações físicas extras em vez de planejamento mental que possibilite evitar essas ações (Norman, 1983). Chandra e Blockley (1995) argumentam que mesmo nas raízes das teorias científicas e sistemas axiomáticos, há uma camada que é primitiva no sentido de que não é explicada ou justificada explicitamente dentro dos sistemas.

Usuários interagindo com artefatos tecnológicos, desenvolvem dois tipos principais de modelos mentais: estrutural e/ou funcional. Se pensarmos, por exemplo, num mapa de metrô (o de Londres é um exemplo típico), o mapa provê uma estrutura que passageiros regulares aprendem a internalizar e usam para responder "*como ir de A para B*". Ao mesmo tempo, há conhecimento necessário para "*como usar o sistema*": o passageiro deve comprar o bilhete certo, no local certo, deve saber o que fazer com o bilhete, como entrar e sair dos trens, etc. para isso há a necessidade de um modelo funcional do sistema.

No modelo mental estrutural (MME) é assumido que o usuário internalizou, na memória, a estrutura de como o artefato funciona. MMEs são usados para descrever a mecânica interna de uma máquina ou sistema em termos de suas partes componentes. O MME atua como substituto da coisa real. Ao explicar como a máquina ou sistema funciona o usuário tem a possibilidade de prever os efeitos de seqüências de ações. Tais modelos são extremamente úteis quando a máquina "quebra" ou ocorre um erro na interação com o sistema. *Será que as pessoas, de um modo geral, desenvolvem naturalmente esse tipo de modelo?* Podemos pensar no carro que usamos todo dia, na TV, no telefone, etc.

No modelo mental funcional (MMF) o usuário internaliza conhecimento procedimental sobre como usar a máquina ou sistema. Nesse modelo as pessoas em vez de desenvolverem o "manual na cabeça" simplesmente desenvolvem um modelo de "como fazer". O MMF se desenvolve a partir de conhecimento anterior de um domínio similar; parece haver um mapeamento tarefa-ação.

Como modelar o MM de determinada pessoa sobre determinado sistema?
Considere *t* um sistema-alvo, *C(t)* o modelo conceitual de *t*, *M(t)* o modelo mental do usuário sobre o sistema *t* e *C(M(t))* nossa conceituação sobre um modelo mental. Sujeitos são convidados a "pensar alto" enquanto interagem com o sistema *t*. São feitos registros da interação (observação, vídeos, gravações, etc.). Os protocolos gerados da descrição do sujeito para suas ações e atividades fornecem pistas do modelo mental desse sujeito para o sistema ou artefato com o qual está interagindo. A literatura acadêmica tem mostrado que a atividade de programar requer acesso a algum tipo de MM (Mayer, 1981; Du Boulay e O'Shea, 1981; Cañas *et al.* 1994). Exemplos bastante interessantes de MM de sujeitos novatos aprendendo a programar ilustram de forma simples o conceito de modelo mental.

Rocha (1991) analisa as dificuldades no aprendizado de programação e propõe representações computacionais auxiliares a esse processo. A partir da análise dos modelos mentais que as pessoas constroem sobre como o código de um programa controla as operações do computador propõe um modelo conceitual cujas componentes são as mais relevantes na formação de um correto modelo mental do funcionamento de um computador. Define que a natureza dos modelos mentais de programação é o de modelo substituto: como um aeromodelo, que permite simular o sistema objetivo e pode ser usado para responder perguntas sobre seu funcionamento. A característica básica é o fato de poderem ser executados.

Resultados de observação de novatos aprendendo a programar em Prolog mostram que, em geral, os novatos usam um MM inicial do sistema que é baseado no discurso da língua natural e em formalismos de outras linguagens de programação. Os exemplos a seguir, mostram concepções de novatos aprendendo a programar em Prolog, como um reflexo de seus modelos mentais do sistema.

> *Através da lógica o computador responde perguntas, como se ele estivesse pensando (impossível), mas parece...* (Baranauskas 1995, p. 100)

> *... considere o problema de procurar o n-ésimo elemento de uma lista, e a solução apresentada:*
> *Elemento([X],1,X).*
> *Elemento([X|Y],N,Z):- preciso escrever algo para "caminhar" na lista, só que "aqui" eu não sei como "caminhar" na lista. Como "eu pego" um elemento nessa lista?* (Baranauskas 1995, p. 103)

O mesmo trabalho anterior mostra, ainda, como os *M(t)*s evoluem em direção ao *C(t)*, à medida que o processo de aprendizagem da linguagem evolui. O uso de um M(t) baseado no paradigma procedimental é um dos grandes responsáveis pela dificuldade de novatos no aprendizado de linguagens baseadas no paradigma da lógica e no paradigma funcional (Baranauskas, 1991). Suspeitamos que o mesmo resultado também explique a dificuldade de novatos com o paradigma orientado a objetos.

A observação de MMs, deve considerar fatores externos que influenciam *M(t)* e *C(M(t))*. Norman (1983) cita, por exemplo, o sistema de crenças: o M(t) de uma pessoa reflete seu sistema de crenças sobre o sistema físico *t*. Esse sistema de crenças é construído pelo sujeito através de observação, instrução ou inferência. Também, deve haver uma correspondência entre os parâmetros e estados do MM e aspectos e estados do sistema físico que o sujeito pode observar. O objetivo do MM é permitir à pessoa entender e antecipar o comportamento de um sistema físico podendo executar "mentalmente" o seu modelo.

Porque Modelos Mentais?
As Ciências Cognitivas podem nos ajudar a entender as estruturas incompletas, indistintas e confusas que as pessoas têm a respeito dos artefatos tecnológicos. Como designers é nossa obrigação desenvolver sistemas para o usuário final que o ajude a construir modelos mentais adequados à sua interação com o sistema. Assim, conceituar o conhecimento do usuário em termos de modelos mentais pode ajudar o designer a desenvolver interfaces apropriadas.

A operação de qualquer artefato seja ele uma garrafa de cerveja a ser aberta, uma planta nuclear ou um sistema computacional, será mais simples se tiver um bom modelo conceitual. É tarefa do designer, com base nos princípios apresentados, construir um modelo conceitual para o artefato, adequado ao uso. Norman (1986) distingue 3 tipos de modelos associados ao artefato: o modelo do designer, o modelo do usuário e a imagem do sistema (Figura 2.27). Os modelos do designer e do usuário são modelos mentais. Conforme discutido anteriormente, as pessoas formam modelos mentais de si próprias, das coisas e das pessoas com as quais interagem. Esses modelos provêem poder de predição e explicação, necessários para condução da interação.

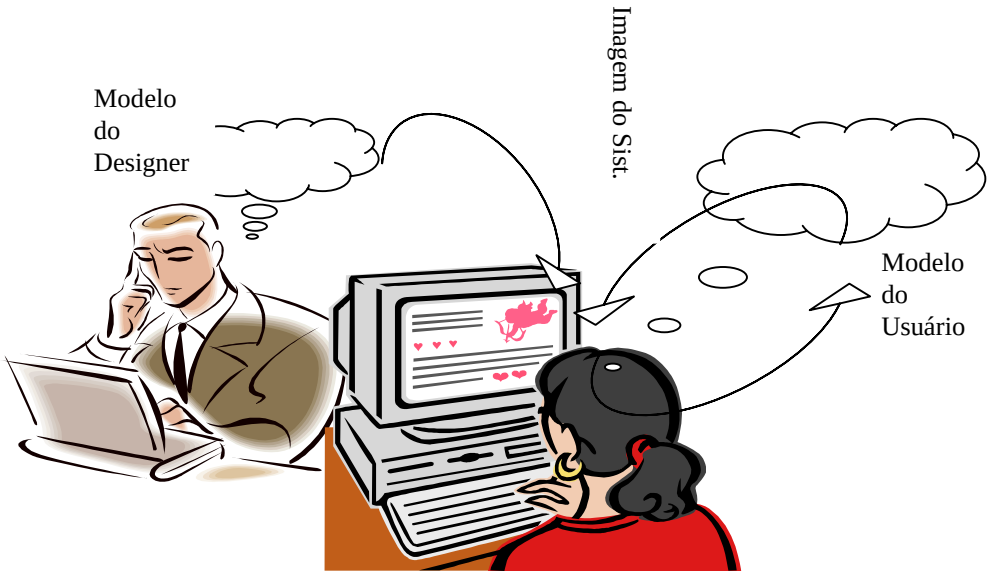


FIGURA 2.27 - TRÊS ASPECTOS DE MODELOS MENTAIS

O Modelo do Designer é a conceituação que o designer tem em mente sobre o sistema. O Modelo do Usuário é o que o usuário desenvolve para entender e explicar a operação do sistema. A aparência física, sua operação e a forma como responde, somados ao *help online* de manuais de instrução formam a "Imagem do Sistema". O designer deve assegurar que a imagem do sistema seja consistente com seu modelo conceitual, uma vez que é através da imagem do sistema que o usuário forma seu modelo mental. Idealmente, ambos Modelo do Designer e Modelo do Usuário deveriam coincidir.

Em ambientes de programação já temos visto esforços, no sentido de tornar visíveis ao usuário, certas operações do sistema. Gentner e Stevens (1983) postulam que os modelos mentais são formados pela estruturação de analogias e metáforas, especialmente em domínios não familiares. Como já introduzido no capítulo anterior, as metáforas desempenham um papel importantíssimo no processo de facilitar ao usuário a construção de um modelo mental adequado à interação com o sistema. No Capítulo 3 voltaremos ao assunto abordando métodos sistemáticos que incorporam metáforas ao design de interfaces.

REFERÊNCIAS:

Baranauskas, M.C.C. (1991) Procedure, Function, Object or Logic?, *Proceedings of the Eighth International Conference on Technology and Education*, Toronto, Canadá, p. 730-731.

Baranauskas, M.C.C. (1995) Observational Studies about Novices Interacting in a Prolog Environment based on Tools, *Instructional Science an International Journal of Learning and Cognition*, 23(1-3): 89-109.

Cañas , J.J., Bajo, M.T. Gonzalvo, P. (1994) Mental Models and Computer programming. *Int. J. Human-Computer Studies*, 40,795-811.

Card, S. K. ,Moran, T.P. e Newell A. (1983) *A Psicologia da Interação Humano-Computador*, Hillsdale, NJ: Laurence Erlbaum Ass..

Carrol, J.M. e Olson, J.M., (1988a) Mental Models. Em *Handbook of Human-Computer Interaction*, M. Helander (ed), Amsterdam:North Holland.

Carrol, J.M. e Olson, J.M.,(1988b) Interface Metaphors and User Interface Design. Em *Handbook of Human-Computer Interaction*, M. Helander, (ed), Amsterdam:North Holland.

Chandra, S. Blockley, D.I. Cognitive and Computer Models of Physical Systems. *Int. J. Human-Computer Studies*, 43,539-559.

Du Boulay, B., O'Shea, T. (1981). Teaching Novices Programming. Em *Computing Skills and the User Interface*, M. Coombs e J. Alty (eds). London: Academic Press.

Fame (1999) *Hall of Fame*. http://www.iarchitect.com/mfame.htm. Consulta em 4/4/2000.

Gentner, D. Stevens, A.L. (1983) *Mental Models*. Hillsdale, NJ: Lawrence Erlbaum

Hutchins, E.(1990) the Technology of Team Navigation . Em *Intelectual Teamwork* J. Galegher, R.E. Kraut e C. Edigo (eds) Hillsdale, NJ: Lawrence Erlbaum Ass..

John, B.E., Kieras, D.E. (1996a) Using GOMS for User Interface Design and Evaluation: Which Technique? *ACM Transactions on Computer-Human Interaction*, 3(4):287-319.

John, B.E., Kieras, D.E. (1996b) The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320-351.

Johnson-Laird P.N.,(1989) Mental Models. *Em Foundations of Cognitive Science* M.I. Posner (ed), Cambridge, MA: MIT Press.

Kieras, D. (1997) A Guide to GOMS Model Usability Evaluation using NGOMSL. Em *Handbook of Human-Computer Interaction*, M. Helander, T.K.Landauer, P. Prabhu (eds) Elsevier Science, p. 733-766.

Laurel, B. (1990) *The Art of Human-Computer Interface Design*. Reading, Mass.:Addison-Wesley.

Lakoff, G. e Johnson, M.(1980) *Metaphors we Live By*. Chicago:University of Chicago Press.

Lindsay, P.H. e Norman, D.A.(1972) *Human Information Processing An Introduction to Psychology*, Academic Press Inc..

Mayer, R.E. (1981) The Psychology of how Novices Learn Computer Programming. *Computing Surveys*, 13,121-141

Norman, D.A. (1983) Some Observations on Mental Models. Em *Mental Models*, D. Gentner e A.L. Stevens (eds), Hillsdale, NJ: Lawrence Erlbaum.

Norman, D. A. (1993) *Things that make us Smart*, Addison Wesley Publishing Company.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T.(1994) *Human-Computer Interaction*. Reading, Mass.: Addison-Wesley.

Rocha, H. (1991) *Representações Computacionais Auxiliares ao Processo de Ensino/Aprendizagem de Conceitos de Programação*, Tese de Doutorado, FEEC, Unicamp.

Sacks, O (1995) *Um Antropólogo em Marte*, SP:Companhia das Letras.

Shame (1999) *Hall of Shame*. http://www.iarchitect.com/mshame.htm. Consulta em 4/4/2000.

Staggers, N., Norcio, A.F. (1993) Mental Models: Concepts for Human-Computer Interaction Research. *International Journal of Man-Machine Studies*, 38,587-605.

Suchman, L. (1999) From Interactions to Integrations, *Atas do II Workshop sobre Fatores Humanos em Sistemas Computacionais, IHC'99* Campinas, SP p. 3.

CAPÍTULO 3

PARADIGMAS DA COMUNICAÇÃO HUMANO-COMPUTADOR E O DESIGN DE INTERFACES

Engineering, medicine, business, architecture, and painting are concerned not with how things are but how things might be – in short, with design
Simon (1982, pxi)

INTRODUÇÃO

O processo de design em IHC tem sido naturalmente centrado no usuário e têm incorporado questões relativas a modelos cognitivos do processamento humano. Neste capítulo discutiremos a natureza da interação com computadores, com base nos fundamentos apresentados no capítulo anterior. Discutiremos o processo de design de várias perspectivas: da Engenharia Cognitiva à Engenharia Semiótica. O capítulo também explorará a influência de aspectos sociais e organizacionais do contexto do usuário e de sua tarefa no processo de design.

Existe uma grande influência de métodos da engenharia, em particular de Engenharia de Software em IHC, através do design e desenvolvimento de software. Por outro lado, teorias de design originais de contextos diversos, em particular o design industrial, arquitetônico e gráfico também têm influenciado a maneira como design de interfaces tem sido feito recentemente. Os modelos do processo de design em IHC envolvem desde uma discussão crítica dos ciclos de vida clássicos para o desenvolvimento de software, originais da Engenharia de Software, até modelos mais específicos do ciclo de design, como, por exemplo, o Modelo Estrela de Hix e Hartson (1993). Esses modelos apresentados também em Baecker *et al.* (1995), em geral envolvem além do desenvolvimento do sistema propriamente dito, um contexto mais amplo que inclui também questões de natureza social e de organização do trabalho.

Iniciamos examinando as bases do design em IHC, através da apresentação e discussão do paradigma que tem fundamentado toda a área. Serão apresentados e discutidos os princípios do Design Centrado no Usuário e a Teoria da Ação, com base no trabalho seminal de Norman e Draper (1986). Passaremos, então, a algumas extensões e variações desse modelo. Será discutido também o Design Participativo (Schuler e Namioka, 1993) como ilustração das abordagens que tentam assegurar a participação efetiva do usuário e seu contexto social no design. Essa abordagem, onde o usuário é parte ativa no processo de design e não simplesmente parte do processo final de avaliação do software, tem sido considerada principalmente para o design de sistemas colaborativos e de trabalho em grupo.

Como um dos suportes ao processo de design, será discutida a Engenharia de Usabilidade (Nielsen, 1993), que apresenta uma abordagem ao design de sistemas onde níveis de usabilidade são especificados *a priori*, e o sistema é construído com objetivos de alcançar essas medidas, conhecidas como métricas. Os métodos da Engenharia de Usabilidade têm sido bem aceitos pela indústria de software por proverem uma sistemática para teste de um produto durante desenvolvimento. Também popular na indústria de software é o Design baseado em *guidelines*, que parte do princípio de que um bom design resulta parcialmente de conhecimento e experiência do designer e da maneira como ele aplica esse conhecimento. O uso de *guidelines*, princípios e regras como suporte ao design será também discutido, segundo suas bases da teoria cognitiva.

Finalmente o capítulo apontará para propostas mais recentes de linguagens de design para interfaces e teorias fundamentadas no paradigma da comunicação, como é o caso da Semiótica Computacional (Andersen *et al*., 1993, Andersen, 1997) e da Engenharia Semiótica (Souza, 1993). Estas últimas introduzem novas concepções para interface, e novos formalismos para análise, design e avaliação de software que instrumentam o designer segundo bases semióticas.

ENGENHARIA COGNITIVA

> ***Method helps intuition if it is not transformed into dictatorship. Intuition augments method if it does not instill anarchy. In every moment of our semiotic existence, method and intuition complement each other.***
> (Nadin, 1988, p.98)

Como vimos nos capítulos anteriores, nossa mente tenta fazer sentido das coisas que vemos ao nosso redor. Os objetos que fazem parte de nosso dia-a-dia são ótimos exemplos para pensarmos nesse processo. Freqüentemente encontramos novos objetos (ou novas apresentações para antigos objetos) no dia-a-dia, enquanto estamos fazendo alguma outra coisa, realizando alguma tarefa. Somos distraídos da tarefa que estamos realizando por alguma coisa que deveria ser simples, e não causar esforço. A maneira como lidamos com essas situações é explicada, em parte pela psicologia dos fatores humanos, da cognição e do pensamento como visto no Capítulo 2. A informação expressa na aparência dos objetos, conforme já discutido anteriormente, de certa forma dirige nosso processo de interpretação e operação sobre esse objeto. A facilidade ou dificuldade com que operamos no mundo dos objetos é, portanto, devida à habilidade do designer em tornar clara a operação sobre o objeto, projetando uma boa imagem da operação e considerando outros elementos do universo de conhecimento do usuário. A Semiótica, conforme veremos mais adiante ainda neste capítulo, também explica essa relação entre o objeto, seu representante e o processo de interpretação, no plano dos signos que encontramos ao nosso redor.

Engenharia Cognitiva é um termo cunhado por Norman e Draper (1986), um tipo de "Ciência Cognitiva Aplicada", conforme ele próprio categoriza, que tenta aplicar o que é conhecido da ciência ao design e construção de máquinas. Os conceitos da Engenharia Cognitiva formam as bases do paradigma dominante atualmente na área de IHC. Entre suas metas principais estão: entender os princípios fundamentais da ação humana que são relevantes à engenharia do design, indo além dos aspectos ergonômicos; criar sistemas "agradáveis de usar", que possibilitem ao usuário um "engajamento prazeroso" na terminologia de Laurel (1990), indo além dos aspectos de facilidade de uso.

A Engenharia Cognitiva considera dois lados na interface: a do próprio sistema e a do usuário. A realização de tarefas complexas por não expertos é considerada uma atividade de resolução de problemas cujo processo é facilitado quando a pessoa possui um bom modelo conceitual do sistema físico.

A complexidade da tarefa é devida às naturezas diferentes das variáveis envolvidas. A pessoa possui metas e intenções – variáveis psicológicas – que se relacionam diretamente às necessidades da pessoa. A tarefa deve ser realizada em um sistema físico, com mecanismos físicos a serem manipulados, resultando em mudanças nas variáveis físicas e no estado do sistema. A pessoa interpreta as variáveis físicas em termos relevantes às suas metas psicológicas e traduz as intenções psicológicas em ações físicas sobre os mecanismos do sistema. Isso significa que há um estágio de interpretação que relaciona variáveis físicas e psicológicas, assim como funções que relacionam a manipulação das variáveis físicas às mudanças no estado físico do sistema.

Mesmo tarefas simples envolvem um número grande de aspectos a considerar. Consideremos, por exemplo, a tarefa de um marinheiro novato tentando manobrar um barco à leme. Há um único mecanismo de controle (o leme) que afeta uma única variável (a direção do barco). A complexidade é em parte devida à discrepância entre o modelo mental do marinheiro e o modelo conceitual do sistema. O mapeamento entre o movimento do leme e a direção do barco em geral é oposta ao que os novatos esperam.

Uma intenção é criada para satisfazer uma meta. No caso do barco à leme, a meta é alcançar um certo alvo. A diferença entre a meta desejada e o estado atual do sistema dá origem a uma intenção – em termos psicológicos – que é traduzida em uma seqüência de ações: a especificação de que ações físicas serão realizadas nos mecanismos do sistema. Portanto, o caminho da intenção para a especificação de ações requer a consideração do mapeamento entre os mecanismos físicos e o estado do sistema e entre o estado do sistema e a interpretação psicológica resultante.

Norman e Draper (1986) propõe “uma teoria da ação” para entender “como as pessoas fazem as coisas”, distinguindo entre diferentes estágios de atividades. Na Teoria da Ação são diferenciados sete estágios da atividade do usuário, conforme ilustra a Figura 3.1. Muitos sistemas computacionais podem ser categorizados por quão bem suportam os diferentes estágios. Imagine, por exemplo, uma pessoa interagindo com um sistema computacional. As metas da pessoa são expressas em termos relevantes à pessoa (psicológicos). Os mecanismos e estados do sistema são expressos em termos relativos a ele (físicos). A discrepância entre variáveis físicas e psicológicas cria os pontos a serem considerados no design, análise e uso de sistemas, que Norman chamou de “golfos da execução e da avaliação”.

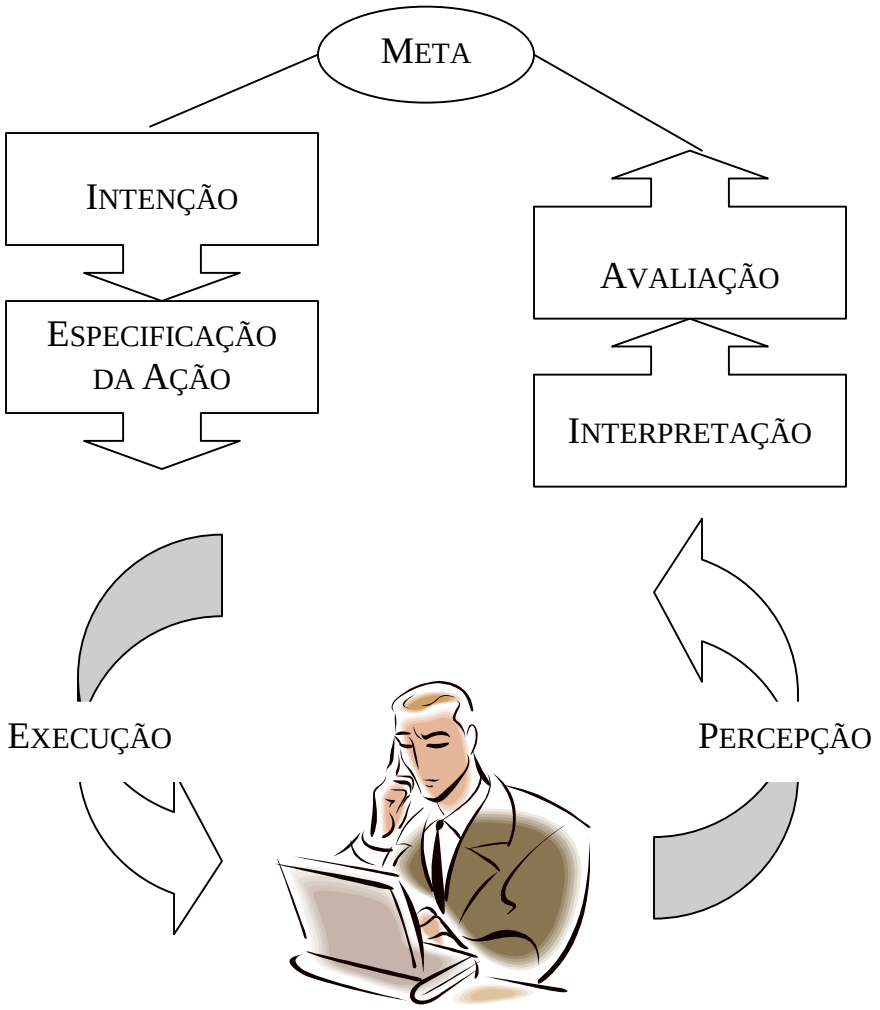


FIGURA 3.1 - OS SETE ESTÁGIOS DA TEORIA DA AÇÃO DE NORMAN

O **Golfo da Execução** envolve as atividades de formação da intenção, especificação da seqüência de ações – o que não é trivial – e execução da ação através do contato com mecanismos de entrada da interface. O **Golfo da Avaliação** requer comparar a interpretação do estado do sistema com as metas e intenções originais. Começa com a apresentação de saída da interface, a partir da qual o usuário passa pela atividade de processamento perceptual da saída, interpretação e avaliação (comparação da interpretação do estado do sistema com a intenção e metas originais).

Possibilitar que o usuário atravesse os golfos significa construir uma interface que se ajuste às necessidades do usuário, de forma que possa ser prontamente interpretada e manipulada. O usuário também pode, ele próprio, ao custo de seus esforços,

atravessar os golfos, criando planos, seqüências de ações e interpretações que movem a descrição de metas e intenções para mais próximo da descrição requerida pelo sistema físico (e não pela tarefa original).

Menus são exemplos de suporte aos estágios de execução e especificação de ações. A presença visual pode ajudar em vários estágios da atividade: como suporte à geração de intenções – lembrando o usuário do que é possível; como suporte à seleção de ação: itens visíveis atuam como tradução direta para ações possíveis; como suporte à execução se associado a dispositivo de apontamento; como suporte à avaliação: lembrando visualmente o que foi feito; como suporte à interpretação através do uso de determinadas representações.

Resumindo, a Engenharia Cognitiva conceitua interface pelos seus "dois lados": o do sistema e o do ser humano. Estágios de execução e percepção (humanos) mediam entre representações físicas (do sistema) e psicológicas (do ser humano). Mecanismos de entrada/saída (do sistema) mediam entre representações psicológicas e físicas. Mudamos a interface, pelo lado do sistema, através de design apropriado. Muda-se a interface pelo lado humano, através de aprendizado e experiência. Na situação ideal, nenhum esforço psicológico deveria ser requerido para se atravessar os golfos.

Design de interface no paradigma da Engenharia Cognitiva, portanto, relaciona três tipos de conhecimento: de design, programação e tecnologia; de pessoas, princípios do funcionamento mental, comunicação e interação e conhecimento da tarefa. Somente o módulo da interface deve estar em comunicação com o usuário: do ponto de vista do usuário a interface "é" o sistema, conforme veremos na próxima seção.

MANIPULAÇÃO DIRETA

> ***Beware the Turing tar-pit in which everything is possible but nothing of interest is easy*** (Perlis, A., apud Hutchins *et al.* 1986)

O que exatamente é a manipulação direta?

Imagine-se dirigindo um carro em que, em vez de direção, pedais e câmbio tem apenas um teclado...e R20:E:A35 seria usado para "reduza para 20km/h, vire para a esquerda, acelere até 35 km/h"

Shneiderman (1983) foi quem usou o termo pela primeira vez para se referir a uma classe emergente de sistemas bastante atraentes na década de oitenta, como as primeiras planilhas eletrônicas, editores de texto, sistemas CAD, videogames, etc. Esses sistemas possuíam interfaces gráficas que permitiam operá-los "diretamente"

usando ações manuais em vez de instruções fornecidas via teclado. Tais sistemas mudaram o paradigma da interação humano-computador, do "diálogo" baseado em linguagem de comando para a "manipulação" baseada na linguagem visual (Frohlich, 1997). Em vez de um meio computacional abstrato, toda programação é feita graficamente, em uma forma que tenta casar com a maneira como pensamos no problema (Hutchins *et al.*, 1986).

Do mundo que se "comanda" passou-se para o mundo com o qual se "interage". O primeiro marco em interface de manipulação direta é o *Sketchpad*, um programa para design gráfico criado por Sutherland em 1963. Seu trabalho é um marco não apenas pela prioridade histórica, mas por antecipar a concepção da saída na tela como "folha de papel", o uso de dispositivos de apontamento e a importância de mostrar abstrações graficamente.

Nas interfaces de manipulação direta não há operações escondidas, sintaxe ou nomes de comandos para aprender. O único conhecimento requerido é no próprio domínio da tarefa. Excelentes exemplos de tais interfaces na época eram os editores WYSWYG (*what you see is what you get*). A diretividade em interfaces, entretanto, apresenta-se em vários níveis dentro de um *continuum*. Tomemos como exemplo um editor gráfico numa tarefa de criar ilustrações; a operação de mover um círculo mostra os graus de indireção que podem estar presentes na interface: A) o usuário aponta na tela o círculo e o move levando-o para a posição desejada com o dedo. B) a introdução do mouse coloca mais um grau de indireção, uma vez que para mover o círculo é necessário mover o mouse. C) com as setas de direção acrescenta-se mais um nível de indireção, uma vez que não há equivalência de movimentos. D) o uso de um comando para mover o círculo acrescenta mais um nível uma vez que a sintaxe e a semântica determinam o que acontece.

A "ilusão" da manipulação direta foi sumarizada por Shneiderman (1998) em três princípios de design:

1. Representação contínua do objeto de interesse;
2. Ações físicas (cliques, arraste, etc.) em vez de sintaxe complexa;
3. Operações incrementais reversíveis, cujo impacto no objeto de interesse é imediatamente visível.

Esses princípios são baseados na suposição de que a manipulação direta resulta em menor comprometimento de recursos cognitivos. Hutchins *et al* (1986) discutem as bases subjacentes para sistemas de manipulação direta.

Retomando a mudança paradigmática na natureza da interação humano-computador, na metáfora da conversação, a linguagem da interface é um "meio" no qual usuário e sistema têm uma conversação sobre um mundo assumido, mas não explicitamente representado. O usuário está em contato com estruturas lingüísticas que podem ser interpretadas como se referindo aos objetos de interesse.

Na metáfora do mundo-modelo, termo cunhado por Hutchins *et al* (1986) para representar o paradigma da manipulação direta, a interface é um mundo onde o usuário age. Esse mundo muda de estado em resposta às ações do usuário. Em vez de descrever as ações de interesse, o usuário realiza as ações.

Diretividade não é uma propriedade da interface isoladamente, entretanto. Dois aspectos são usados para se medir a diretividade: distância e engajamento. Distância refere-se à distância entre o pensamento de alguém e os requisitos físicos do sistema em uso; relação entre a tarefa que o usuário tem em mente e a maneira como a tarefa pode ser realizada através da interface. Engajamento é o sentimento de que se está manipulando diretamente os objetos de interesse.

O sentimento de diretividade é inversamente proporcional à quantidade de esforço cognitivo requerido para manipular e avaliar um sistema. Esse esforço cognitivo é resultado direto dos Golfos de Execução e Avaliação, conforme discutido na última seção. Assim, quanto mais as interfaces ajudarem a aproximar os Golfos, menor será o esforço cognitivo requerido e mais direto será o sentimento de interação resultante.

Independentemente da metáfora utilizada para a interface, seja ela a "conversação" ou o "mundo-modelo", há uma "linguagem" da interface. E toda expressão na linguagem da interface tem um significado e uma forma. Essa independência entre forma e significado, para efeitos de análise, permite descrever duas propriedades da linguagem da interface, que Hutchins *et al* chamaram de Distância Semântica e Distância Articulatória.

Distância semântica (DS) reflete a relação entre as intenções do usuário e o significado na linguagem da interface. **Distância articulatória** (DA) reflete a relação entre a forma física de uma expressão na linguagem de interação e seu significado.

Duas perguntas básicas resumem a medida de diretividade semântica em uma interface: *É possível dizer o que se quer dizer nessa linguagem? As coisas de interesse podem ser ditas de forma concisa?*

Com relação ao Modelo da Teoria da Ação, a diretividade semântica pode ser medida no golfo de execução observando-se quanto da estrutura requerida é fornecida pelo sistema e quanto pelo usuário. Quanto mais estrutura o usuário tiver que prover, maior será a distância que ele terá que transpor. No golfo de avaliação a diretividade semântica pode ser observada pela quantidade de estruturas requeridas para o usuário determinar se uma dada meta foi alcançada ou não.

Como, então, reduzir a distância semântica? Pelo lado do sistema o designer pode construir linguagens especializadas de mais alta ordem, fazendo a semântica da entrada/saída do sistema "casar" com os modelos mentais de entrada/saída do usuário. Um exemplo simples dessa redução no golfo de avaliação é fazer a saída do

sistema mostrar os conceitos diretamente, em lugar de deixar o usuário computá-los mentalmente. Algumas representações gráficas para tabelas numéricas e sistemas *wysiwyg* para editores de texto são exemplos inquestionáveis de redução da distância semântica. Essa estratégia representa uma mediação entre intenções e expressão na linguagem. Portanto, a linguagem da interface deve prover uma maneira poderosa e produtiva para o usuário pensar sobre o domínio. Essa estratégia de redução da distância semântica, do ponto de vista de design, pode introduzir um conflito entre generalidade do sistema e orientação a um domínio específico.

O usuário também reduz a distância semântica, independentemente do sistema, construindo estruturas mentais para atravessar os golfos, aprendendo a pensar na mesma linguagem requerida pelo sistema. Assim, as metas iniciais e intenções do usuário são formadas na mesma linguagem requerida pelo sistema. Isso explica o "sentimento de diretividade" que o uso freqüente de uma interface – que não é de manipulação direta – pode resultar. Um usuário Unix, por exemplo, pode "sentir" a ação do "rm arq" como mais direta do que o "arrastar do arq para a lixeira", uma vez que já tem a atividade de planejamento substituída pela atividade de recuperação da memória, nas ações requeridas para a tarefa. Como designers é essencial distinguir o comportamento automatizado – que pode contribuir para o sentimento de diretividade – da diretividade semântica propriamente dita. Esse fenômeno é explicado pela teoria do determinismo lingüístico de Worf, segundo a qual a maneira como pensamos em algo é moldada pelo vocabulário que temos para pensar (sobre).

Deve-se notar que observar a interface isoladamente, em geral, não é suficiente para se determinar a distância semântica de um sistema. Hutchins *et al.* (1986) citam o exemplo dos instrumentos musicais: o teclado do piano é mais direto semanticamente que as cordas do violino, para a tarefa de produzir notas. Entretanto, o violino é melhor para controlar características mais sutis do som. A diretividade semântica é uma medida da distância entre a meta e intenção do usuário e o significado da expressão disponível na interface. Uma análise da natureza da tarefa sendo realizada é essencial para se determinar a diretividade semântica da interface.

Todo item de vocabulário em toda linguagem tem uma estrutura física. Por exemplo, a palavra na linguagem natural tem uma estrutura fonética e uma estrutura tipográfica. Assim também os itens que constituem a linguagem da interface têm uma estrutura física. Na entrada são exemplos: seqüências de caracteres que são teclados em uma interface de linguagem de comandos, ou movimentos e cliques no mouse ou, ainda, uma cadeia fonética. Na saída, cadeias de caracteres, mudanças em ícones, grafos, diagramas, animação, sinais audíveis, etc. A diretividade articulatória é uma medida da distância entre a forma física da expressão dos elementos da interface e seu significado.

Há maneiras de se criar linguagens de modo que relações entre os itens de vocabulário e seus significados não sejam arbitrários. Um bom exemplo disso na linguagem natural são as onomatopéias. Na linguagem da interface, a diretividade articulatória pode ser aumentada, por exemplo, permitindo-se ao usuário a especificação de uma ação imitando-a, ou permitindo que ele use conhecimento anterior para criar a relação forma-significado. O mouse é um bom exemplo de periférico que provê diretividade articulatória para tarefas representadas espacialmente. A intenção de mover o cursor na tela, e a ação sobre o mouse são movimentos similares.

Linguagens icônicas são exemplos de representações em que a forma da expressão é relacionada a seu significado. Conforme veremos na seção dedicada à Semiótica, entretanto, o abuso de utilização de símbolos gráficos em interfaces, que nem sempre correspondem ao conceito de ícone, não garantem a diretividade articulatória completa.
A Figura 3.2, adaptada de Hutchins *et al.* (1986) resume graficamente os conceitos tratados nesta seção.

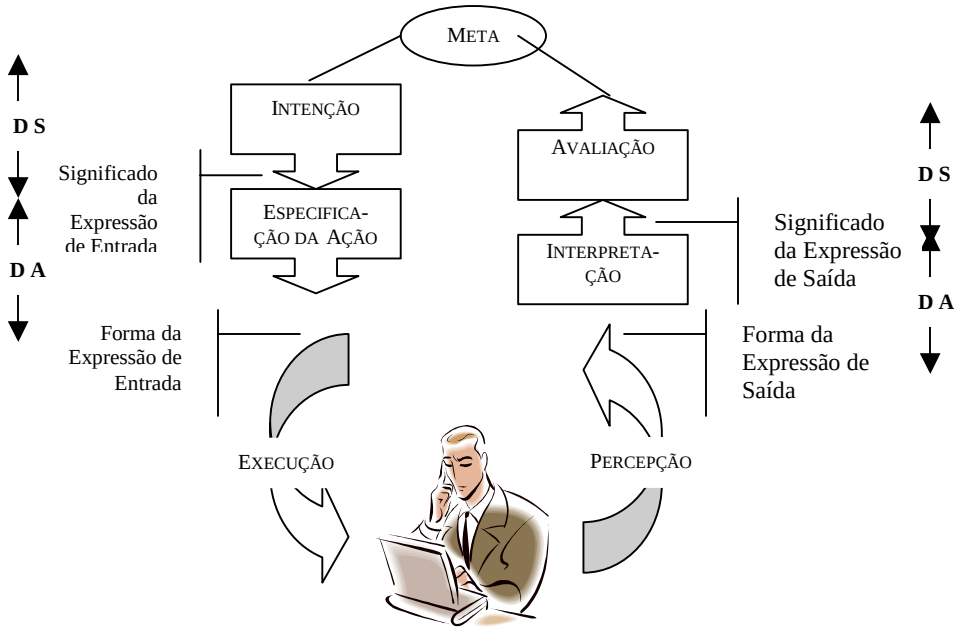


FIGURA 3.2 - DISTÂNCIAS SEMÂNTICAS E ARTICULATÓRIAS NOS GOLFOS DE EXECUÇÃO E AVALIAÇÃO

Resumidamente, para possibilitar o sentimento de engajamento direto na interface o design deve remover a percepção do computador como um intermediário,

possibilitando execução e avaliação diretas, representação contínua do estado do sistema e linguagens de entrada/saída inter-referenciais – a expressão de entrada deve incorporar ou usar a expressão de saída anterior, para criar a "ilusão" de manipulação direta dos objetos. A redução na carga cognitiva para manter mentalmente informação relevante sobre o estado do sistema e a forma de interação contribuem para o sentimento de engajamento.

Shneiderman (1983) sintetiza os benefícios da manipulação direta para a usabilidade de sistemas computacionais, nos seguintes aspectos:

- Facilidade de aprendizado do sistema.
  Novatos podem aprender rapidamente a funcionalidade básica;
- Facilidade de memorização.
  Usuários freqüentes podem reter mais facilmente conceitos operacionais;
- Performance melhorada do experto no domínio da tarefa;
- Redução de mensagens de erro;
- Aumento no controle do usuário.
  Usuários têm a ansiedade reduzida, porque o sistema é compreensível e porque as ações são facilmente reversíveis.

Ao mesmo tempo, alguns elementos são críticos no design de interfaces de manipulação direta, em especial a qualidade da representação gráfica selecionada. A representação deve ter significado preciso para o usuário, o que veremos em maior detalhe na seção sobre Semiótica.

MODELOS DO DESIGN DE SOFTWARE

Developing user-oriented systems requires living in a sea of changes

Nobody can get it right the first time

observações sobre design de sistemas, em Gould *et al* (1997)

Design de software, que costuma ser traduzido em nossa língua por "projeto de software", tenta relacionar a forma e função de um sistema de software à estrutura do processo que produz esse sistema. A tradição herdada de princípios da engenharia apresenta uma abordagem à problemática da crise de software da década de 60, que se baseia na crença de que um processo rigoroso de transformação de requisitos em sistemas é a chave para um design confiável (Denning e Dargan, 1996).

O processo de design na Engenharia de Software (ES) parte de três pressupostos básicos: o resultado do design é um produto, seja ele um artefato, máquina ou

sistema; o produto é derivado de especificações fornecidas pelo cliente – em princípio com conhecimento suficiente e poder de computação essa especificação pode ser mecanizada. Finalmente, uma vez que o cliente e o designer concordaram com as especificações, há pouca necessidade de contato entre eles até a entrega do produto.

O "modelo cascata" (Boehm, 1995, p. 282) caracteriza bem a visão tradicional da ES para o desenvolvimento de software, como um conjunto de processos e representações produzidas de maneira linear (Figura 3.3).

FIGURA 3.3 - O MODELO CASCATA

O principal problema com o modelo cascata é que é impossível entender completamente e expressar os requisitos do usuário antes que algum design tenha sido feito. Além disso, as possibilidades de mudanças no software a partir da etapa

de manutenção são mínimas, em função dos comprometimentos e custos envolvidos ao longo da cadeia.

Em resposta aos problemas do modelo cascata, Boehm (1995, p.284) propõe o "modelo espiral" (Figura 3.4).

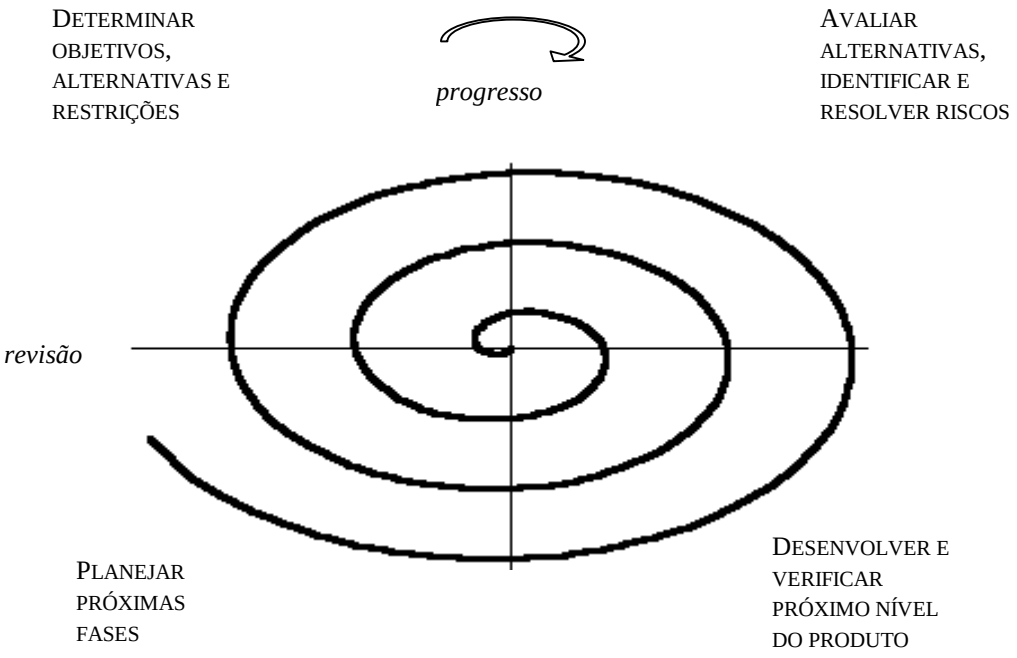


FIGURA 3.4 - O MODELO ESPIRAL

Embora ainda use os mesmos processos do modelo anterior – análise de requisitos, design e implementação – e seja orientado ao produto, o modelo espiral já mostra que várias interações são necessárias e introduz a idéia de prototipagem para maior entendimento dos requisitos. Mas o que leva a um bom design?

Peter Denning (quando presidente da ACM e editor da *Communications*) e Pámela Dargan (software designer) entrevistaram designers de bons sistemas e observaram que suas respostas estavam longe da cultura da ES convencional. O processo de design em engenharia oferece pouca relação entre ações do designer e as necessidades dos usuários, produzindo uma "cegueira" no domínio de ações no qual os usuários vivem e trabalham (Denning e Dargan, 1996).

Como reação à problemática do design centrado no produto, surgiu na década de 80 a escola do design centrado no humano (DCH), que se fundamenta no entendimento

do domínio de trabalho no qual as pessoas estão engajadas e no qual interagem com computadores. Como pressupostos do DCH, Denning e Dargan (1996) destacam: o resultado de um bom design é a satisfação do cliente; o processo de design envolve uma colaboração entre designers e clientes – o design evolui e se adapta aos seus interesses (que também mudam) e esse processo é que produz uma especificação como subproduto. Fundamentalmente o cliente e o designer estão em constante comunicação durante todo o processo.

O DCH tem como objetivo produzir sistemas fáceis de aprender e usar, seguros e efetivos em facilitar as atividades do usuário. Reconhece a importância de testes freqüentes com o usuário usando representações informais e prototipagem. O aspecto central do DCH é o envolvimento efetivo de usuários ao longo do processo de design, não apenas para "comentar" decisões do designer.

Vários modelos para o processo de design têm sido apresentados na literatura de IHC. Entre eles destaca-se o modelo de Eason (apud Preece e*t al.* 1994, p.372), ilustrado na Figura 3.5. Nesse modelo, o processo de design é representado como um processo de natureza cíclica centrado em pessoas, trabalho e tecnologia e é ordenado e não *ad-hoc*.

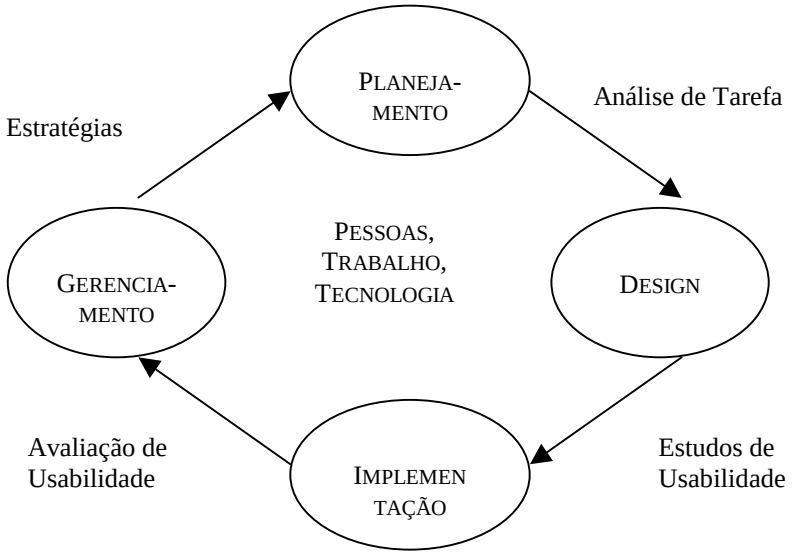


FIGURA 3.5 - O MODELO DE EASON

O "modelo estrela" (Hix e Hartson, 1993), derivado de extensa análise da prática corrente de design à época, é bastante popular entre a comunidade de IHC. Esse modelo, ilustrado pela Figura 3.6, apresenta uma abordagem ao desenvolvimento como "ondas alternantes". As atividades são similares às do modelo cascata, mas a avaliação é central e o início do processo pode acontecer em qualquer uma das demais atividades.

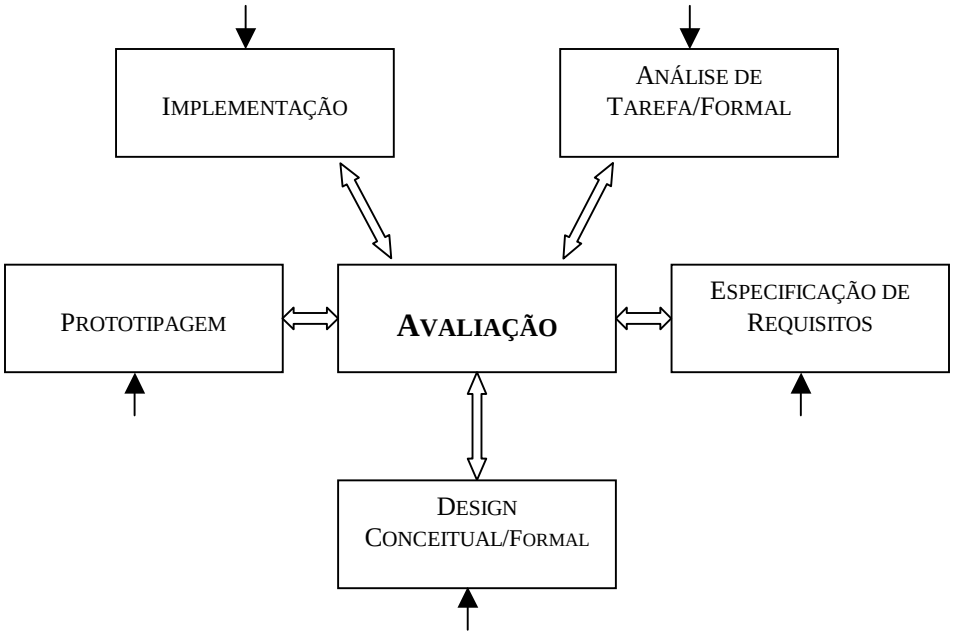


FIGURA 3.6 - O MODELO ESTRELA

As abordagens da ES e da IHC possuem forças e fraquezas complementares; juntas formam uma nova disciplina: a arquitetura de software. Shneiderman (1998) propõe um modelo para design baseado metaforicamente em 3 "pilares" (Figura 3.7). No início do processo o designer deve gerar (ou requerer) um conjunto de *guidelines* (princípios e regras de design, conforme veremos mais adiante neste capítulo). O segundo pilar é composto de ferramentas para prototipagem (HyperCard, Visual Basic, Borland Delphi, Visix Galaxy, Sun Java, etc.). O terceiro pilar é dedicado a testes de usabilidade – avaliação por expertos e testes com usuários.

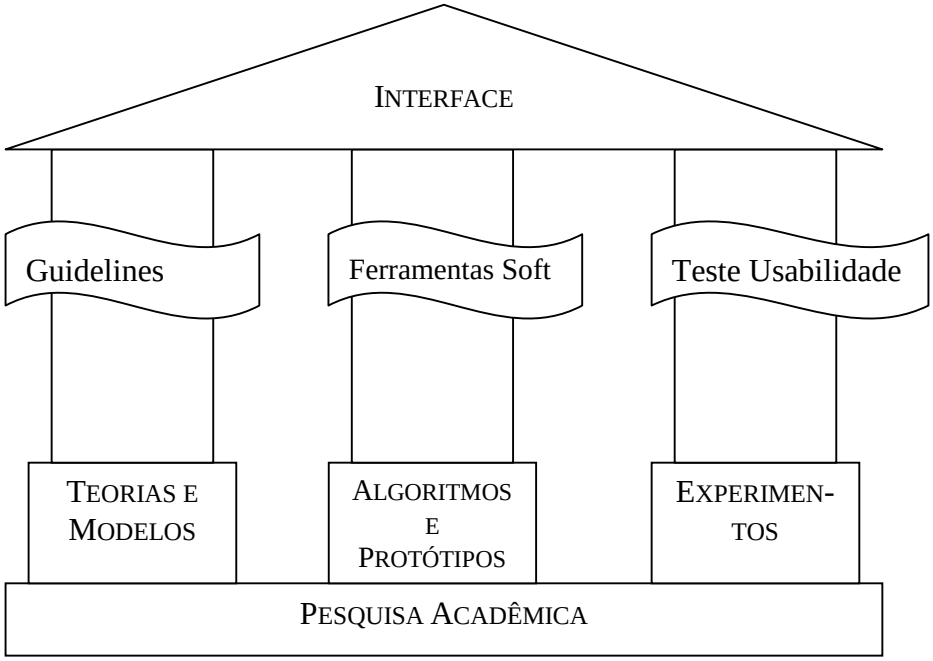


FIGURA 3.7 - O MODELO DE SHNEIDERMAN

Parece já haver um consenso de que qualquer metodologia de DCH deve mesclar-se à metodologias da ES. O modelo LUCID (Logic User-Centered Interface Design), antigo QUE (Quality Usability Engineering) (Kreitzberg, apud Shneiderman, 1998) representa esse esforço. O modelo é representado por uma seqüência de 6 fases, listadas a seguir:

1. Desenvolver o conceito do produto;
2. Realizar pesquisa e análise das necessidades – usando construção de cenários, design participativo, fluxo e seqüência de tarefas;
3. Criar conceitos e protótipos de telas – usando *guidelines*, guias de estilo e metáforas para o design;
4. Design iterativo e refinamento – expandindo o protótipo para sistema completo; inclui a avaliação por expertos e testes de usabilidade;
5. Implementação do software;
6. Suporte;

Vários aspectos influenciam e devem ser considerados na escolha do modelo de design em IHC; entre eles o tipo do sistema a ser desenvolvido, em termos do tamanho, complexidade e propósito. Considerar também se está sendo tratado o design de sistema completamente novo ou de re-design de sistema já existente.

No primeiro caso, há o problema de selecionar entre um grande conjunto de opções e diferentes implicações no design. No segundo caso, a liberdade do designer é severamente restringida por decisões anteriores de design original, linha do produto, etc. Considerar, ainda, as restrições inerentes ao sistema, suas condições de contorno, sistemas críticos em relação à segurança, por exemplo.

Das respostas de uma entrevista feita com designers bem sucedidos à questão de "o que leva a um bom design", Denning e Dargan (1996, p.113) sintetizam as seguintes sugestões:

- Escolha um domínio no qual muitas pessoas estão envolvidas;
- Estude a natureza das ações dessas pessoas naquele domínio, especialmente em ações repetitivas; o que eles reclamam mais? Que ações gostariam de realizar?
- Defina software que imite padrões de ação incluindo funções que não poderiam ser feitas manualmente;
- Crie protótipos o mais cedo possível e observe como as pessoas reagem, o que "quebra" a experiência delas;
- Mantenha comunicação com eles.

ENGENHARIA DE USABILIDADE

It's not broken; that's how it's supposed to work;
We didn't anticipate THIS;
We are surprised that…
The help system will take care of this;
We'll take care of it in the NEXT release…
Comentários de designers de sistemas, em Gould *et al* (1997)

Engenharia de Usabilidade é o termo que se usa para definir o processo de design de sistemas computacionais que objetivam a facilidade de aprendizado, de uso, e que sejam agradáveis para as pessoas.

O processo de design para usabilidade foi inicialmente uma recomendação de vários pesquisadores independentes (Gould e Lewis, 1985; Nielsen, 1992) e grupos de pesquisa na DEC e IBM em Engenharia de Usabilidade que, já na década de 80, constataram que confiar na experiência do designer e em padrões, *guidelines* ou em várias filosofias de design racionais e analíticas não era suficiente para chegar a bons sistemas de computador. A Engenharia de Usabilidade propõe a aplicação de métodos empíricos ao design de sistemas baseados no computador.

As fases do processo surgiram como consenso desses grupos iniciais e estenderam o ciclo tradicional de desenvolvimento que começava com a definição do produto e terminava com sua entrega. O processo de design para usabilidade possui 4 fases: pré-design, design inicial, desenvolvimento iterativo e pós-design.

A fase de pré-design é caracterizada pela busca de informação e conceituação sobre o usuário e seu contexto de trabalho e sobre sistemas relacionados, padrões de interface, *guidelines*, ferramentas de desenvolvimento, etc. A fase do design inicial é constituída da especificação inicial da interface. A próxima fase é a do desenvolvimento iterativo alimentado por feedback de testes até que os objetivos tenham sido alcançados. Finalmente há a fase do pós-design com a instalação do sistema no local de trabalho do usuário e acompanhamento com medidas de reação e aceitação do sistema pelo usuário final.

Os estágios do design para usabilidade ilustram os quatro princípios básicos que fundamentam esse processo: foco no usuário mais cedo, medição empírica, design iterativo e design integrado de todos os aspectos de usabilidade do sistema (Gould *et al.*, 1997).

O estágio de pré-design envolve conhecer os usuários: características individuais (nível escolar, idade, experiência no trabalho, no uso de computadores, sua tarefa atual, como lidam com emergências, etc.) e definir o que eles estarão fazendo com o sistema. Decisões de design subseqüentes – organização do sistema, funções requeridas, interface – devem refletir essas respostas. Também nesse estágio cabe uma análise comparativa de produtos existentes (competidores) e testes com usuários no uso desses produtos. Ainda na fase de pré-design devem ser estabelecidas as metas de usabilidade para o sistema. As cinco características principais de usabilidade, como visto no Capítulo 1 são: facilidade de aprendizado, eficiência de uso, facilidade de retorno ao uso por usuários casuais, freqüência e severidade dos erros dos usuários e satisfação subjetiva do usuário (Nielsen, 1992). Conhecer as metas de usabilidade clarifica o processo de design.

Vários métodos podem ser usados nesse estágio para conseguir o foco cedo e contínuo no usuário: visitas ao local de trabalho do usuário para conhecer a organização do trabalho, observação do usuário em seu trabalho, gravação em fita, do usuário trabalhando, análise de tarefa, design participativo, *think aloud* do usuário, etc.

Os estágios de design inicial e desenvolvimento iterativo têm como premissa básica que não se consegue que o sistema dê certo logo na primeira vez, não importando quão experiente o designer seja. Além disso, não se saberá se o sistema está funcionando até que se comece a testá-lo. Os objetivos desse estágio são, portanto, concretizar em um protótipo o design que segue de princípios de usabilidade e verificar empiricamente o design com usuários reais, para assegurar ter atingido as metas.

Na fase do design inicial é recomendado o uso de métodos participativos, uma vez que, embora os usuários não sejam designers, são muito bons em reagir a design que não os agrada ou não funciona na prática. É recomendado, ainda, o uso de *guidelines* gerais – aplicáveis a qualquer interface, *guidelines* de categoria específica

aplicáveis à classe de sistema sendo desenvolvido e *guidelines* específicas para o produto. Um exemplo de *guideline* geral para interfaces: "falar a língua do usuário". *Guidelines* e Design Participativo serão discutidos separadamente em próximas seções deste capítulo.

O quarto princípio subjacente ao design para usabilidade – o design coordenado (desenvolvimento paralelo da funcionalidade, da interface, do *help*, do material de treinamento, etc.) já aparece nesta fase de design inicial, buscando consistência entre as diferentes mídias que compõem a interface, não apenas às telas. O uso de padrões chamados *de facto* e *in-house*, aumentam o re-uso de código e facilitam a documentação.

A fase do desenvolvimento iterativo é baseada na prototipagem e testes empíricos a cada iteração do ciclo de desenvolvimento. Avaliação qualitativa é aplicada ao sistema em processo de design para verificação dos aspectos da interface que funcionam e principalmente dos que causam problemas. A avaliação heurística é um dos métodos bastante utilizados nesta fase e será tratado no Capítulo 4. Em interfaces quase terminadas são feitas medições quantitativas para checagem das metas.

Muito importante nesta fase é o design *rationale*, um registro que explicita cada decisão de design. Um formalismo que pode ser utilizado para esse registro é o gIBIS (graphical Issue-Based Information System). O *design rationale*, além de manter a memória do processo de design, ajuda a manter a consistência ao longo de diferentes versões do produto.

O estágio de pós-design caracteriza-se por conduzir estudos de campo do produto em uso, para obter dados para nova versão e produtos futuros. Esses estudos devem ir além do registro imediato de reclamações buscando avaliar o impacto do produto na qualidade do trabalho do usuário. Os usuários devem ser visitados em seu local de trabalho e devem ser colecionados registros de sessões de uso do sistema para análise.

Com o objetivo de priorizar métodos de usabilidade, Nielsen (1992) mostra os resultados de um questionário respondido por engenheiros de usabilidade a duas questões: quais métodos teriam usado em projetos recentes e o impacto do método na usabilidade do sistema. Os cinco métodos mais usados (de uma lista de 33) foram: visita ao local de trabalho do usuário no pré-design, design iterativo, design participativo, prototipagem (usando ferramentas computacionais) e análise de produtos competidores. Os 6 métodos de maior impacto na usabilidade foram: design iterativo, análise da tarefa do usuário, teste empírico com usuários reais, design participativo, visita ao local de trabalho do usuário e estudo de campo para verificar como o produto é usado depois de sua instalação.

Entre os principais benefícios da Engenharia de Usabilidade citados na literatura está o tempo economizado em não implementar funções que a análise de usabilidade mostrou não serem utilizadas pelos usuários. Estudo de caso documentado e reportado em Nielsen (1992) mostrou também uma economia financeira equivalente ao dobro do que foi investido, com redução de treinamento para determinados produtos. Além da economia de tempo e dinheiro, a adoção de produtos adicionais é quase certa, se são fáceis de usar.

Design para Usabilidade: porque não? Ainda há os que acreditam que o processo seria encompridado, que iteração é apenas refinamento, que há falta de ferramentas que facilitem o design iterativo e ainda há quem espere uma "abordagem científica e analítica que leve a uma boa interface na primeira vez..."
A Figura 3.8 ilustra, de forma resumida, o Modelo de Engenharia de Usabilidade.

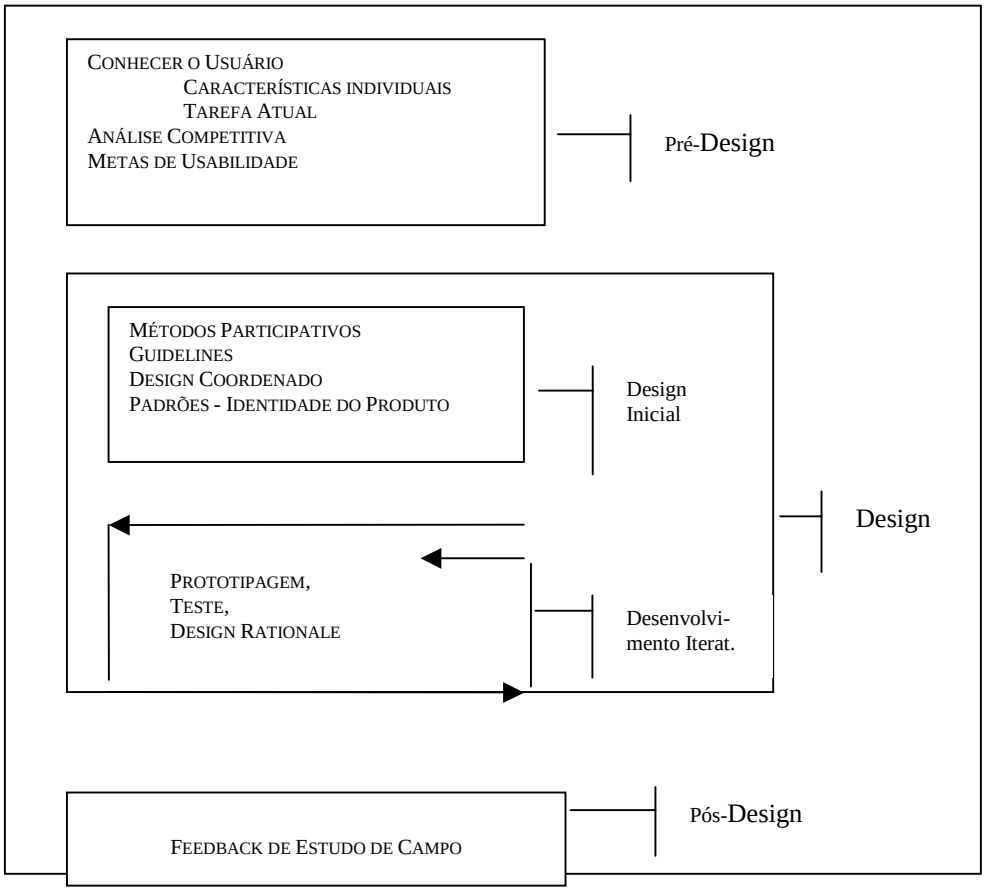


FIGURA 3.8 - O MODELO DA ENGENHARIA DE USABILIDADE

O USO DE *GUIDELINES* EM DESIGN

Falar a língua do usuário

Guidelines são muito populares em design de interfaces por constituírem um *framework* que orienta o designer na tomada de decisões consistentes através dos elementos que constituem o produto. São muito utilizadas por fabricantes que definem, com elas, uma certa identidade à marca. Possuem formas variadas, várias origens – artigos acadêmicos, manuais, estilos associados a marcas, etc. Devem ser entendidas e aplicadas de forma contextualizada.

O uso de *guidelines* não deve ser entendido como "receita de design", mas sim como um conjunto de princípios norteadores do design. Preece *et al.*(1994) argumenta que o verdadeiro sentido das *guidelines* é o de princípios em alto nível, largamente aplicáveis. A seguir comentaremos algumas.

Falar a língua do usuário
Língua deve ser entendida de forma ampla, no contexto sócio-cultural estabelecido da população de usuários. Envolve conhecer essa população, estar atento para as diferentes necessidades do usuário, promover sua satisfação pessoal e permitir que ampliem e facilitem a realização de suas tarefas. Uma interface que fale a língua do usuário ajuda-o a atravessar o golfo de execução e interagir com o sistema. Só para citar um contra-exemplo bastante simples, num editor de textos para crianças, havia como opção de menu para escolha do tipo de letra, o termo "cursiva". Certamente essa não é uma palavra do vocabulário infantil – os usuários finais do sistema eram crianças em processo de alfabetização – que conhecem as "letras de forma" e "letras de mão".

Ao mesmo tempo em que é um princípio e como tal bastante geral, sua aplicação não é simples: envolve reconstruir os sistemas semióticos de uma população que só se conhecerá ao longo do processo de design.

Reduzir a carga cognitiva
Isso significa que o usuário não deve ter que se "lembrar" de grande quantidade de informação para usar bem o sistema. Como vimos no capítulo 2, a teoria psicológica para a capacidade da memória humana define em 7+-2 itens de informação (*chunks*) a capacidade de nossa memória de curta duração. Sobrecarrega-la significa exigir maior processamento cognitivo para atividades de uso do sistema nem sempre relevantes à tarefa propriamente dita. Quantas vezes, ao navegar pela Internet, você se esqueceu do que estava buscando inicialmente?

Criar para o erro

Pressupõe a observação geral sobre design de que mesmo que se tenha feito o melhor sistema possível, usuários – tanto os novatos quanto os experientes – cometerão erros ao usá-lo. O design que considera a condição humana do erro deve forçar ações que previnam ou dificultem o erro do usuário. Prover ações reversíveis ajuda a minimizar a ansiedade e o medo do novato de "destruir alguma coisa". Mensagens de erro efetivas e feedback ajudam o usuário a saber o que fazer quando o resultado de suas ações não produz o que ele espera, atravessando o golfo da avaliação e continuando o ciclo de interação com o sistema. Um bom exemplo de aplicação desta *guideline* é a possibilidade de desfazer operações (*undo*) e repetir operações (*redo*) presentes em algumas interfaces.

Um contra-exemplo bastante simples é a mensagem dos *browsers* para endereços inválidos de páginas Web: "*ERROR 404*"

Manter consistência

Como vimos na seção anterior, consistência emerge do uso de padrões, que são mantidos ao longo do design de todos os componentes que constituem o produto. Consistência também é derivada do uso apropriado de metáforas que ajudam o usuário a construir e manter um modelo mental apropriado do sistema – idealmente coincidindo com o modelo mental do próprio designer.

A imagem ao lado é a janela que é mostrada ao se desinstalar o *FreeLoader*, um *browser* de internet *off-line (Shame, 1999)*. Os *checkboxes* convidam à manipulação do usuário, sugerindo que ele pode indicar quais componentes do software devem ser desinstalados. Ao contrário, porém, os *checkboxes* são usados nesse software para mostrar o progresso do estado de desinstalação! Conforme a desinstalação progride, o sistema vai assinalando nos *checkboxes* as partes já completadas.

Além da necessidade de manter a consistência, *guidelines* devem ser aplicadas de forma cuidadosa, uma vez que, como princípios gerais, são interpretadas. Durret e Trezona (apud Preece *et al.* 1994 p.490) apresentam um princípio de design prescrevendo o número de cores a serem usadas na tela, sugerindo que "não deve ser esperado que o usuário médio lembre (o significado de) mais que 5 a 7 cores...*displays* deveriam ter não mais que 4 cores..."

A *guideline* apresentada mostra uma distorção na interpretação da teoria psicológica na qual a *guideline* foi baseada. O uso exagerado de cores deve ser evitado por

problemas perceptuais – de distração principalmente e não por problemas de cognição – significado, lembrança.

Esse tipo de distorção mostra que simplesmente "aplicar" *guidelines* não leva a um bom design. Princípios de design devem ser interpretados e traduzidos em estratégias que produzam regras de design não-ambíguas, apropriadas ao sistema. Essas regras de design, embora também sejam chamadas de "*guidelines*" por alguns autores, são instruções que podem ser seguidas pelo designer sem exigir muita interpretação. Exemplos de regras: "usar DD-MM-AA para entrada numérica", "posicionar o botão OK no canto inferior direito da tela". Regras são mais comuns nas *guidelines* de determinados fabricantes, para definir a identidade visual da interface e garantir consistência tanto no produto, quanto entre produtos de um mesmo fabricante.

A aplicação de *guidelines*, embora não seja trivial, ajuda o designer a focar no que é necessário e a lidar com restrições e compromissos de design. A definição de *guidelines* a serem utilizadas no design de determinado sistema deve ser acompanhada de exemplificação de seu uso, exceções e dados psicológicos que a justificam. A Tabela 3.1 mostra um exemplo de *guideline* para "formato consistente", extraída de Smith e Mosier, (apud Preece *et al.* 1994 p. 491).

Guideline	***Adotar uma organização consistente para as posições na tela, dos vários elementos do sistema***
Exemplo	Posição para título Área para dados de saída Área para opções de controle Área para instruções Área para mensagens de erro Área para entrada de comando
Exceção	Pode ser desejável mudar formatos para distinguir entre tarefas diferentes
Comentário	Consistência ajuda na orientação do usuário

TABELA 3.1 - EXEMPLO DE PROPOSTA DE *GUIDELINE*

A avaliação de *guidelines* não é tarefa simples; exige conhecimento especializado nas teorias subjacentes que suportam a *guideline*, em dados de uso disponíveis que permitam generalizações, deduções sobre os efeitos da não observância do princípio, etc. Entretanto, fica mais fácil julgar sua aplicabilidade a contextos de design quando apresentadas com seus respectivos argumentos e teorias que as fundamentam, como ilustrado na Tabela 3.1

METÁFORAS NO DESIGN DE INTERFACES

...a good metaphor is essential to an easy-to-use human interface.
(Erickson, 1990)

Metáforas nos ajudam a construir Modelos Mentais sobre o artefato com o qual interagimos e, muitas vezes elas representam nossos Modelos Mentais (ver capítulo 2), permitindo-nos usar conhecimento de objetos concretos, familiares e experiências anteriores para dar estrutura a conceitos mais abstratos.

Lakoff e Johnson (1980) descrevem metáforas como o entendimento e a experimentação de uma coisa em termos de outra. Erickson (1990) define metáfora como um emaranhado invisível de termos e associações que é subjacente à maneira como falamos e pensamos sobre um conceito. É essa estrutura estendida que faz da metáfora parte essencial e poderosa de nosso pensamento. Nossa linguagem é baseada em abstrações metafóricas, como introduzido no Capítulo 1. Muitas coisas são associadas a "dinheiro", por exemplo, o "tempo": gastamos, perdemos, economizamos, roubamos de alguém... A interface de nossos sistemas computacionais está repleta de elementos metafóricos, a começar dos termos "interface", "interação". Na Desktop, a mais famosa das metáforas em interfaces, lidamos com "objetos" na tela, "pincéis" de desenho, "assistentes", em diferentes tipos de "diálogo".

Na metáfora, a comparação entre os domínios origem e destino é implícito. Embora uma metáfora sugira o relacionamento entre os dois domínios, é deixado para o usuário elaborar, descobrir, construir os detalhes da relação (Neale e Carroll, 1997). Bruner (1960) considera as metáforas como um mecanismo de sustentação (*scaffolding*) para o aprendizado, possibilitando que informação previamente aprendida torne-se aplicável a novas situações. O foco no uso de metáforas em interfaces evoluiu da motivação inicial como facilitadora do aprendizado para incluir a facilidade de uso.

OLHANDO MAIS DE PERTO O ASSUNTO

Até que ponto e como elas ajudam o usuário a interagir com sistemas computacionais?

Pessoas usando o processador de texto pela primeira vez vêem similaridade com a máquina de escrever – ambos têm elementos em comum: um teclado, barra de espaço, tecla de retorno. Ambos têm, também, relações em comum: somente um caractere pode ser teclado por vez, ao pressionar-se uma tecla, o caractere correspondente aparece em um meio visível, etc. Essa similaridade é que permite

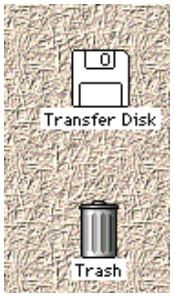
que o sujeito ative o MM da máquina de escrever para interpretar e predizer como o processador de textos funciona. Elementos e relações são, portanto, carregados de um domínio familiar para um domínio não familiar.

Preece *et al.* (1994) distingue dois tipos de metáforas: as verbais e as virtuais. Metáforas podem ser fornecidas na forma de instrução escrita ou falada. Junto com o processador de texto, por exemplo, instruções de como os arquivos são criados, armazenados e recuperados no sistema podem ser apresentadas em termos do "arquivo de aço". Em vez de criar metáforas verbais, a Xerox criou a metáfora na interface, com base no escritório real, tornando virtual o topo da mesa de escritório. *Star* apresentava o equivalente eletrônico para os objetos físicos do escritório – papel, pastas, documentos, arquivos, bandejas, e ações similares: abrir, fechar, copiar, etc. Em vez de serem entidades abstratas, com nomes arbitrários, arquivos foram transformados em representações pictóricas, fáceis de identificar e entender.

A metáfora "verbal" convida o usuário a "perceber" as similaridades e diferenças entre o sistema e o domínio familiar. A metáfora "virtual" é parte da interface, e combina o sistema e o domínio familiar em uma nova entidade. Através de metáforas virtuais o usuário é levado a desenvolver um MM mais próximo do mundo metafórico – MM funcional e não um modelo do sistema subjacente (MM estrutural).

Nem tudo é similar, entretanto. Há um ponto em que a metáfora deixa de acomodar características do sistema novo. Por exemplo, a tecla de retrocesso no processador de textos também apaga o último caractere. O *Shift-Caps*, em teclas com mais de um tipo, não aciona o tipo superior, só funcionando para expressar letras maiúsculas. Essa perda de paralelismo entre o domínio familiar e o não familiar apresenta contraposições às expectativas do usuário de como os elementos e suas relações funcionam. Há propriedades que não são mapeáveis de um domínio para outro. Em algum momento o usuário precisará entender como o sistema novo funciona, como um sistema computacional que é. A metáfora do desktop, na realidade é uma composição de metáforas, assim criada para permitir flexibilidade de ação. Por exemplo, a barra de rolagem é um objeto que não existe no escritório real. Menus e janelas foram emprestados de outros contextos.

Uma metáfora na interface que sugira o MM incorreto causa dificuldades ao usuário. Por exemplo, para eliminar arquivos e documentos, a lixeira é uma metáfora intuitiva. Entretanto, no Macintosh a metáfora foi estendida para incluir uma função nova: o "*eject*" do disquete. O arraste do disquete para a lixeira para retirá-lo do computador é incompatível com a associação metafórica anterior e causava problemas conceituais ao usuário, que tinha medo de ter o conteúdo de seu disquete deletado.



O uso de metáforas do mundo real em interfaces tem causado muita polêmica, em função do emprego equivocado de metáforas em algumas interfaces. A imagem ao lado mostra a caixa de diálogo da impressora *Mannesman Tally* (*Shame*, 1999), que utiliza a metáfora do VCR para controlar a impressora. Uma pergunta básica: na tarefa de impressão de um documento, que associação o designer esperaria que o usuário fizesse com o botão de *Rewind*?

Outro exemplo mal informado do uso de metáforas em interfaces é apresentado no *Read-Please2000* (*Shame*, 1999), uma aplicação útil para tradução de texto para fala, em que não é clara a associação que deva ser feita com um *Palm Pilot.*

Como conseqüência do papel que as metáforas exercem em possibilitar que o usuário construa MMs de forma a tornar os sistemas mais fáceis de usar, os designers precisam de métodos sistemáticos que as incorporem ao design.

COMO GERAR METÁFORAS ADEQUADAS NA INTERFACE?

Conscientes de que o objetivo da metáfora na interface é prover o usuário com um modelo do sistema com o qual deverá interagir, Erickson (1990) propõe o uso de metáforas em design através de um processo baseado nas seguintes etapas: (1) entender a funcionalidade do sistema a ser criado; (2) como nenhuma metáfora consegue modelar todos os aspectos da funcionalidade de um sistema, deve-se identificar as partes mais difíceis para o usuário; (3) metáforas que "suportem" o modelo requerido, devem ser geradas e avaliadas. Na geração de metáforas candidatas, notar metáforas já implícitas na descrição do problema e procurar eventos reais, objetos ou organizações que incorporem algumas das características que os usuários acham difícil entender. Das metáforas geradas, escolher uma através da qual será expressa a funcionalidade do sistema, com base nos aspectos de estrutura, sua aplicabilidade, poder de representação, adequação à audiência e possibilidade de extensão.

Em relação à estrutura, o objetivo é verificar quanto de estrutura a metáfora provê para o usuário pensar no sistema. Em relação à aplicabilidade deve ser verificado quanto da metáfora é relevante ao problema; metáforas que podem conduzir o

usuário na direção errada ou levantar falsas expectativas devem ser evitadas. Quanto à representação, metáforas ideais têm representações visuais distintas e palavras específicas associadas. Em relação à adequação à audiência, deve-se verificar se o público-alvo entende a metáfora; ela seria inútil em caso negativo. Estrutura adicional é uma característica desejável, uma vez que extensão do sistema pode ser necessária mais tarde.

Madsen (1994, p. 59-60), com base em estudos de casos realizados e coletados da literatura, propõe uma série de *guidelines* para o design baseado em metáforas. O autor distingue 3 diferentes atividades nesse processo: (1) geração de metáforas candidatas à aplicação no design; (2) avaliação com relação à adequação ao domínio particular de tarefas e (3) desenvolvimento ou seja adaptação da metáfora à situação de design. Detalharemos a seguir as *guidelines* sugeridas para cada fase.

- **Fase (1) – geração de metáforas:**
  Observar como os usuários entendem seus sistemas computacionais.
  Construir sobre metáforas já existentes.
  Usar artefatos predecessores como metáforas.
  Notar metáforas já implícitas na descrição do problema.
  Procurar eventos do mundo real que exibam aspectos chave.

- **Fase (2) - avaliação de metáforas candidatas ao design:**
  Escolher uma metáfora com uma estrutura rica.
  Avaliar a aplicabilidade da estrutura.
  Escolher uma metáfora adequada à audiência.
  Escolher metáforas com significado literal bem entendido.
  Escolher metáforas com uma distância conceitual entre a fonte e o significado metafórico.
  Ter pelo menos um conceito como "ponte" entre o significado literal e o metafórico.
  Não necessariamente incorporar a metáfora no design final.

- **Fase (3) - desenvolvimento do sistema propriamente dito:**
  Elaborar o conceito principal.
  Procurar novos significados para o conceito.
  Reestruturar a nova percepção da realidade.
  Elaborar suposições tornando explícito o que a metáfora esconde e o que ela salienta.
  Contar a estória da metáfora, falando do domínio alvo como se ele fosse o domínio fonte.
  Identificar as partes não usadas da metáfora.
  Gerar situações de conflitos.

Na pesquisa mais recente, as metáforas são concebidas como mapeamentos entre domínios que possibilitam ao usuário usar conhecimento e experiências específicos

de um domínio familiar, para entender e se comportar em situações que são novas. Nesse processo, além do uso de representações emprestadas de domínio familiar, vários autores sugerem que a concepção do design baseado em metáforas deva incorporar também o contexto de uso de tais representações, resultando em influências maiores no design de sistemas.

O Jogo do Alvo (Nied, 2000) é um exemplo do uso do tiro ao alvo como metáfora para visualização de conceitos de Controle Estatístico de Processo (CEP). Como mostra a Figura 3.9, a disposição dos tiros no alvo representa visualmente a distribuição de medidas de determinada peça, em relação a um valor nominal almejado (o centro do alvo).

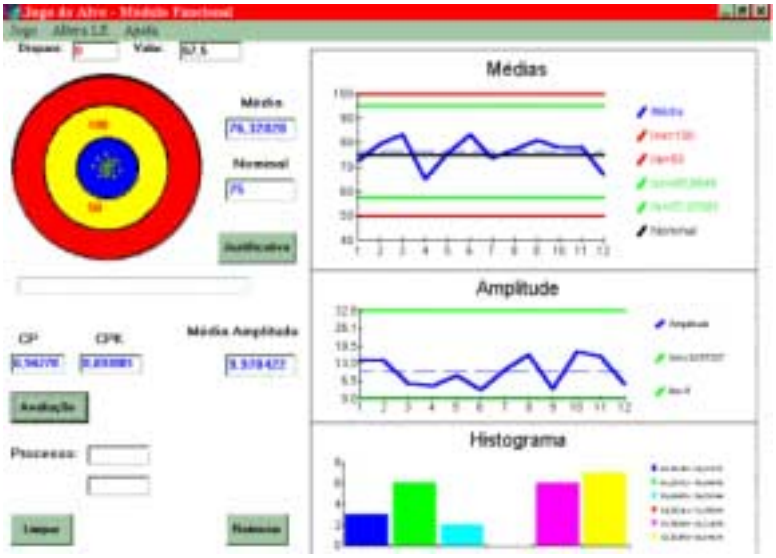


FIGURA 3.9 - TELA DO JOGO DO ALVO

O alvo é uma metáfora já implícita na descrição do problema, uma vez que o objetivo no processo de manufatura é conseguir produzir peças cuja medida seja o mais próxima possível do valor nominal (o centro). Além de possuir um significado literal bem entendido, a estrutura da metáfora serve para representar condições de estabilidade do processo, pela distribuição dos tiros no alvo.

Para o trabalhador da linha de manufatura, a metáfora do alvo consegue captar abstrações do processo e facilitar a interpretação de situações reais sem exigir dele sofisticação matemática. O alvo permite reestruturar uma nova percepção da realidade da manufatura com base na associação de conceitos do CEP com configurações de tiros no alvo.

DESIGN BASEADO EM CENÁRIOS

A mudança de paradigma que nos leva a enxergar os computadores não apenas como artefato tecnológico, mas principalmente como artefato da cultura humana, leva-nos a repensar os objetivos e métodos de design e desenvolvimento de sistemas.

Como artefato tecnológico, o computador deve produzir resultados corretos, ser confiável, executar eficientemente, ser fácil – ou possível de manter. Como artefato da cultura, entretanto, outros requisitos são necessários: os computadores devem ser acessíveis a pessoas de outros domínios de conhecimento, não envolvidos diretamente com a tecnologia; para tal devem ser fáceis de aprender e de usar. Além de atingir as expectativas das pessoas, devem estender as atividades humanas melhorando a qualidade de vida e levando o usuário à satisfação de suas experiências de trabalho, educação e lazer.

Como implicação dessa nova concepção do uso de computadores, são necessários métodos flexíveis e ricos para incorporar descrições de usuários potenciais e dos usos que eles podem fazer de um sistema imaginado à lógica do design. Idealmente o próprio usuário deveria ser envolvido nesse processo.

Carrol (1997) propõe para o processo de design, buscar novos vocabulários e representações – acessíveis aos próprios usuários – para discussão e caracterização de design em termos de atividades projetadas. Além disso, tais representações devem ser integradas e coordenadas com outras produzidas ao longo do desenvolvimento do sistema. A avaliação de alternativas de design deve ser feita, também, com critérios orientados ao uso e integrados à avaliação tradicional de correção, confiabilidade, eficiência, etc.

Cenários foram propostos por Carrol (1997) como um meio de representar, analisar e planejar como um sistema computacional pode causar impacto nas atividades e experiências do usuário. Um cenário é uma descrição em geral narrativa, mas também em outros formatos, que as pessoas fazem e experimentam conforme elas imaginam ou tentam fazer uso de sistemas e aplicações.

Um exemplo simples de cenário mostrando a narrativa de uma situação na qual uma pessoa (usuário) descreve como ela imagina o uso de um sistema prospectivo, é mostrado a seguir.

> "***Cenário 1**: formação em planejamento e gerenciamento sem interromper o trabalho normal*
>
> *A fábrica X tem um problema. Está sendo pressionada pela matriz, que fica em outro país, para melhorar a formação e capacidade de planejamento e gerenciamento de seus funcionários, estimulando-os a se familiarizar com sistemas novos como JIT, TQC, TPM e outros. Vários exercícios, folhetos, cursos de treinamento impressos e em vídeo são recebidos da matriz para treinar e estimular a formação de funcionários em todos os níveis nos novos sistemas. Só que estes exercícios e cursos requerem a presença de vários funcionários ao mesmo tempo, de vários setores da fábrica durante muitas horas, interferindo com a produção da fábrica.*
>
> *A matriz também sugere treinamento e ensaios com clientes e fornecedores. Mas os principais clientes e fornecedores de X estão dispersos por várias partes do Brasil e é inviável convidá-los para participar dos cursos e exercícios presenciais. Os exercícios não são utilizados e a formação dos funcionários fica postergada.*
>
> *Consultado, o Nied-Unicamp sugere o desenvolvimento de sistemas computacionais de suporte à atividade de formação a distância e o uso da Internet, Intranet e Extranet montados pela empresa, para implementar os exercícios e cursos para os funcionários junto com os clientes e fornecedores. Todos estes funcionários podem participar destas atividades de formação em horas convenientes à não-interrupção da produção."* (extraído de Mazzone, J. documento interno Nied-Unicamp)

Cenários iniciais do tipo exemplificado ajudam o designer a clarificar as metas de design. Desenvolver e explorar o espaço de design com um mínimo de comprometimentos ajuda o designer a entender o que é necessário fazer e conhecer as percepções individuais do usuário para com o sistema, atitudes em direção à colaboração, etc.

Cenários podem captar conseqüências e compromissos de design a serem analisados. Por exemplo, do cenário pode-se extrair considerações do tipo:

> *"as redes de comunicação baseadas no computador são um meio bastante interessante para a situação descrita no cenário 1, mas essa solução de design deve considerar se há cultura do uso desse meio na fábrica X, além das questões tecnológicas relacionadas à velocidade da informação nessas redes"*

Embora a forma mais comum para representar cenários seja a textual, podem estar também na forma de *story-boards, cartoons* anotados, maquetes em vídeo, protótipos em *script*, etc. O formato usado para a expressão de cenários tem sido bastante flexível e variado entre os praticantes do design baseado em cenários. Karat e Bennett (apud Preece *et al* 1994) propõem os seguintes componentes para cenários:

Nome – um rótulo curto para referência a um cenário específico;
Descrição – em geral texto ilustrando uma situação específica;
Lógica Essencial – com relação ao usuário, representações e ações que devem estar disponíveis ao usuário, independentemente de aspectos relacionados à implementação; com relação ao sistema, informações necessárias para que o sistema funcione como requerido;
Passos Genéricos – seqüência de passos que o usuário realizaria, independentemente de aspectos de implementação;
Passos específicos – seqüência de ações do usuário seguidas de feedback do sistema, considerando possibilidade de ações erradas do usuário.

Os níveis de descrição variam, podendo ser articulados em diferentes níveis de granularidade para especificar mais precisamente a funcionalidade do sistema. A seguir exemplificamos um segundo cenário para sistema relatado no Cenário 1, com base em sistema desenvolvido e relatado em Baranauskas et al (1999):

Cenário 2 *– Iniciando Jogo da Fábrica: atividade síncrona a distância e baseada em sistema computacional*

<***descrição***> *- Funcionários da fábrica X conectam-se via Internet e iniciam Jogo da Fábrica: uma simulação de conceitos e processos de manufatura com objetivo de formação.*

<***lógica essencial***> *- (usuário) Cada usuário em seu local de trabalho, ao conectar-se ao sistema, vê a tela inicial do jogo e informações sobre a conexão dos demais participantes, cada um ocupando uma célula da linha de manufatura representada no jogo. Cada usuário saúda os demais que estão conectados a distância, pelo canal de comunicação, aguardando início do jogo.*
(sistema) Informação necessária para conexão: IP da máquina servidora

<***passos genéricos***> *- Buscar opção de conexão no menu. Entrar com dados solicitados.*

<***passos específicos***> *- Selecionar ".....*
...
Entrar com IP do servidor"

O cenário identifica o usuário como tendo certas motivações para o uso do sistema, descreve as ações tomadas e razões para essas ações. Para o designer, ajuda a visualizar aspectos da atividade e experiência adquirida ou necessária do usuário.

Carrol (1997) propõe o uso de cenários ao longo de todo o ciclo de design e desenvolvimento do sistema. Durante a fase de análise de requisitos designers e usuários negociam explicitamente os cenários e descrições do domínio de uso do sistema. Descrições de cenários hipotéticos facilitam na descoberta das necessidades do usuário que não são óbvias ou aparentes para eles próprios.

Cenários podem ser a unidade de análise para desenvolver o design *rationale*, explicando as decisões de design através de cenários particulares de interação do usuário e da análise de cenários alternativos.

Na fase de design propriamente dita, cenários podem ser analisados para identificar os objetos centrais do domínio do problema e articular o estado, comportamento e interação funcional dos objetos de design. A abordagem *use-case* em análise orientada a objetos faz uso desse tipo de visualização de situações concretas com o objetivo de identificar objetos computacionais essenciais no sistema.

Na fase de avaliação, cenários podem ser usados para se coletar informação detalhada de como os usuários percebem o sistema. Design de telas podem ser apresentados a usuários potenciais que tentam explicar o que pensam ser possível fazer e efeitos esperados de suas ações.

Mesmo na fase de implementação o uso de cenários apresenta benefícios, uma vez que ajuda a manter os designers focados em dar suporte às atividades dos usuários.

A documentação do sistema pode facilitar, também, o próprio treinamento do usuário; as pessoas fazem melhor uso da documentação se ela é apresentada no contexto de tarefas típicas que eles têm que realizar, especialmente as mais críticas.

A proposta de design baseado em cenários, de certa maneira, se contrapõe à abordagem convencional para o ciclo de design e desenvolvimento de sistemas. Na abordagem estabelecida tradicionalmente, as descrições são abstratas, com foco em tipos genéricos, o processo é completo, exaustivo, e orientado à tecnologia, os métodos são formais e rigorosos e os resultados bem especificados. Na abordagem baseada em cenários as descrições são concretas, com foco em instâncias particulares, o processo é fragmentado e aberto, orientado ao trabalho, o método é informal e coloquial e os resultados imaginados apenas.

Uma contribuição importantíssima da abordagem dos cenários no processo de design é o estabelecimento de um canal de comunicação de mão dupla, usuário-designer. Cenários são a língua franca da ação e experiência do usuário final.

A proposta do design baseado em cenários pressupõe a visão de sistemas computacionais como transformadores das tarefas do usuário e de suas práticas sociais. Estórias são elementos coesivos importantes em qualquer sistema social; os cenários que a equipe compartilha motivam e direcionam a construção do grupo. Representam, ainda, uma busca pelo equilíbrio entre intuição criativa e análise, entre a flexibilidade e a informalidade, necessários para a evolução sistemática em direção a um sistema usável.

DESIGN PARTICIPATIVO

Design with the user,
rather than design for the user...
(Kuhn e Winograd, 1996)

Conforme o título indica, o Design Participativo (DP) caracteriza-se pela participação ativa dos usuários finais do software ao longo de todo o ciclo de design e desenvolvimento. Mais do que serem usados como fontes de informação ou serem observados em sua rotina de trabalho, ou no uso do produto, os usuários finais trazem contribuições efetivas em todas as fases do ciclo de design e desenvolvimento, que refletem suas perspectivas e necessidades. A participação do usuário não é restrita aos estágios de testes de protótipos ou avaliação, mas acontece ao longo do processo de design e desenvolvimento.

O Design Participativo em geral acontece no local de trabalho, incorporando o usuário não somente como sujeito de observação e experimentos, mas como membro da equipe de design. Três características específicas definem o DP: ele é orientado ao contexto (de trabalho), envolve colaboração em vários níveis e apresenta uma abordagem iterativa ao design.

Em uma mesa redonda na Conferência sobre Design Participativo de 1994, Tom Erickson, da Apple Computer (citado em Kuhn e Winograd, 1996), definiu quatro dimensões ao longo das quais a participação do usuário pode ser medida: 1. a diretividade da interação com os designers; 2. a extensão do seu envolvimento no processo de design; 3. o escopo de participação no sistema como um todo; 4. o seu grau de controle sobre as decisões de design.

O movimento do Design Participativo teve origem no início da década de 70, na Noruega, com Kristen Nygaard (um dos criadores de Simula), quando este colaborou com o sindicato para criar o *Codetermination Agreement*, especificando os direitos dos trabalhadores de participar em decisões de design relativas ao uso de novas tecnologias no trabalho.

Outro marco inicial do DP foi o Projeto DEMOS, que ocorreu na segunda metade da década de 70 e envolveu uma equipe interdisciplinar de pesquisa nas áreas de Ciência da Computação, Sociologia, Economia e Engenharia. Era financiado pela *Swedish Trade Union Federation* e o mote do projeto era *Trade Unions, Industrial Democracy and Computers* (Kuhn e Winograd, 1996).

Entre as motivações para o uso de abordagens participativas em design estão: a questão da democracia, o compromisso com o desenvolvimento organizacional, a eficiência, expertise, qualidade potenciais, e a efetividade do ponto de vista epistemológico.

A idéia do DP foi concebida em sua formulação escandinava original, no contexto de um movimento em direção à democracia no local de trabalho: o desenvolvimento de competências e o poder de o trabalhador exercer influência em decisões que afetariam seu trabalho. Nessa proposta há um compromisso implícito com o desenvolvimento organizacional, através da crença de que o sistema terá mais chances de ser aceito se seus usuários finais estiverem envolvidos no processo.

Eficiência, expertise e qualidade seriam conseqüências dessa abordagem. A efetividade do design e desenvolvimento de software é aumentada se inclui a expertise dos próprios usuários. A eficiência é aumentada se os usuários finais provêem entradas para outros design e feedback em um design completo. A qualidade no design e no sistema resultante é aumentada através de um melhor entendimento do trabalho do usuário e melhor combinação do background dos diversos participantes.

Finalmente a efetividade epistemológica é sustentada pela premissa básica de que nenhuma pessoa ou disciplina, isoladamente, tem todo o conhecimento necessário para o design do sistema.

A concepção original do DP é, portanto, sustentada pela crença de que o trabalho democrático no nível de design tem o potencial de melhorar ambos: tanto o processo de desenvolvimento do software quanto o trabalho dos usuários. Nesse sentido, democracia, epistemologia e efetividade comercial caminham juntas.

Convém ressaltar aqui que a terminologia do “design participativo” tem sido usada em modelos de design que não necessariamente concordam com as motivações do DP, para simplesmente expressar alguma forma de participação do usuário, ou para se referir ao uso isolado de métodos do DP.

Desejamos distinguir essa apropriação da linguagem e de alguns métodos, do DP propriamente dito. Nosso objetivo nesta seção é, portanto, clarificar o conceito de participação como infraestrutura e referencial teórico subjacente para um grupo de

métodos e processos complexos, que não são lineares. Discutiremos, a seguir, uma taxonomia para práticas participativas em design.

Os métodos em DP caracterizam-se pelo uso de técnicas simples e pouco comprometimento com recursos; as técnicas de *brainstorming*, *storyboarding* e de *workshops* são bastante utilizadas.

Embora seu uso seja mais conhecido na fase de design propriamente dito, as práticas participativas estendem-se ao longo de todo o ciclo de vida (convencional) do software, desde a fase de identificação do problema, passando pelas fases de levantamento e análise de requisitos, design, avaliação, customização e re-design.

A seguir apresentaremos resumidamente alguns métodos que podem ser utilizados em diferentes momentos do processo de design e desenvolvimento do software. Nosso objetivo é ilustrar, do ponto de vista pragmático, o DP. Para uma visão mais abrangente, Muller (1997) apresenta uma coleção bastante extensa de 61 práticas participativas utilizadas em design.

Em nossa descrição, abordaremos o que o método faz, que materiais usa, como as pessoas se comunicam umas com as outras, como utilizam os materiais, como tomam decisões, quem é envolvido no trabalho e se existem papéis especiais de certos membros da equipe, quais os benefícios que são produzidos, como o resultado é usado.

Storytelling Workshop é um método usado na fase de identificação e clarificação do problema de design.

Cada participante de um grupo de usuários finais e facilitadores (max. de 20 pessoas) traz para a oficina duas histórias curtas sobre o uso de sistemas computacionais – em geral experimentadas em seu trabalho. Uma história deve ser positiva e outra negativa com respeito ao resultado desse uso. Os participantes compartilham suas histórias, comentando semelhanças e contrastes de suas experiências. Nenhum material especial é requerido. Como resultados da oficina, são apontados: uma coesão aumentada entre os usuários finais e entre esses últimos e os designers, reconhecimento das dificuldades e consciência de que elas não são únicas, conhecimento de características e dificuldades da população de usuários pelos designers.

Picture Card é um método utilizado na fase de análise de requisitos, em situações nas quais os usuários finais e profissionais de design e desenvolvimento do software não compartilham, ainda, a mesma linguagem. Eles se comunicam usando cartões pictóricos para desenvolver a representação do trabalho.

São utilizados como material cartões contendo figuras de objetos e eventos do mundo de trabalho do usuário. Esses cartões são agrupados em seis categorias:

Pessoa, Ação, Estação, Ferramenta, Evento, Local (PAEFEL). Os cartões são arranjados em seqüências lineares começando com as categorias PAEFEL e refinando-as em subclasses específicas, refletindo as histórias e cenários do ambiente de trabalho.

Como resultado, as histórias contadas pelos usuários, inicialmente expressas através de cartões, são traduzidas em texto; para o designer servem para tornar explícito o dialeto do trabalho e alimentar o dicionário de objetos, ações, etc.

O tamanho do grupo deve ser pequeno, com alguns usuários "contadores de histórias", o designer que controla a mesa de cartões e observadores.

HOOTD- Hierarchical Object-Oriented Task Decomposition é um método para ser utilizado na fase inicial de design, embora também possa ser útil na fase de análise.

Participantes decompõem uma descrição de tarefa em objetos e ações e assinalam grupos desses objetos a janelas de interface. Cartões são usados como material.

Cada participante, em paralelo, escreve cada tarefa – representada por um substantivo e um verbo – em seu cartão. Os participantes, então em grupo, ordenam esses cartões em pilhas segundo critério de escolha do grupo. O esquema é registrado. Reordenam, então, segundo outros critérios, registrando todos os esquemas. Escolhem um dos esquemas. Cada pilha do esquema escolhido torna-se uma tarefa do domínio, contendo em uma janela da interface os objetos e ações da pilha de cartões.

Como resultados tem-se a definição das janelas de interface e seus respectivos objetos. É recomendável que o grupo não seja grande. Análise de tarefas (com o GOMS, por exemplo) pode ser usado como método formal complementar.

BrainDraw é um método participativo para uso na fase de design propriamente dita. É constituído de um *brainstorming* cíclico, gráfico, com o objetivo de preencher rapidamente um espaço de várias opções de design para a interface.

O material é composto de papel e canetas arranjados em uma série de estações de desenho colocadas em círculo. Cada participante faz um desenho inicial em uma das estações. Ao final de um intervalo de tempo estabelecido, cada participante move-se para a estação seguinte e continua o desenho lá encontrado. O processo continua rodando até que todos tenham colaborado na criação de cada um dos outros participantes.

Como resultado tem-se a geração de muitos design candidatos à interface do sistema, cada um deles tendo a participação de todos os envolvidos. Cada design resultante é a fusão da idéia de todos e não são idênticos uma vez que cada um deles teve um início diferente.

Um grupo de tamanho médio (aproximadamente 10 pessoas) de usuários finais, designers e artistas são os participantes potenciais. Alternativamente ao movimento dos participantes pelas estações de desenho, os desenhos podem rodar pelos participantes colocados em círculo.

Icon Design Game é um método participativo que pode ser utilizado na fase de design para a criação dos ícones e símbolos gráficos da interface. Um dos participantes (*sketcher*) desenha ícones enquanto que os outros tentam "adivinhar" o conceito que o *sketcher* está tentando expressar. Os desenhos tornam-se rascunhos para a criação de ícones.

O material utilizado é composto de papéis de desenho e canetas. O participante no papel de *sketcher* seleciona um conceito e tenta comunicar ao grupo apresentando desenhos relacionados ao conceito. O grupo tenta descobrir enquanto um observador toma notas sobre desenhos que parecem mais efetivos ou mais confusos. Os desenhos que expressam melhor o conceito são passados para a produção gráfica dos ícones.

Como resultado tem-se, portanto, *sketches* de ícones para arte final. Pode-se usar o método, também na escolha de metáforas para a interface.

Dependendo do tamanho do grupo, este pode funcionar no estilo cooperativo ou subdividido em vários grupos para produção competitiva dos desenhos.

CISP – Cooperative Interative Storyboard Prototyping é um método que pode ser usado em várias fases do ciclo de design e desenvolvimento: análise de requisitos, design e avaliação.

Uma equipe de designers e usuários gera e modifica cooperativamente designs de interfaces, avaliam interfaces existentes comparando alternativas. Um software associado ao método ou outro ambiente para criação de *storyboards*, como o *HyperCard, Borland Delphi*, etc. em geral são utilizados como material.

O processo envolve iterações de 3 passos principais: exploração do *storyboard* para realização da tarefa pelo usuário final, enquanto o software registra; avaliação do *storyboard*, através de análise e discussão do registro da interação; modificação do *storyboard*.

Como resultado tem-se o *storyboard* ou o protótipo melhorados e o registro da interação dos usuários. Podem ser usados em complemento ao CISP na fase de avaliação, métodos de inspeção de usabilidade (que serão apresentados no Capítulo 4).

Buttons Project é um método para ser utilizado no pós-design, na customização do sistema pelo usuário final.

O material utilizado para compartilhar customização é um software que suporta o design de funções customizáveis. Usuários compartilham suas customizações enviando botões uns para os outros.

Através de templates os usuários especificam funcionalidades em botões. Enviam esses botões uns para os outros. Receptores de botões podem modificá-lo.

Como resultado tem-se nova funcionalidade compartilhada entre os usuários, além do registro das inovações na forma de customizações executáveis.

Priority Workshop é um método utilizado no re design de um sistema. Usuários e designers colaboram na prática do re design através de uma seqüência de oito atividades conduzidas em formato de workshop.

O processo é iniciado com uma discussão introdutória de objetivos. Segue-se uma apresentação dos usuários sobre características positivas, negativas e desejáveis no sistema. Segue-se uma apresentação dos designers sobre planos e prioridades relativas ao sistema. A quarta atividade é a exploração conduzida em pequenos grupos de protótipos (em papel) alternativos. Segue-se uma discussão em plenário. Um sumário de prioridades e qualidades são rotuladas com + ou - pelos usuários. Segue-se uma discussão das conseqüências – para os usuários – das mudanças. O método termina com uma discussão final de planos de continuidade do processo.

Como resultado têm-se decisões sobre características a serem incluídas e/ou modificadas no re-design do sistema.

A Tabela 3.2 mostra a distribuição dos métodos participativos descritos nas diferentes etapas do processo de design.

Pré-design	Design	Avaliação	Pós-design
Identificação/Requisitos	Inicial/Iterativo	Testes	Customização/redesign
StoryTelling PictureCard	HOOTD BrainDraw	CISP	Buttons Priority Workshop

TABELA 3.2 - MÉTODOS PARTICIPATIVOS EM DIFERENTES ETAPAS DO DESIGN

Muller (1997) discute a relação entre práticas participativas e modelos formais e contratuais de desenvolvimento de software e comenta que certas metodologias orientadas a objeto encorajam a construção de *use cases*, conforme comentado anteriormente, como cenários para atividades do usuário relacionadas ao sistema.

Embora a terminologia seja semelhante, o foco da atenção nos *use case* é o software e o paradigma ainda é o orientado ao produto. O modelo de casos de uso não é substituto para o trabalho com usuários finais, uma vez que a definição das ações do usuário é feita pelos designers.

Também, normas como a ISO9001 para assegurar a qualidade do produto, encorajam um acordo ou contrato entre a organização que desenvolve o software e a organização onde o usuário trabalha. O usuário não é representado formalmente nessa relação. O foco do padrão é a qualidade técnica e necessidades das gerencias em contraste com a qualidade de uso, estética e necessidades do usuário final.

Há vários questionamentos sobre as dimensões éticas e políticas na forma de participação do usuário no design do sistema. Algumas técnicas de extração do conhecimento, por exemplo, nos sistemas especialistas, enfatizam a "participação" dos trabalhadores com o objetivo de aumentar o conhecimento que será usado pelo profissional de desenvolvimento.

Muller (1997) cita também a questão do usuário como "objeto"; certos testes de usabilidade tratam os usuários como indicadores de medidas da produtividade associada ao produto, sem considerar suas necessidades, conforto, qualidade do ambiente de trabalho. Não é o usuário quem escolhe quais atributos de sua experiência são relevantes.

Há ainda o resultado da participação de determinados clientes potenciais para determinação de atributos atrativos ao mercado, para uso em campanhas de marketing. Outro questionamento que se coloca é a ilusão de controle que a participação pode dar ao usuário enquanto que o poder de decisões continua sob o controle da hierarquia superior da organização.

Derivadas do DP, várias abordagens ao design com foco em aspectos da "democracia industrial" são discutidas na literatura: *situated activity* (Suchman, 1987), *work-oriented design* (Ehn,1988), *design for learnability* (Brown, Duguid, 1992), *situated design* (Greenbaum, Kyng, 1991), entre outras.

MÉTODOS ETNOGRÁFICOS EM DESIGN DE INTERFACES

Theorizing or reflecting and engaging in some form of practice are essential to every kind of human endeavor, be it research, design, or any other form of work practice

Suchman e Trigg (1995, p.238)

Ao contrário dos métodos experimentais em Psicologia, onde os sujeitos são submetidos a situações criadas em laboratório, antropólogos e sociólogos usam

métodos etnográficos para estudar pessoas *in the wild*, em seu habitat nativo. A meta da antropologia é aprender sobre todos os aspectos de uma cultura. A observação participativa, um dos principais métodos da etnografia, ajuda o pesquisador a enxergar o mundo através dos olhos do nativo (Nardi, 1997). A observação participativa desenvolveu-se no final do século 19 e início do século 20, quando antropólogos americanos foram a campo estudar e documentar aspectos da cultura de sociedades americanas nativas.

A abordagem etnográfica em design de software é tratada no trabalho pioneiro de Suchman (1987), onde ela propõe que o ideal para a investigação da tecnologia em uso é aquele em que a atividade de trabalho ocorre naturalmente em cenários construídos pelos próprios participantes.

Entre os objetivos e tarefas principais da abordagem etnográfica em design está o entendimento da prática corrente do trabalho das pessoas usando tecnologias. Podemos estar interessados em entender como as pessoas usam um espaço compartilhado de desenho para um trabalho conjunto de design, a partir do cenário dos próprios participantes. Ou podemos estar interessados em entender como as pessoas usam o protótipo de uma nova ferramenta para fazer seu trabalho, em cenário que pode ser construído pelo pesquisador.

Vários tipos de registros da observação podem ser realizados, de forma a captar em diferentes mídias, diferentes aspectos do ambiente observado. Em registros orientados ao ambiente propriamente dito, uma ou várias câmeras de vídeo são posicionadas de forma a cobrir o máximo possível da atividade sendo analisada, no espaço físico. Por exemplo, numa secretaria de atendimento ao público, uma câmera captaria o balcão de atendimento aos usuários, outra o espaço físico do arquivo de aço que guarda documentos em papel, outra o computador. Esses registros poderiam mostrar a freqüência de cada tipo de atividade – atendimento no balcão, uso de sistemas computacionais, uso do arquivo em papel, procedimentos e rotinas usuais, tipos de interrupção, relações entre as diferentes atividades, etc.

O registro pode ser orientado à pessoa, quando estamos interessados em entender o trabalho do ponto de vista de uma determinada pessoa em determinada função no ambiente de trabalho. Uma câmera acompanha a pessoa na seqüência de atividades que realiza. Dificuldades, ações repetitivas, re-trabalho são exemplos de dados que poderiam ser extraídos desse registro.

O registro pode ser, ainda, orientado a um objeto ou artefato tecnológico, para captar, por exemplo, situações de uso desse artefato. Na análise de usabilidade de um protótipo de sistema, por exemplo, uma câmera pode captar toda a seqüência de ações de um usuário interagindo com o sistema. O objetivo, neste caso, é analisar a interface com respeito à adequação ao trabalho a ser realizado.

Quando o objetivo é entender a tarefa, são feitos registros de pessoas que têm uma meta em comum, em diferentes locais e momentos da composição de determinada tarefa. Tomando como exemplo uma secretaria de cursos na Universidade, uma determinada tarefa – a geração de determinado relatório – pode envolver também a secretaria dos departamentos, que possui parte da informação necessária ao relatório – dados de publicações de docentes, por exemplo. Um registro orientado à tarefa envolve, portanto, registros de sub-tarefas realizadas por pessoas diferentes em diferentes locais. Relações de precedência, pressuposição, interdependência, etc. podem ser captadas desses registros.

Após o registro de uma extensa atividade, a segunda parte dos procedimentos etnográficos é a transcrição do registro para a análise inicial. No caso do registro em vídeo, o primeiro passo deve ser a descrição dos eventos observados, indexando-os cronologicamente. Uma transcrição cuidadosa do material deve incluir também as dimensões não vocais da interação: gestos, postura, por exemplo. Ainda, ações das pessoas usando máquinas ou sistemas computacionais devem ser concatenadas com registros obtidos no próprio sistema (*logs*) para análise da interação pessoa-computador e análise da inter-relação com atividades de outras pessoas.

Na análise da interação, o objetivo é descobrir as regularidades na ação das pessoas no uso dos recursos do ambiente, com outras pessoas e com o sistema computacional (ou artefato). Para isso são construídas "coleções": instâncias de interação que queremos ver como uma classe (Suchman e Trigg, 1995). Ao colecionar essas instâncias, as características comuns e distintivas ficam mais visíveis.

Embora a fita de vídeo não elimine a necessidade de interpretação do analista, ela corrige nossa tendência de "ver em uma cena o que esperamos ver", conforme já discutimos no Capítulo 2, Suchman e Trigg, (1995) citam um exemplo bastante interessante desse fenômeno: quando em certas circunstâncias um casal é observado sorrindo um para o outro, estando próximos fisicamente, é comum a inclusão de "toques" no relato do observador, mesmo quando não existiram na realidade.

A análise da interação é um intensivo e extenso trabalho de ver e rever várias vezes o registro feito, transcrevendo e buscando seqüências relevantes à análise. A fita de vídeo traz informações que a edição de texto – no caso de registros da observação através de anotações – não possibilita. É muito difícil captar em palavras os movimentos de mover o cursor, por exemplo, interpretar ao mesmo tempo a expressão do usuário, etc. A densidade dos detalhes de comportamento, e a ausência de vocabulário para anotações, faz do registro em vídeo uma das formas mais utilizadas nos métodos etnográficos.

OBSERVAÇÃO DIRETA OU INDIRETA?

Usuários individualmente podem ser observados diretamente, fazendo seu trabalho normal ou tarefas específicas para a situação de observação. O observador, então,

toma notas de comportamentos interessantes ou registra seu comportamento de outras maneiras, por exemplo, medindo o tempo de realização de seqüências de ações.

A observação direta é considerada o método de observação mais invasivo, uma vez que o usuário fica o tempo todo consciente de que está sendo observado por outra pessoa e sua performance está sendo monitorada. Como efeitos pode haver alteração no comportamento e no nível de performance desse usuário. Essa alteração tanto pode desfavorecer quanto favorecer os resultados de uso do sistema ou artefato. Como favoreceria? Alguns usuários podem ter sua performance na tarefa melhorada, não propriamente pelos benefícios do uso do sistema, mas também por se sentirem "valorizados" ao receberem maior atenção de outros (observadores no caso). Esse fenômeno é conhecido como "efeito Hawtorne".

Outra desvantagem da observação direta é que ela permite apenas um passo na coleta de dados, o tempo real da observação. Além disso, é nesse mesmo tempo que o avaliador deve tomar decisões do que é importante registrar. Finalmente, o avaliador raramente consegue um registro completo da atividade do usuário, embora o uso de uma notação abreviada e de um *checklist* previamente estabelecido ajudem no registro de ocorrência de determinados eventos.

A observação indireta através de gravação em vídeo cria uma distância maior entre o observador e o usuário, minimizando o sentido "invasivo" da observação. A gravação pode ser sincronizada com outros registros da interação do usuário com o sistema – arquivos *log*, por exemplo, gerando quadro completo da interação. Vários aspectos da atividade podem ser captados por câmeras diferentes; uma pode focalizar o teclado e a tela enquanto a outra focaliza o usuário, registrando para onde ele olha na tela, se consulta outro material, linguagem do corpo, etc.

Como haverá muito mais dados a analisar, o que consumirá mais tempo do designer, a observação precisa ser planejada. Deve ser definido quando começar e terminar, onde colocar fisicamente o equipamento, etc. Mesmo sendo menos invasivo do que a observação direta, os usuários são conscientes de que seu comportamento está sendo gravado. Uma maneira de reduzir o impacto é deixar o equipamento no local da gravação, por vários dias antes de começar (Preece *et al*., 1994).

A análise dos dados coletados em vídeo pode ser baseada em tarefas, quando o objetivo da análise é saber como os usuários lidam com a tarefa, onde estão as maiores dificuldades e o que poderia ser feito. A observação pode referir-se ao contexto de trabalho do usuário, ainda sem o sistema computacional – alvo do processo de design, ou pode referir-se à tarefa mediada por versões do protótipo do sistema em processo de design.

A análise pode, ainda, ser baseada na performance do usuário, gerando esquemas classificatórios para freqüências de acertos, de erros, tempo gasto para realização de

partes da tarefa, freqüência de uso de determinados elementos de interface, tempo usado em atividades cognitivas: pausas em comandos, entre comandos, em leituras inspecionando áreas de interface, etc.

Protocolos pós-evento são, também, elementos de informação importantes na análise da observação. Os usuários são convidados a ver a gravação, comentar sobre suas ações. Quando são convidados a participar da análise dos dados, eles são estimulados a relembrar detalhes úteis sobre seus problemas. Essa releitura do usuário pode, entretanto, trazer uma "racionalização" de suas ações que poderiam não corresponder exatamente ao acontecido de fato.

Protocolos pós-evento são essenciais em situações de observação de tarefas que requerem cuidadosa concentração do usuário e o seu tempo é crítico como, por exemplo, em salas de controle de tráfego aéreo, salas de terapia intensiva e outras situações na área de saúde.

Protocolos verbais são registros das falas do usuário e representam uma dimensão a mais à informação coletada, pois expressam parte da atividade cognitiva subjacente ao comportamento físico – ações, postura, gestos – do usuário. O espectro da observação é aumentado com informações sobre a maneira como o usuário planejou realizar a tarefa, a sua identificação de nomes de menus e ícones, suas reações quando alguma coisa "dá errado", seu entendimento de mensagens fornecidas pelo sistema, sentimentos subjetivos expressos no tom de voz, em comentários que faz, etc.

Os protocolos verbais são obtidos com o método do "pensar alto" (*think aloud*): o usuário é convidado a dizer em voz alta o que está pensando enquanto realiza a tarefa. Informações extraídas de protocolos verbais são riquíssimas para o processo de análise, mas representam um esforço cognitivo extra para o usuário na realização da tarefa. O usuário deve fazer, ao mesmo tempo, a tarefa propriamente dita e falar sobre suas ações e o que está pensando. Além do nível meta cognitivo envolvido (pensar sobre o pensar), sabemos da Psicologia Cognitiva que nossos mecanismos perceptuais, cognitivos e motores são pouco eficientes em manter nossa atenção dividida por mais que poucos minutos. Um bom exemplo simples desse esforço, discutido no capítulo 2, é a tarefa de conversar enquanto se dirige um automóvel.

Outro problema a lidar no registro de protocolos verbais é como prevenir longos períodos de silêncio do usuário, uma vez que pensar, fazer e falar ao mesmo tempo não é uma situação natural. Criar um arranjo para a situação a ser observada de modo que dois usuários trabalhem juntos na realização da tarefa e possam conversar entre si. Outra estratégia seria permitir que o usuário faça perguntas ao observador, que deve responder minimamente, apenas para dar continuidade à tarefa. Podem também ser usados pelo observador, *promptings* do tipo "como você faz...?", "porque você faz ...", etc. Essa forma de protocolo da interação será revista no Capítulo 4, quando discutirmos testes de usabilidade.

Um método de registro bastante popular para captar informações de uso de versões mais completas de protótipos de sistemas computacionais é o *logging*. O método não requer a presença do pesquisador e parte do processo de análise pode ser automatizado; não é um método invasivo, embora possa levantar questões éticas: os usuários devem ser informados desse registro. Já existem várias ferramentas para se fazer o *logging* de software. Laboratórios de usabilidade costumam fazer uso dessas ferramentas. Há ferramentas que registram cada tecla que o usuário pressiona e o tempo exato do evento. Outras registram a interação entre o usuário e o sistema, de forma que o observador pode vê-la exatamente como ocorreu. Sistemas de *playback* (Neal e Simons, 1983, apud Preece et al., 1994) possibilitam que o observador veja na tela de seu monitor colocado em sua sala, as entradas do usuário e as respostas do sistema. O observador pode, ainda, adicionar comentários para cada operação do usuário.

A escolha do método de observação a ser utilizado em geral é um compromisso entre o tempo a ser gasto e a profundidade da análise. Um feedback informal sobre determinadas tarefas mediadas pelo sistema em design pode ser obtido em poucos dias, através de observação direta ou indireta. Para um entendimento mais detalhado das ações do usuário, observação indireta combinando gravação de vídeo com *loggings* são mais adequadas. É necessário coletar e analisar protocolos, selecionando medidas de performance relevantes, em geral revendo a fita várias vezes. Essa atividade consome tempo de análise numa proporção de 5 para 1 em relação ao tempo de registro. O desenvolvimento de ferramentas, para análise de vídeo baseada em computador, já aparece na literatura em caráter experimental (Suchman e Trigg, 1995; Preece *et al.*. 1994).

A aplicação da análise de vídeo em design é uma constante no DIAL – *Designer Interaction Analysis Lab*, um laboratório de design da Xerox no Centro de Pesquisa de Palo Alto. Nele, um grupo de antropólogos, cientistas da computação e designers desenvolve métodos que encorajam o movimento da análise para o desenvolvimento e para a análise novamente, e cooperam na aplicação desses métodos em problemas concretos de design de sistemas.

A meta da abordagem etnográfica em design é associar intuições e possibilidades tecnológicas a um entendimento detalhado da prática do trabalho real. Suchman e Trigg (1995) ilustra essa abordagem com o design do sistema Commune - uma ferramenta multi-usuário para desenho. Nesse projeto, o grupo busca entender como o uso do espaço compartilhado de trabalho suporta e é organizado pela estrutura de atividade de desenho.

A Figura 3.10, adaptada de Suchman e Trigg (1995, p.238) ilustra os fundamentos da abordagem etnográfica em design, enquanto representa graficamente o mote do início desta seção. Design, prática e pesquisa são as três perspectivas necessárias ao processo de criação. Dependendo do vértice em que nos encontramos, temos uma determinada visão do problema e essa posição não deve ser fixa; a visão a partir de

cada uma delas, em maior ou menor grau, é necessária no processo de criação dos artefatos tecnológicos que mediam nossas tarefas.

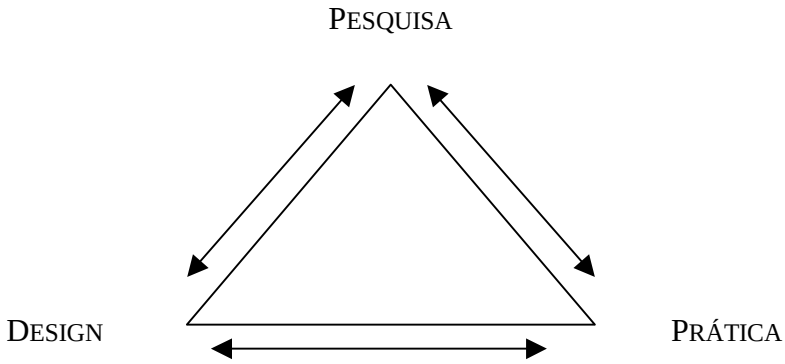


FIGURA 3.10 - ABORDAGEM ETNOGRÁFICA EM DESIGN

SEMIÓTICA EM SISTEMAS COMPUTACIONAIS

The pencil of the drawing program is not a real pencil that can be used to chew on, it merely stands for a pencil, represented by a collection of pixels on the screen
(Andersen, 1997, p.1)

As abordagens cognitivas à conceituação e ao design de interfaces, como apresentamos anteriormente estão fundamentadas principalmente nas propostas de Card, Moran e Newell (1983) do Modelo do Processador de Informação Humano (discutido no Capítulo 2) e na Teoria da Ação e Engenharia Cognitiva de Norman e Draper (1986), discutidas neste capítulo. Nesse paradigma, a interação do ser humano com o computador é governada por atividades de interpretação e avaliação realizada por usuários que têm o desafio de traduzir metas para eventos de entrada no computador e julgar reações do sistema a partir de sua percepção de elementos de saída do sistema computacional. Aspectos de comunicação são associados à diretividade semântica e articulatória e à inter-referencialidade dos elementos da entrada e saída do sistema.

Mesmo sendo o computador membro da classe dos artefatos simbólicos, somente em anos recentes, na medida em que deixou de ser ferramenta exclusiva de especialistas, e a sofisticação do software cresceu, é que essa natureza simbólica passou a atrair a atenção de grupos que estudam fatores humanos e interfaces. Embora ainda possa ser considerado uma "ferramenta", em analogia a outras como, por exemplo,

máquinas de escrever, pincéis de pintura, pastas de arquivos, etc., o computador difere destas ferramentas por não existir ou ser usado primariamente como objeto físico, mas sim como sistema de signos. Conforme bem coloca Andersen (1997, p.1), o lápis do programa de desenho não é um lápis real, ele meramente "representa" (está para) um lápis, através de uma coleção de pixels na tela.

Os sistemas computacionais estão, cada vez mais, mediando nossas ações. Em particular, com a nova tendência de uso da tecnologia de redes de computadores, com espaços virtuais compartilhados e trocas de mensagens ele passou a ter funções similares às de outras mídia, onde a importância da Semiótica como referencial já é bem estabelecida. Na perspectiva semiótica o papel do computador é basicamente o de um *medium – uma substância na qual signos podem ser manifestados para uso em comunicação* (Andersen, 1997, p. 333).

Semiótica como disciplina teve seu desenvolvimento a partir dos trabalhos do filósofo norte-americano Charles Sanders Peirce (1839-1914) e do lingüista suíço Ferdinand de Saussure (1857-1915). Peirce e Saussurre formam as duas matrizes principais da Semiótica contemporânea. Os trabalhos de Saussurre têm origem na lingüística enquanto que a Semiótica de Peirce é desenvolvida dentro de um corpo filosófico e é concebida como Lógica. Alguns autores diferenciam as duas vertentes usando o termo "semiótica" apenas à linha desenvolvida por Peirce enquanto usam o termo "semiologia" para as teorias derivadas da proposta original de Saussurre. Não faremos distinção da terminologia e conceitos neste livro. Uma apresentação e discussão mais detalhadas das duas abordagens podem ser encontradas em Oliveira e Baranauskas (1998 a). Os conceitos básicos são apresentados aqui segundo a proposta de Peirce.

A Semiótica objetiva estudar os signos e sistemas de signos. Um signo é qualquer coisa que está no lugar de outra coisa sob determinados aspectos ou capacidades, para alguém (Peirce, CP2.228). Isto é, qualquer marca, movimento físico, símbolo, sinal, etc. usado para indicar e "transportar" pensamentos, informações e comandos constituem signos (Sebeok, 1994, p.xi). Uma foto é um signo na medida em que ela "está para" os elementos nela representados, para alguém que a interpreta. Se, na interpretação de alguém, a palavra "amarelo" está para a cor amarelo, a pronúncia da palavra "cavalo" está para o animal cavalo, fumaça está para fogo, o desenho de uma impressora na tela de um computador está para imprimir, então a palavra "amarelo", a pronúncia de "cavalo", a fumaça e o desenho da impressora na tela são todos exemplos de signos. Observe que, sem o signo, nossa comunicação no mundo seria muito pobre, uma vez que seríamos obrigados a nos comunicar fazendo uso, apenas, dos próprios objetos a que queremos nos referir, conforme discute Santaella (1996, p.7).

A Semiótica tem por objetivo a investigação de todas as linguagens possíveis, ou seja, a investigação de qualquer fenômeno como fenômeno de produção de significado e sentido. Seu campo de atuação é vasto; é matéria semiótica qualquer

signo produzido ou interpretado por nós, seres humanos, ou por outros animais, plantas, protozoários, fungos e bactérias, artefatos desenvolvidos por alguma entidade viva ou super-natural (Sebeok, 1994, p.6).

Na Semiótica Peirceana, o signo é apresentado como uma relação triádica entre o objeto – aquilo que é representado, o *representamen* – aquilo que representa e o interpretante – o processo de interpretação, conforme ilustra a Figura 3.11.

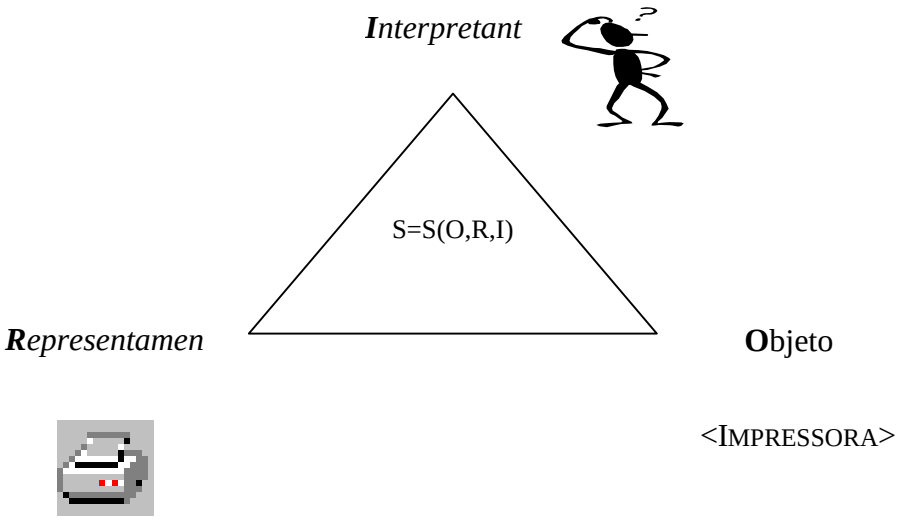


FIGURA 3.11 - O SIGNO DE PEIRCE COMO UMA RELAÇÃO TRIÁDICA, EXEMPLIFICADO

O *representamen* representa o objeto, sob certos aspectos e capacidades; ele não é o objeto. O interpretante não é o intérprete do signo, mas sim um processo relacional criado na mente do intérprete. Peirce refere-se à "mente" como um conceito formal, não propriamente na acepção psicológica do termo. O *representamen* se coloca em uma relação triádica com seu objeto de modo a determinar que o interpretante assuma a mesma relação com o objeto. A relação triádica é genuína, no sentido de que seus 3 membros estão por ela ligados de modo a não consistir em nenhum complexo de relações diádicas (Peirce, 1974).

Da definição de signo para Peirce, decorre o conceito de semiose ilimitada – *semiosis*, ilustrado pela Figura 3.12. O interpretante é um processo de geração infinita de significações: aquilo que é um terceiro numa relação triádica, torna-se primeiro em outra relação triádica. O interpretante determinado por um objeto transforma-se em um *representamen* de um outro signo que remete a outro objeto, num processo que determina um novo interpretante e assim sucessivamente.

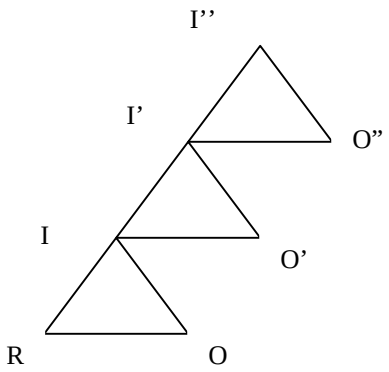


FIGURA 3.12 - O PROCESSO DE SEMIOSE ILIMITADA

Algumas coisas têm como razão primária funcionar como signo, por exemplo, letras, sons vocais, sistemas de computador, etc. Um sistema de reserva de vôos "está para" aviões, lugares, vôos, etc. A representação de um objeto e as conseqüentes interpretações dessa representação podem ser classificadas nas categorias icônica, indicial ou simbólica. **Representações icônicas** são baseadas nas semelhanças e características comuns ao objeto a que se referem; desenho de uma impressora na interface de determinado software é um ícone para a impressora real e a tarefa de imprimir. Representações que guardam a relação de causa e efeito entre objeto e *representamen* são chamadas índices; fumaça usada como *representamen* para fogo ou o desenho de uma ampulheta "significando" o correr do tempo são exemplos de **representações indiciais**. Representações baseadas em convenções estabelecidas são chamadas **símbolos**, a exemplo da linguagem natural e de formalismos lógico e matemático. Palavras reservadas em linguagens de programação são exemplos de símbolos. A Figura 3.13, adaptada de Nadin (1988) ilustra de forma operacional essa classificação dos signos.

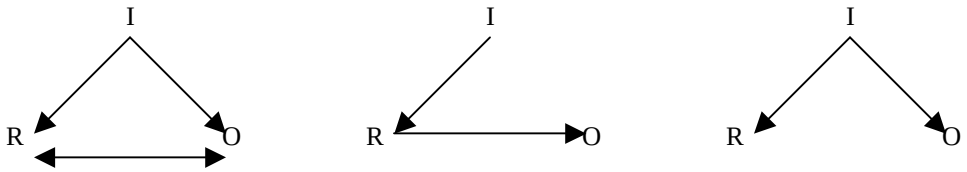


FIGURA 3.13 REPRESENTAÇÕES ICÔNICA, INDICIAL E SIMBÓLICA (ADAPTADO DE NADIN, 1988, P.55)

Como a tecnologia sobre a qual construímos interfaces muda muito rapidamente, princípios semióticos fornecem fundamentos para o design e avaliação de interfaces de forma mais compreensível. Por exemplo, o entendimento do que um ícone representa – e não propriamente o que ele "retrata" – é essencial no design da linguagem da interface. Consideremos, por exemplo, os sinais gráficos que popularmente chamamos de ícones na interface; na verdade não são todos ícones,

conforme conceito apresentado, mas muitos deles guardam uma relação simbólica com seus referentes no mundo. Dos vários exemplos que podemos encontrar nas interfaces gráficas, considere o botão de *forward* da ferramenta de *email* Eudora Light:  . A informação codificada fisicamente para comunicar informação nesse botão estabelece uma relação simbólica (baseada em convenção) com seu referente – o programa *Forward*. Ao contrário, o sinal  (presente no botão de *forward* do *Netscape Mail*) estabelece uma relação icônica com seu referente, pois neste caso o conjunto de características do sinal e do referente no mundo têm intersecção não vazia.

Entre os esforços de trazer a Semiótica ao contexto do design de software, vários autores dentre eles Nadin (1988), Andersen (1990, 1997) e Souza (1993) fizeram contribuições seminais.

Nadin (1988) introduz uma das primeiras tentativas de aplicar a Semiótica ao design de interfaces, com base na teoria de Peirce. A Figura 3.14 mostra como a unidade básica da Semiótica, isto é, o conceito de signo, é entendido por Nadin no contexto do computador.

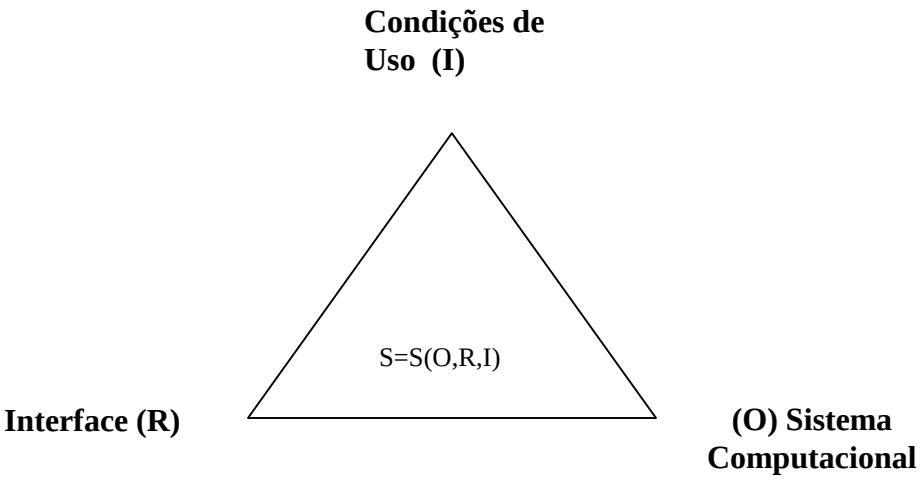


FIGURA 3.14 - O CONCEITO DE SIGNO NO CONTEXTO DO COMPUTADOR (ADAPTADO DE NADIN, 1988, P.58)

Na proposta de Nadin, a interface do sistema – o *representamen*, ou aquilo que representa o objeto, torna "transparente" através das três representações possíveis (icônica, indicial, simbólica), qualquer tarefa ou ação a ser realizada com a utilização do computador. Segundo o mesmo autor, se no processo de constituição da unidade interface-aplicação-condições para uso e avaliação, representada na Figura 3.13, um dos elementos impõe requisitos que afetariam sua relação de reciprocidade, a

necessidade de melhorar o design do artefato deve ser reconhecida. São exemplos de condições onde essa relação é afetada: interfaces não suficientemente transparentes para as aplicações que representam, aplicações difíceis de usar ou inapropriadas à atividade desejada.

Na perspectiva de Nadin, o foco da atenção no design de software deve ser colocado na semiótica da comunicação de sua interface. Um escritório (real) não é uma coleção de arquivos, calculadoras, etc. – isto é, uma coleção de "objetos concretos", mas um ambiente onde comunicação é necessária (troca de documentos, armazenamento e recuperação de dados, planejamento, etc.). Comunicação é entendida como a atividade semiótica que coloca usuário e designer juntos através de um intermediário que são as linguagens que eles usam. Designer e usuário, como parte de uma dada cultura, compartilham convenções estabelecidas e participam no estabelecimento de novos sistemas de signo, se necessário. Linguagens de programação são exemplos de sistemas semióticos que tornam possíveis novas ferramentas de natureza cognitiva (Nadin, 1988, p.70).

Andersen (1990, 1997) encontrou na escola européia criada por Saussurre e desenvolvida por Hjelmslev, o substrato teórico que o levou a propor a "Semiótica Computacional" – uma aplicação da Semiótica não apenas ao design de interfaces, mas também à programação, análise e projeto de software.

A interface é definida por Andersen (1997, p.143) como uma coleção de signos baseados no computador, isto é, uma coleção das partes do software que podem ser vistas ou ouvidas, usadas e interpretadas por uma comunidade de usuários. Para Andersen, o design da interface deve emergir de padrões de uso, ou seja, da maneira como o usuário faz uso do "dialeto" baseado no computador. Design é visto como um processo iterativo no qual propostas são continuamente desenvolvidas, usadas e avaliadas. Em cada iteração desse processo de design, há um conjunto de signos para ser analisado. As relações entre as unidades constituintes da linguagem da interface são analisadas e como resultado modificações são propostas com o objetivo de adaptar o design dos signos da interface a padrões de uso do dialeto baseado no computador.

Os signos baseados no computador, diferentemente dos signos usados nas linguagens verbais, são transientes no sentido de que, ao longo do tempo, podem alterar suas características como cor, ou posição que ocupam na tela. Para acomodar essa característica, Andersen propõe que a análise de cadeias de signos baseadas no computador leve em conta cadeias concorrentes, isto é, aquelas compostas de signos e partes de signos que ocorrem juntos no mesmo ponto do tempo e cadeias seqüenciais – cadeias ou partes de signos que ocorrem um após o outro em diferentes pontos no tempo. Exemplos do uso da abordagem de Andersen em análise, design e re-design de interfaces podem ser encontrados em Baranauskas *et al.* (1998), Prado e Baranauskas (1999) e Rossler (2000), entre outros.

Souza (1993) propõe a Engenharia Semiótica, para o design de linguagens de interface de usuário. Na Engenharia Semiótica (ES), a interface é entendida como um artefato de meta-comunicação, isto é, a interface é composta por mensagens enviadas do designer para o usuário e cada mensagem, por sua vez, pode enviar e receber mensagens do usuário. Nesse sentido, a interface cumpre dois papéis: (1) comunicar a funcionalidade da aplicação (o que a interface representa, que tipos de problemas está preparada para resolver) e o modelo de interação (como se pode resolver um problema); (2) possibilitar a troca de mensagens entre o usuário e a aplicação.

Na Engenharia Semiótica o foco está na comunicação unidirecional e indireta do designer para com os usuários. O designer cumpre um papel comunicativo explícito ao utilizar a interface para dizer algo ao usuário. Por sua vez, o usuário cumpre os papéis de agente da interação e de receptor da comunicação indireta do designer.

Em sua proposta original, a Engenharia Semiótica apoia-se na teoria da produção de signos de Eco (1997), para definir *guidelines* teoricamente motivadas para design de linguagens de interface de usuário. Atualmente um conjunto considerável de trabalhos estende e dá corpo à ES (Leite e Souza, 1999; Prates e Souza, 1999; Martins e Souza, 1998; Barbosa e Souza, 1999).

Um novo entendimento para o conceito de interface a partir de bases semióticas está sendo proposto por Oliveira (2000), onde a interface é entendida como um espaço de comunicação para entidades humanas e não-humanas (botões, heróis em jogos, janelas, etc.) que participam do "jogo semiótico" comunicando-se pela sua aparência e pela sua capacidade de produzir e interpretar signos. Nessa proposta, bases semióticas são aplicadas ao design das entidades propriamente ditas, sua consubstanciação na interface e ao design da comunicação entre elas (Oliveira e Baranauskas, 1999).

O desenvolvimento das teorias cognitivas em Interação Humano-Computador trouxe-nos uma visão do computador como ferramenta cognitiva que nos possibilita aumentar nossas capacidades de entendimento, memorização, tomada de decisão, etc. As abordagens semióticas ao design de software permitem-nos considerar, não apenas os aspectos imediatos (físicos) da interação com computadores, mas também seu aspecto interpessoal e cultural focando na expressão e interpretação dos elementos na interface do software. A visão da interação mediada por sistemas semióticos representa um paradigma relativamente recente para o design de software; resultados de pesquisa têm sido discutidos nos principais fóruns de IHC, no exterior, (por exemplo, CHI2000) e em nosso país (IHC99, IHC98, entre outros).

REFERÊNCIAS:

Andersen, P.B.(1990, 1997) *A Theoy of Computer Semiotics. Semiotic Approaches to Construction and Assessment of Computer Sustems.* Cambridge: Cambridge University Press.

Andersen, P.B., Holmqvist, B. e Jensen, J.F.(1993) *The Computer as Medium*, Cambridge University Press.

Baecker, R. , Grudin, J., Buxton, W. e Greenberg, S. (eds.)(1995) *Readings in Human-Computer Interaction: Toward the Year 2000.* San Mateo, CA: Morgan Kaufmann.

Card, S.K., Moran, Newell, A.(1983) *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates

Baranauskas, M.C.C., Gomes Neto, N.G., Borges, M.A.F. (1999) Gamming at Work: a Learning Environment for Synchronized Manufacturing. *Proceedings of The International Conference on Engineering and Computer Education*, ICECE99 Rio de Janeiro, Brasil.

Baranauskas M C C, Rossler, F, Oliveira, O L (1998) Uma Abordagem Semiótica à Análise de Interfaces: um estudo de caso. Em *Atas do I Workshop sobre Fatores Humanos em Sistemas Computacionais*, Maringá, PR, p. 75-84.

Barbosa S D e Souza C S (1999) Making More Sense out of Users' Utterances. Em *Atas do II Workshop sobre Fatores Humanos em Sistemas Computacionais*, Campinas, SP, p.29

Boehm, B.W. (1995) A Spiral Model of Software Development and Enhancement. Em *Readings in Human-Computer Interaction: Toward the Year 2000,* R. Baecker, J. Grudin, W. Buxton, S. Greenberg, (eds.) San Mateo, CA: Morgan Kaufmann.

Brown, J.S., Duguid, P. (1992) Enacting Design for the Workplace. Em *Usability: Turning Technologies into Tools*, P. Adler, T. Winograd (eds). NY:Oxford University Press

Bruner, J.S. (1960) *The Process of Education*. London: Oxford University Press

Carrol, J.M. (1997) Scenario-Based Design. Em *Handbook of Human-Computer Interaction*, M.G.Helander, T.K.Landauer, P.V.Prabhu (eds) 2nd.edition, Elsevier.

CHI2000 http://peirce.inf.puc-rio.br/chi2000ws6/

Denning, P. Dargan, P. (1996) Action-Centered Design. Em *Bringing Design to Software,* T. Winograd (ed) Ma:Addidon-Weslley, pp. 105-127.

Eco, U. (1997) Tratado Geral de Semiótica. Editora Perspectiva, São Paulo, Brasil.

Ehn, P. (1988) *Work-Oriented Design of Computer Artifacts*. NJ: Lawrence Erlbaum Ass.

Erickson, T.D. (1990) Working with Interface Metaphors. Em B. K. Laurel (ed) *The Art of Human-Computer Interface Design*. Reading: Addison-Wesley Publishing Company.

Frohlich, D.M. (1997) Direct Manipulation and Other Lessons. Em *Handbook of Human-Computer Interaction*, M.G.Helander, T.K.Landauer, P.V.Prabhu (eds) 2nd.edition, Elsevier.

Good, M. (1995) "Participatory Design of a Portable Torque-Feedback Device." Em *Human-Computer Interaction: Toward the Year 2000*, R. Baecker, J. Grudin, W. Buxton, e S. Greenberg (eds) pp 293-303. Morgan Kaufmann Publishers, inc.

Gould, J.D., Boies, S.J., Ukelson, J. (1997) How to Design Usable Systems. Em *Handbook of Human-Computer Interaction*, M.G.Helander, T.K.Landauer, P.V.Prabhu (eds.) Elsevier Science, pp. 231-254

Gould, J.D., Lewis, (1985) Designing for Usability – key principles and what designers think. *Communications of the ACM*, 28,300-311

Greenbaum, J., King, M. (1991) *Design at Work.* Hillsdale, NJ:Lawrence Erlbaum Ass.

Grudin, J. (1995) Interactive Systems: Bridging the gaps between developers and users. Em *Human-Computer Interaction: Toward the Year 2000*, R. Baecker, J. Grudin, W. Buxton, e S. Greenberg (eds) pp 293-303. Morgan Kaufmann Publishers, Inc.

Hix ,D.e Hartson, H.R.(1993) *Developing User Interfaces: Ensuring Usability through Product and Process*. New York: John Wiley.

Hutchins, E.L., Hollan, J.D., Norman, D.A. (1986) Direct Manipulation Interfaces. Em *User Centered System Design: New Perspectives on Human-Computer Interaction*, D.A. Norman, e S.W. Draper, (eds.) (1986) Hillsdale, NJ: Lawrence Erlbaum Associate Publishers.

IHC98 *Atas do I Workshop sobre Fatores Humanos em Sistemas Computacionais* http://www.inf.puc-rio.br/~ihc98

IHC99 *Atas do II Workshop sobre Fatores Humanos em Sistemas Computacionais* http://www.unicamp.br/~ihc99

Kuhn, S., Winograd, T. (1996) Participatory Design. Em *Bringing Design to Software*, Ma:Addidon-Weslley.

Laurel, B. (ed) (1990) *The Art of Human-Computer Interface Design*, Reading, Mass.:Addison-Wesley.

Lakoff, G. e Johnson, M.(1980) *Metaphors we Live By*. Chicago:University of Chicago Press.

Leite J. e Souza C.S. (1999) Uma Linguagem de Especificação para a Engenharia Semiótica de Interfaces de Usuário. Em *Atas do II Workshop sobre Fatores Humanos em Sistemas Computacionais*, Campinas, SP, p.23

Madsen, K.H. (1994) A Guide to Metaphorical Design. *Communications of the ACM* vol.37, n.12, pp. 57-62.

Martins I.H. e Souza C.S. (1998) Uma Abordagem Semiótica na Utilização dos Recursos Visuais em Linguagens de Interface. Em *Atas do I Workshop sobre Fatores Humanos em Sistemas Computacionais*, Maringá, PR, p. 38-47.

Mazzone, J. (2000). Comunicação Pessoal em Relatório Projeto Fapesp (documento Interno do NIED – Unicamp)

Muller, M. (1997) Participatory Practices in the Software Lifecycle. Em *Handbook of Human-Computer Interaction*, M.G.Helander, T.K.Landauer, P.V.Prabhu (eds.). Elsevier Science, pp. 255-297

Nadin M.(1988) Interface Design. *Semiótica*, v.69, n.3/4, p.269-302.

Nardi, B. (1997) The Use of Ethnographic Methods in Design and Evaluation. Em *Handbook of Human-Computer Interaction*, M.G.Helander, T.K.Landauer, P.V.Prabhu (eds.) Elsevier Science, pp. 361-366

Neale, D.C., Carroll, J.M. (1997) The Role of Metaphors in User Interface Design. Em *Handbook of Human-Computer Interaction*, M. Helander, T.K. Landauer e P. Prabhu (eds.) (1997) chapter 20, Elsevier Science pp. 441-462.

Nied (2000) *Jogo do Alvo*. http://www.nied.unicamp.br. Consulta em 3/4/2000.

Nielsen, J. (1992) Usability Engineering, *Computer*, March.

Nielsen, J. ,(1993) *Usability Engineering.* Boston: Academic Press.

Norman, D. (1993) *Things that Make us Smart* Reading, Ma:Addison Wesley

Norman, D. (1996) Design as Practiced. Em *Bringing Design to Software* T. Winograd (ed) Ma:Addidon-Weslley p. 233 - 251.

Norman, D.A. e Draper, S.W. (eds.) (1986) *User Centered System Design: New Perspectives on Human-Computer Interaction* Hillsdale, NJ: Lawrence Erlbaum Associate Publishers.

Oliveira, O L. e Baranauskas, M.C.C. (1998[a]) A semiótica e o Design de Software. Relatório técnico IC98-09 (disponível via Web: http://www.dcc.unicamp.br/ic-tr-ftp/1998/Titles.htm)

Oliveira, O L e Baranauskas, M C C (1998b) Semiotic Proposals for Software Design: Problems and Prospects. Relatório técnico IC98-10 (disponível via Web: http://www.dcc.unicamp.br/ic-tr-ftp/1998/Titles.htm)

Oliveira OL. e Baranauskas M.C.C. (1999) Communicating Entities: a Semiotic-Based Methodology for Interface Design. Em *Human-Computer Interaction – Ergonomics and User Interface*, H.J. Bullinger e J. Ziegler (eds) vol.1. London: Lawrence Erlbaum Associates Publishers.

Oliveira, O L (2000) *Design da Interação em Ambientes Virtuais: uma abordagem semiótica*. Tese de Doutorado. Instituto de Computação – Unicamp.

Peirce, C.S. (1974) Collected Papers of Charles Sanders Peirce. Vols. 1-6. C. Harshorme e P. Weiss (eds). Cambridge:Harvard University Press.

Prates e Souza (1999) Um Modelo de Apoio à Expressão de Projetistas de Interfaces Multiusuário. Em *Atas do II Workshop sobre Fatores Humanos em Sistemas Computacionais*, Campinas, SP, p.21

Prado e Baranauskas (1999) Projeto Granel: Investigando possibilidades da abordagem Semiótica em Design de Interfaces. Em *Atas do II Workshop sobre Fatores Humanos em Sistemas Computacionais*, Campinas, SP, p.17

Preece, J., Rogers, Sharp, H., Benyon, D., Holland, S., Carey, T. (1994) cap. 18 em *Human-Computer Interaction - Methods for User-Centred Design*, Adison Wesley, pp. 371-382.

Rossler, F (2000) *Contribuições da Semiótica ao Redesign de Interfaces para Ferramentas de Comunicação Eletrônica*. Dissertação de Mestrado. Instituto de Computação-Unicamp.

Santaella, M.L. (1996) O que é Semiótica. 12.ed São Paulo:Editora Brasiliense.

Schuler, D. e Namioka, A.(eds.) (1993) *Participatory Design: Principles and Practices*, Hillsdale,NJ: Lawrence Erlbaum Ass..

Sebeok, T.A. (1994) *Signs – An Introduction to Semiotics*. Toronto:University of Toronto Press Incorporated.

Shame (1999) *Hall of Shame*. http://www.iarchitect.com/mshame.htm. Consulta em 3/4/2000

Shneiderman, B. (1998) *Designing the User Interface*. Reading,MA: Addison-Weslley.

Shneiderman B. (1983) Direct manipulation: a step beyond programming languages, *IEEE Computer*, 16(8),57-69.

Simon (1982) *The Sciences of the Artificial*. Cambridge, MA:MIT Press.

Souza, C.S.,(1993) The Semiotic Engineering of user interface Languages, *International Journal of Man-Machine Studies*, n. 39, 753-733

Suchman, L. (1987) *Plans and Situated Actions*. Cambridge:Cambridge University Press.

Suchman, L. Trigg, R.H.(1995) Understanding Practice: Video as a Medium for Reflection and Design. Em *Human-Computer Interaction: Toward the Year 2000* R. Baecker, J. Grudin, W. Buxton, e S. Greenberg (eds) pp 293-303. Morgan Kaufmann Publishers, Inc.

Winograd, T.,(1996) *Bringing Design to Software*, Ma:Addidon-Weslley.

CAPÍTULO 4

AVALIAÇÃO DE INTERFACES

For a company to say 'we don´t need evaluation' is just the same as saying 'our system designers don´t need to see anything when they are driving: they can drive with their eyes shut and achieve the goal that they want'. You can´t possibily produce a good product blindfolded.

Brian Shackel apud Preece *et al*., 1994, p. 600

INTRODUÇÃO

Avaliação não deve ser vista como uma fase única dentro do processo de design e muito menos como uma atividade a ser feita somente no final do processo e se "der tempo". Idealmente, avaliação deve ocorrer durante o ciclo de vida do design e seus resultados utilizados para melhorias gradativas da interface. É claro que não se pode pretender efetuar extensivos testes experimentais durante todo o processo de design, mas técnicas informais e analíticas devem ser utilizadas. Nesse sentido existe uma forte correlação entre avaliação e as técnicas de modelagem e construção de protótipos discutidas no Capítulo 3, pois essas técnicas garantem que o design está constantemente sob avaliação. E na maioria de modelos de desenvolvimento de interfaces usáveis a avaliação tem um papel central, como no modelo Estrela (Hix e Hartson, 1993) apresentado no capítulo anterior (Figura 4.1).

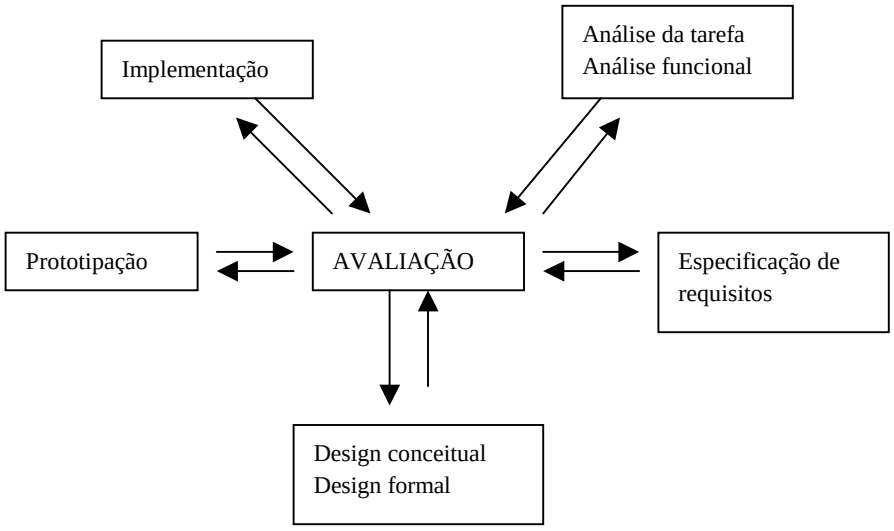


FIGURA 4.1 - O CICLO DE VIDA ESTRELA (ADAPTADO DE HIX E HARTSON, 1993)

Diferentes tipos de avaliação são necessárias em diferentes estágios do design. Nos estágios bem iniciais onde idéias estão sendo exploradas e tentadas, muitas vezes testes bastante informais são suficientes. Por exemplo, depois de uma sessão de discussão *(brainstorming)* para explorar diferentes metáforas, o conjunto inicial de opções certamente estará bem reduzido. Outras vezes, principalmente em estágios um pouco mais avançados do processo, avaliações mais formais devem ser planejadas.

Os fatores determinantes de um plano de avaliação incluem (Nielsen, 1993; Hix and Hartson, 1993; Preece *et al*., 1994; Schneiderman, 1998):

- estágio do design (inicio, meio ou fim)
- quão pioneiro é o projeto (bem definido *versus* exploratório)
- número esperado de usuários
- quão crítica é a interface (por exemplo, um sistema de controle de tráfego aéreo *versus* um sistema de orientação de um shopping)
- custo do produto e orçamento alocado para o teste
- tempo disponível
- experiência dos desingers e avaliadores

O domínio do plano de avaliação pode estar entre um ambicioso teste de dois anos, com múltiplas fases, de um novo sistema de controle de tráfego aéreo até um discreto teste de três dias com seis usuários de um sistema interno de contabilidade. Da mesma forma, o custo pode variar de 10 % até 1% do custo total do projeto.

Mesmo considerando que é uma atitude irresponsável não efetuar alguma forma de avaliação, deve-se ter claro que um certo grau de incerteza sempre permanece mesmo após exaustivos testes com múltiplos métodos. Perfeição não é possível, portanto qualquer planejamento deve prever métodos de avaliação contínua e reparo de problemas durante todo ciclo de vida de uma interface.

Deve-se ter em vista ao analisar métodos de avaliação que eles apresentam resultados bastante satisfatórios para grande parte dos sistemas em condições normais de uso, mas a performance de sistemas com grande complexidade de entradas, como por exemplo, as emergências de um sistema de controle de um reator nuclear ou de tráfego aéreo, são muito difíceis de serem testadas.
O objetivo deste capítulo não é apresentar todas as possíveis técnicas de avaliação e nem prescrever exatamente quando e como usá-las. Estaremos tratando com mais detalhe três técnicas que julgamos representativas: uma delas a mais tradicional que é o teste com usuários e as outras duas, avaliação heurística e percurso cognitivo, por serem as que apresentam mais e melhores resultados práticos, além de poderem ser aprendidas com certa facilidade. E discutiremos ao longo do capítulo as razões da existência de diferentes abordagens de avaliação.

Ressaltamos que na maioria das situações práticas uma ou mais técnicas são adotadas e adaptadas de forma a atender as necessidades específicas da interface sob avaliação. Geralmente os recursos disponíveis têm um grande impacto no tipo de avaliação a ser feita. Portanto, selecionar a técnica de avaliação adequada envolve *escolher, misturar e adaptar* técnicas a partir do conjunto de técnicas disponíveis.

OBJETIVOS DA AVALIAÇÃO

De forma geral, se faz avaliação para conhecer o que os usuários querem e os problemas que eles experimentam, pois quanto melhor informados sobre seus usuários os designers estiverem, melhor serão os design de seus produtos.

Muitas questões poderiam ser objetivo de uma avaliação. O setor de marketing poderia estar interessado em como o produto de sua empresa se compara com produtos de outros competidores do mercado. Por exemplo, se a funcionalidade e a aceitação do produto é melhor, ou pelo menos igual, à do principal competidor. Produtos também podem ser avaliados no sentido de verificar se estão de acordo com padrões específicos, como as normas ISO, por exemplo.

Avaliações são necessárias para responder dúvidas que surgem durante o processo de design e desenvolvimento de um produto. Em muitos pontos do processo de design as pessoas de desenvolvimento necessitam respostas a questões de modo a verificar se suas idéias são realmente o que os usuários necessitam ou desejam. Desse modo, avaliação direciona e se mescla ao design, auxiliando na criação de um produto útil e usável, como visto no Capítulo 3. Em contraste, podemos ter as avaliações que ocorrem depois do produto desenvolvido e que se preocupam em fazer julgamentos sobre o produto final: sua performance considerando produtos competitivos, sua adequação à uma determinada família de produtos, etc. Como um dos focos desse livro é em design estaremos neste capítulo nos atendo mais às avaliações feitas durante o processo de design e seus resultados.

Resumidamente, podemos dizer que avaliação tem três grandes objetivos: avaliar a funcionalidade do sistema, avaliar o efeito da interface junto ao usuário e identificar problemas específicos do sistema.

A **funcionalidade** do sistema é importante no sentido de estar adequada aos requisitos da tarefa do usuário, ou seja, o design do sistema deve permitir ao usuário efetuar a tarefa pretendida e de modo mais fácil e eficiente. Isso inclui não somente ter a funcionalidade adequada disponível, mas também torná-la usável, na forma de ações que o usuário precisa efetuar para executar a tarefa. Avaliação nesse nível envolve também medir a performance do usuário junto ao sistema, ou seja, avaliar a eficiência do sistema na execução da tarefa pelo usuário.

Adicionalmente, é preciso medir o **impacto do design junto ao usuário**, ou seja, avaliar sua usabilidade. Isso inclui considerar aspectos tais como: avaliar quão fácil é aprender a usar o sistema; a atitude do usuário com relação ao sistema; identificar áreas do design as quais sobrecarregam o usuário de alguma forma, por exemplo, exigindo que uma série de informações sejam relembradas; etc. Muitos dos métodos concentram a avaliação sobre aspectos padrão de usabilidade, como o uso de guidelines como apresentado no Capítulo 3.

O terceiro objetivo da avaliação é **identificar problemas específicos com o design**, ou seja, identificar aspectos do design os quais quando usados no contexto alvo, causam resultados inesperados ou confusão entre os usuários. Isso é claro está correlacionado tanto com a funcionalidade quanto com a usabilidade do design.

Atendendo a esses objetivos pode-se classificar os métodos de avaliação em duas dimensões: se usuários reais estão ou não envolvidos e se a interface está ou não implementada. Considera-se implementação qualquer protótipo executável (Whitefield *et al*., 1991). Nessas dimensões estaremos neste capítulo tratando de dois grupos de métodos:

- **inspeção de usabilidade** *(predictive evaluation)* - sem envolver usuários e podendo ser usado em qualquer fase do desenvolvimento de um sistema (implementado ou não)

- **testes de usabilidade** - métodos de avaliação centrados no usuário que incluem métodos experimentais ou empíricos, métodos observacionais e técnicas de questionamento (como nos métodos etno-gráficos vistos no Capítulo 3). Para se usar esses métodos é necessária a existência de uma implementação real do sistema em algum formato que pode ser desde uma simulação da capacidade interativa do sistema, sem nenhuma funcionalidade, um protótipo básico implementando, um cenário, ou até a implementação completa.

Outros grupos de métodos que não serão tratados neste capítulo incluem:

- **Experimentos controlados** - envolvem efetuar um bem projetado e controlado experimento de laboratório. Tem-se sempre definida uma hipótese a ser testada e todas as variáveis de interesse necessitam ser controladas. Conhecimento estatístico é necessário para validar os resultados. Controlar todas as variáveis dentro de interações complexas envolvendo humanos além de difícil é de validade muito questionável. A metodologia experimental é seguida, tendo-se o experimentador controlando certas variáveis enquanto examina outras. Os experimentos são feitos em laboratórios especialmente construídos para esse fim e existe muito rigor na observação e monitoramento do uso do sistema. Os dados coletados são analisados quantitativamente de modo a produzir métricas que guiem o design (Preece *et a*l, 1994; Dix *et al*., 1998).

- **Métodos de avaliação interpretativos** - o objetivo desse tipo de avaliação é possibilitar aos designers um maior entendimento de como os usuários se utilizam dos sistemas em seu ambiente natural e como o uso desses sistemas se integra com outras atividades. Portanto, alguma forma de participação do usuário na coleta, análise ou interpretação dos dados é bastante comum. Os métodos que pertencem a esse grupo incluem avaliação participativa e conceitual que são dois métodos desenvolvidos especialmente para avaliar IHC, e avaliação etnográfica, uma técnica emprestada da antropologia, conforme discutido no Capítulo 3. Formas de registro como vídeos e áudio podem ser feitas como em outros métodos mas a forma de análise é bastante diferenciada (Preece *et al*, 1994; Monk *et al*, 1993; Greenbaum e Kying, 1991).

INSPEÇÃO DE USABILIDADE

Define-se inspeção de usabilidade como um conjunto de métodos baseados em se ter avaliadores inspecionando ou examinando aspectos relacionados a usabilidade de uma interface de usuário. Os avaliadores podem ser especialistas em usabilidade, consultores de desenvolvimento de software, especialistas em um determinado padrão de interface, usuários finais, etc.

Diferentes métodos de inspeção têm objetivos diferentes, mas normalmente inspeção de usabilidade é proposta como um modo de avaliar design de interfaces baseado no julgamento de avaliadores e são sustentados pela confiança depositada em seus julgamentos. Os métodos variam no sentido de como os julgamentos são efetuados e em quais critérios se espera que o avaliador baseie seus julgamentos.

Pode-se contrastar os métodos de inspeção com outros modos de se obter dados de usabilidade: **automaticamente**, onde medidas de usabilidade são computadas executando-se um software de avaliação que recebe como entrada uma especificação formal da interface; **empiricamente**, testando a interface com usuários reais; **formalmente**, usando modelos exatos e fórmulas para calcular as medidas de usabilidade; e **informalmente**, usando a habilidade e experiência de avaliadores. Inspeções de usabilidade correspondem a categoria de métodos informais.

No estado atual da arte, métodos automáticos não funcionam e métodos formais são muito difíceis de serem aplicados não funcionando bem quando se tem interfaces complexas e altamente interativas (Kahan e Prail, 1994).

Métodos empíricos ou testes de usabilidade são o principal modo de avaliar interfaces e certamente o mais tradicional (estaremos discutindo mais profundamente no decorrer deste capítulo). Mas geralmente, usuários reais são difíceis e caros para serem recrutados de forma a se poder testar todas as fases do desenvolvimento evolutivo de uma interface. Muitos estudos demonstram que muitos problemas encontrados por métodos de inspeção não são detectados com testes de usuários e vice-versa. Esses estudos sugerem que os melhores resultados são obtidos combinando testes com usuários e inspeções.

OBJETIVOS DA INSPEÇÃO

Inspeção de usabilidade objetiva encontrar problemas de usabilidade em um design de uma interface de usuário e com base nesses problemas fazer recomendações no sentido de eliminar os problemas e melhorar a usabilidade do design. Isso significa que inspeções de usabilidade são feitas em um estágio onde a interface está sendo gerada e a sua usabilidade (e utilidade) necessita ser avaliada

Problemas de usabilidade podem ser definidos como aspectos da interface do usuário que podem causar uma usabilidade reduzida ao usuário final do sistema.

Usabilidade, como já visto em capítulos anteriores, é um termo bastante amplo que se refere a quão fácil é para o usuário final aprender a usar o sistema, quão eficientemente ele irá utilizar o sistema assim que aprenda como usar e quão agradável é o seu uso. Também, a frequência e a severidade dos erros do usuário são consideradas como partes constituintes da usabilidade. Entretando, um usuário pode achar um elemento da interface problemático por muitas razões: torna o sistema difícil de aprender, torna-o lento na execução de suas tarefas, causa erros de uso, ou pode ser simplesmente feio e desagradável. Muito do trabalho da inspeção de usabilidade é classificar e contar o número de problemas de usabilidade. Esta análise depende da exata definição do que é um problema de usabilidade e julgamentos de como diferentes fenômenos constituem manifestações de um único problema. Freqüentemente, é muito difícil fazer essas distinções, mas na maioria dos casos bom senso é suficiente para determinar o que é um problema de usabilidade. Para uma definição geral de problema de usabilidade, pode-se dizer que é qualquer aspecto de um design onde uma mudança pode melhorar uma ou mais medidas de usabilidade.

A identificação dos problemas na interface é importante, mas é apenas uma parte de um grande processo. Depois de ser gerada a lista de problemas de usabilidade, a equipe de desenvolvimento precisa fazer um redesign da interface de modo a tentar corrigir a maior quantidade possível de problemas. Para que isso seja feito serão precisos outros tipos de informações e análises dentro do contexto mais amplo da engenharia de usabilidade.

Métodos de inspeção de usabilidade são geralmente melhores na deteção de problemas do que na direção de como melhorar a interface, mas tipicamente relatórios gerados a partir dos métodos contêm sugestões para redesign. Em muitos casos, conhecendo sobre o problema de usabilidade, é clara a maneira de corrigí-lo. Além disso, muitos dos métodos sugerem encontros entre a equipe de avaliadores e a equipe de desenvolvimento, quando esta é distinta, para discutir soluções de redesign.
O uso efetivo de uma lista de problemas de usabilidade irá requerer que esses problemas sejam priorizados com relação à gravidade de cada problema. Prioridades são necessárias para não se dispender esforços desproporcionais corrigindo problemas que não irão alterar em muito a interação do usuário com a interface. **Graus de severidade** são geralmente derivados do impacto gerado pelo problema tanto no usuário quanto no mercado. Por serem muitas vezes critérios dependentes da aplicação, a definição de graus de severidade não é muito bem estabelecida na literatura, mas alguns autores dão exemplos de como atribuir graus de severidade à problemas de usabilidade (Nielsen, 1994; Karat, 1994; Desurvire, 1994).

Precisa ser estimado o custo associado à implementação das sugestões de redesign. Certamente, problemas de usabilidade com alto grau de severidade devem ser corrigidos não interessando o quanto custem. Freqüentemente, muitos dos problemas não muito graves podem ser corrigidos com alterações mínimas de código. Esse compromisso não pode ser considerado como parte do método de inspeção de

usabilidade, pois é preferível ter uma relação completa de todos os problemas sem o pré julgamento da viabilidade ou não da sua correção.

Concluindo, ressaltamos que métodos de inspeção podem ser aplicados em fases iniciais ou finais do design e o resultado é um relatorio formal dos problemas identificados com recomendações para mudanças. Alternativamente, e fortemente recomendável, a inspeção pode resultar em uma discussão ou apresentação para os designers e gerentes do projeto. Avaliadores precisam estar sensíveis à habilidade profissional e ao alto grau de envolvimento da equipe de desenvolvimento, e as sugestões devem ser feitas com cuidado: é difícil para alguém que tem um contato inicial com o sistema efetuando uma inspeção entender todas as razões de design e história de desenvolvimento de uma interface. Daí os revisores anotarem os possíveis problemas para discussão, mas as decisões de como corrigir devem ser deixadas sob responsabilidade dos designers.

Uma inspeção pode demorar de 12hs até 40hs, dependendo da complexidade da interface e de seus procedimentos operacionais, além do contexto da tarefa que necessariamente deverá ser conhecido pelo avaliador.

MÉTODOS DE INSPEÇÃO

Dentre os diversos métodos de inspeção existentes podemos destacar:

- **Avaliação Heurística:** é feita a inspeção da interface tendo como base uma pequena lista de heurísticas de usabilidade. Esse método será discutido com mais detalhe na próxima seção deste capítulo.

- **Revisão de *Guidelines***: a interface é analisada no sentido de verificar se está de acordo com uma lista de *guidelines* de usabilidade. Geralmente essa lista contém uma seqüência de cerca de 1.000 *guidelines*, o que torna o uso desse método muito raro dada a expertise que é exigida de um revisor.

- **Inspeção de Consistência**: o avaliador verifica a consistência dentro de uma família de interfaces, quanto à terminologia, cores, *layout*, formatos de entrada e saída, e tudo o mais dentro da interface. Também é avaliado o material *online* de treinamento e de ajuda .

- **Percurso Cognitivo:** o avaliador simula o usuário "caminhando" na interface para executar tarefas típicas. Tarefas mais freqüentes são o ponto inicial de análise, mas tarefas críticas, tais como recuperação de erro, também são percorridas. Percurso cognitivo foi desenvolvido para interfaces que podem ser aprendidas de forma exploratória, mas também são úteis em interfaces que

exigem muito treinamento. Esse método será discutido em seção próxima deste capítulo.

Segundo Nielsen (1993), os métodos de inspeção de usabilidade não exigem muito esforço de quem pretende usá-los e podem ser facilmente integrados aos mais variados esquemas de produção de software. Não é necessário modificar fundamentalmente o modo como sistemas são desenvolvidos e gerenciados de modo a obter grandes benefícios da inspeção de usabilidade. Os resultados são rápidos e fornecem concretas evidências de quais aspectos da interface devem ser aperfeiçoados.

Percurso cognitivo e avaliação heurística são os precursores dos métodos de inspeção de usabilidade e geralmente não exigem uma grande experiência ou longo treinamento para que possam ser utilizados. Além disso, a utilização deles proporciona uma relevante experiência educacional para designers novatos. A exposição a preocupações adicionais de usabilidade tem se mostrado um meio efetivo de incrementar a perspectiva e valor do desenvolvimento de software orientado para o usuário. Esses dois métodos estarão sendo apresentados com detalhe nas próximas seções deste capítulo.

AVALIAÇÃO HEURÍSTICA

A maioria dos métodos de inspeção terão um efeito significativo na interface final somente se forem usados durante o ciclo de vida do projeto. Infelizmente, isso ainda não é uma realidade. Muitos desenvolvedores consideram os métodos intimidadores, muito caros, difíceis e que necessitam muito tempo para serem aplicados (Nielsen, 1994). No sentido de inverter essa tendência Nielsen propõe a denominada engenharia econômica de usabilidade - *discount usability engineering* - (Nielsen, 1989; Nielsen, 1993) com métodos que são baratos, rápidos e fáceis de serem usados. Avaliação heurística é o principal método dessa proposta. Segundo o autor, é fácil (pode ser ensinada em 4hs); é rápida (cerca de 1 dia para a maioria das avaliações); e tão barata quanto se deseje.

COMO CONDUZIR UMA AVALIAÇÃO HEURÍSTICA

Deve ser vista como parte do processo de design interativo de uma interface. Ela envolve um pequeno conjunto de avaliadores examinando a interface e julgando suas características em face de reconhecidos princípios de usabilidade, denominados heurísticas.

De modo geral, é difícil de ser feita por um único avaliador, porque uma única pessoa nunca é capaz de encontrar todos os problemas de usabilidade de uma interface. A experiência tem mostrado que diferentes pessoas encontram diferentes problemas, e portanto se melhora significativamente os resultados da avaliação heurística utilizando múltiplos avaliadores. A recomendação é que se use de três a cinco avaliadores.

A avaliação heurística é feita em um primeiro momento individualmente. Durante a sessão de avaliação cada avaliador percorre a interface diversas vezes (pelo menos duas) inspecionando os diferentes componentes do diálogo e ao detetar problemas os relata associando-os claramente com as heurísticas de usabilidade que foram violadas. As heurísticas (Tabela 4.1 e Tabela 4.2), são regras gerais que objetivam descrever propriedades comuns de interfaces usáveis (Nielsen, 1994).

1. **Diálogo simples e natural**
   - simples significa informação não irrelevante ou raramente utilizada
   - natural refere-se à adequação à tarefa
2. **Falar na linguagem do usuário**
   - usar conceitos do mundo do usuário
   - não usar termos computacionais específicos
3. **Minimizar a carga de memória do usuário**
   - não fazer com que o usuário tenha que relembrar coisas de uma ação em uma próxima ação
   - deixar informação na tela até ela não ser mais necessária
4. **Ser consistente**
   - seqüência de ações aprendidas em uma parte do sistema devem poder ser aplicadas em outras partes
5. **Prover feedback**
   - dar conhecimento aos usuários do efeito que suas ações têm sobre o sistema
6. **Saídas claramente marcadas**
   - se o usuário entra em uma parte do sistema que não lhe interessa, ele deve ser capaz de sair rapidamente sem estragar nada
   - não colocar o usuário em armadilhas
7. **Prover *Shortcuts***
   - auxiliar o usuário experiente a evitar extensos diálogos e mensagens de informações que ele não quer ler
8. **Mensagens de erro construtivas e precisas**
   - informar ao usuário qual foi o problema e como corrigí-lo
9. **Prevenir erros**
   - sempre que encontrar uma mensagem de erro, verificar se aquele erro poderia ser evitado

TABELA 4.1 - LISTA ORIGINAL DE HEURÍSTICAS DE USABILIDADE (NIELSEN, 1990)

1. **Visibilidade do status do sistema**
   - sistema precisa manter os usuários informados sobre o que está acontecendo, fornecendo um feedback adequado dentro de um tempo razoável
2. **Compatibilidade do sistema com o mundo real**
   - sistema precisa falar a linguagem do usuário, com palavras, frases e conceitos familiares ao usuário, ao invés de termos orientados ao sistema. Seguir convenções do mundo real, fazendo com que a informação apareça numa ordem natural e lógica
3. **Controle do usuário e liberdade**
   - usuários frequentemente escolhem por engano funções do sistema e precisam ter claras saidas de emergência para sair do estado indesejado sem ter que percorrer um extenso diálogo. Prover funções *undo* e *redo*.
4. **Consistência e padrões**
   - usuários não precisam adivinhar que diferentes palavras, situações ou ações significam a mesma coisa. Seguir convenções de plataforma computacional
5. **Prevenção de erros**
   - melhor que uma boa mensagem de erro é um design cuidadoso o qual previne o erro antes dele acontecer
6. **Reconhecimento ao invés de relembrança**
   - tornar objetos, ações e opções visíveis. O usuário não deve ter que lembrar informação de uma para outra parte do diálogo. Instruções para uso do sistema devem estar visíveis e facilmente recuperáveis quando necessário
7. **Flexibilidade e eficiência de uso**
   - usuários novatos se tornam peritos com o uso. Prover aceleradores de formar a aumentar a velocidade da interação. Permitir a usuários experientes "cortar caminho" em ações freqüentes.
8. **Estética e design minimalista**
   - diálogos não devem conter informação irrelevante ou raramente necessária. Qualquer unidade de informação extra no diálogo irá competir com unidades relevantes de informação e diminuir sua visibilidade relativa
9. **Ajudar os usuários a reconhecer, diagnosticar e corrigir erros**
   - mensagens de erro devem ser expressas em linguagem clara (sem códigos) indicando precisamente o problema e construtivamente sugerindo uma solução.
10. **Help e documentação**
    - embora seja melhor um sistema que possa ser usado sem documentação, é necessário prover *help* e documentação. Essas informações devem ser fáceis de encontrar, focalizadas na tarefa do usuário e não muito extensas.

TABELA 4.2 - VERSÃO REVISADA DAS HEURÍSTICAS (NIELSEN, 1993)

Depois dessa etapa inicial, as listas de problemas dos avaliadores são consolidadas em uma só. Tipicamente uma sessão de avaliação, em sua etapa individual, dura cerca de duas horas. Sessões mais extensas de avaliação podem ser necessárias para o caso de interfaces muito grandes ou muito complexas, com um substancial número de componentes de diálogo. Nesse caso, é recomendável dividir a avaliação em pequenas sessões, cada qual avaliando um cenário específico de interação.

Adicionalmente ao conjunto de heurísticas gerais a ser considerada em cada componente do diálogo, o avaliador pode também considerar heurísticas específicas da categoria do produto sendo analisado, por exemplo, heurísticas derivadas da análise de produtos similares e seus resultados de uso.

Por exemplo, um sistema altamente dependente de menus ou com diálogo centrado na entrada de informações via formulários pode definir heurísticas que avaliem especificamente a usabilidade desses componentes mais relevantes. Um exemplo, é o mencionado por Romani e Baranauskas (1998) quando apresentam o resultado da avaliação heurística de um sistema denominado *Sistema de Acompanhamento e Avaliação de Rebanhos Leiteiros (ProLeite)*, que é usado na organização das informações de desempenho produtivo e reprodutivo dos animais de rebanhos leiteiros. O sistema possui grande quantidade de formulários para entrada de dados, possibilita consultas e emissão de relatórios. A partir dos resultados da avaliação heurística do sistema é mostrada a necessidade de um conjunto de heurísticas de categorias específicas, para captar aspectos específicos do domínio considerado.

1.	Opções de menu significativas e agrupadas logicamente: O agrupamento e os nomes das opções do menu são pistas para o usuário encontrar a opção requerida.
2.	Facilidade no modo de operação: O sistema deve prover um modo de operação facilitado, principalmente para sistemas de entrada de dados. Tais sistemas são construídos a base de formulários de entrada de dados sendo importante minimizar a quantidade de toques do usuário.
3.	Agrupamento lógico e seqüencial dos campos: O sistema deve dispor os campos de forma lógica e seqüencial nos formulários. O agrupamento lógico facilita a entrada de dados tanto para usuários iniciantes quanto experientes.
4.	Diferenciação entre campos não editáveis, obrigatórios e opcionais: O sistema deve prover cor de fundo diferenciada para campos não editáveis sinalizando para o usuário que determinados campos em um formulário não precisam ser digitados. O sistema deve fornecer alguma forma de identificação para campos obrigatórios para auxiliar na entrada de dados pois evidencia quais campos devem ser preenchidos e quais podem ficar em branco. Facilita a entrada dos dados e acaba evitando erros.
5.	Permite identificação do tipo de dado e quantidade de caracteres: No sistema deve

	estar em evidência para o usuário o tipo de dado para cada campo, através de mensagem explicativa (hint), formatação do campo (por ex. --/--/---- para datas) ou no próprio rótulo. O sistema deve alertar sobre a quantidade de caracteres possível por campo.
6.	Agilidade na movimentação do cursor: O sistema deve facilitar a mudança de um campo para outro nos formulários através de teclas de atalho além do mouse, como o TAB ou ENTER.
7.	Facilidade na correção de erros durante a entrada de dado: O sistema deve permitir a correção rápida de erros durante a entrada de dados através de teclas como DEL, BACKSPACE ou outra. Além disso, ele deve possibilitar overtyping nos campos e a remoção de um campo inteiro.
8.	Aproveitamento de dados entrados anteriormente: Para sistemas de entrada de dados, é extremamente importante o aproveitamento de dados na digitação de novos registros. Esta característica diminui consideravelmente a quantidade de toques por registro.
9.	Localização de informação rapidamente: O sistema deve permitir a localização de um registro específico durante a entrada do dado. Esta opção de localização deve possibilitar buscas elaboradas não apenas por um código ou chave que identifique o registro na base de dados.

TABELA 4.3 – HEURÍSTICAS ESPECÍFICAS (ROMANI E BARANAUSKAS, 1998)

Dentre as heurísticas específicas definidas listamos algumas na Tabela 4.3. Como pode ser observado as heurísticas específicas são um refinamento das heurísticas gerais, provendo somente meios de uma avaliação mais específicas dos componentes de um diálogo.

Como durante a avaliação heurística os avaliadores não estarão usando o sistema de fato, ou seja, para efetuar tarefas reais, é possível efetuar a avaliação em interfaces que existam apenas em papel e que ainda não foram implementadas.

Se o sistema é de domínio não específico ou para ser usado pela população em geral ou os avaliadores são peritos no domínio, nenhuma assistência adicional é necessária. Caso contrário, será preciso prover meios de auxiliar o avaliador de forma a que ele seja capaz de usar o sistema adequadamente. Uma forma é ter sempre uma pessoa da equipe de desenvolvimento disponível para responder perguntas dos avaliadores. Outra maneira, é prover cenários típicos de uso (ver Capítulo 3), listando os vários passos que um usuário deveria efetuar para realizar um conjunto de tarefas reais. Tal cenário deve ser construído com base na análise da tarefa dos usuários reais de modo a ser tão realístico quanto possivel. Esta é uma abordagem usada com freqüência e com bastante sucesso.

Como dissemos, o resultado de uma avaliação heurística é uma lista de problemas de usabilidade da interface com referências aos princípios de usabilidade que foram violados. O avaliador não pode simplesmente dizer que não gosta de um determinado aspecto, tem que justificar com base nas heurísticas e tem também que ser o mais específico possível e listar cada problema encontrado separadamente. Geralmente a avaliação heurística não objetiva prover meios de corrigir os problemas ou um modo de avaliar a qualidade de um redesign. Entretanto, como ela explica cada problema encontrado referenciando as respectivas heurísticas que foram violadas, geralmente não é difícil gerar um design revisado baseado nas diretrizes que foram providas pelo príncipio de usabilidade violado.

EXEMPLOS DE PROBLEMAS ENCONTRADOS NA AVALIAÇÃO HEURÍSTICA

Estaremos apresentando alguns problemas isolados detetatos em avaliações heurísticas, de forma a deixar mais claro o tipo de resultados que podem ser obtidos pelo método. Importante observar que muitas vezes um único componente de diálogo fere mais que uma heurística e isso será indicado em nossos exemplos.

Exemplo 1

Em um sistema para oferecimento de cursos a distância denominado TelEduc[1], a ferramenta de correio eletrônico não permite o envio de mensagens sem o preenchimento do campo assunto (*subject*). Essa característica é uma fonte de erros constante, pois a maioria dos sistemas de correio permite isso, emitindo apenas um aviso. A heurística violada é a de **prevenção de erros.** Também deve ser observada a mensagem de erro inadequada - **Você não informou o assunto da correspondência** - usando o termo correspondência que não é natural no contexto de mensagens (Figura 4.2). Heurística violada - **consistência e padrões**. A recuperação do erro também é difícil, pois ao retornar à tela de composição da mensagem ela tem que ser completamente refeita. Portanto, mais uma heurística violada - **ajudar os usuários a reconhecer, diagnosticar e corrigir erros** .

Exemplo 2

Outro exemplo, extraído do componente **Lista de Discussão** da mesma ferramenta de ensino a distância do exemplo anterior. No sistema TelEduc as telas têm sempre o mesmo formato: dividida em dois *frames*, o da esquerda sempre está presente e contém a relação de todas as ferramentas disponíveis no ambiente e o da direita muda conforme a ferramenta escolhida. Quando se seleciona Lista de Discussão, a direita aparece uma

[1] http://www.nied.unicamp.br/tele_educ

nova tela onde se tem todas as listas abertas e as operações possíveis com elas (Figura 4.3).

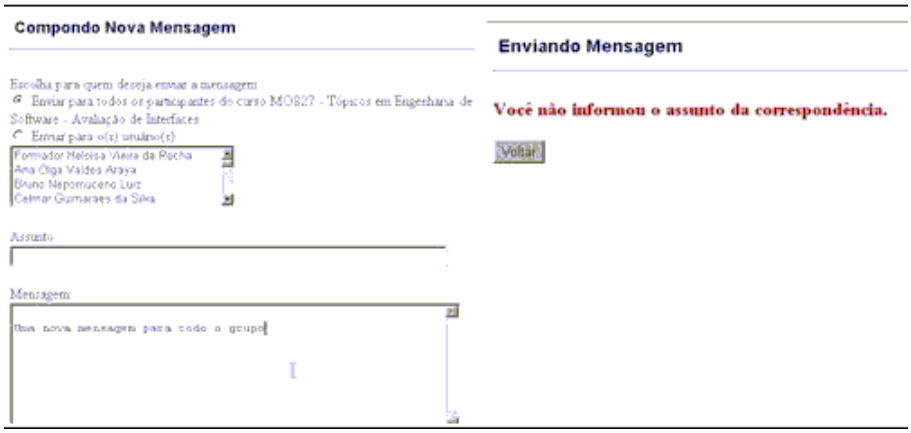


FIGURA 4.2 – QUEBRA DE USABILIDADE - **EXEMPLO 1** (TELEDUC – CORREIO)

FIGURA 4.3 – QUEBRA DE USABILIDADE - **EXEMPLO 2** (TELEDUC-GRUPOS DE DISCUSSÃO)

Ao se selecionar a opção Ver que possibilita ler as mensagens de uma determina lista a tela é substituída e se tem acesso às mensagens trocadas. E nessa tela temos uma coleção de heurísticas não respeitadas, algumas delas:

- **Estética e design minimalista; consistência e padrão** - existem dois componentes que têm exatamente a mesma função. Um é a flecha no topo direito da tela e outro o botão em baixo a direita. Um deles deve ser eliminado. A flecha tem o problema adicional de confundir com a flecha de *back* da maioria dos *browsers* (**prevenção de erros**). E o botão Voltar tem que ser mais específico indicando para onde vai voltar (ou então pode-se pensar que simplesmente é um *back* análogo ao do *browser*, o que também é ocasiona a quebra da heurística **prevenção de erros**).

- **Compatibilidade do sistema com o mundo real** - as mensagens podem ser vistas em diversas ordens (cronológica, alfabética por remetente, etc.) mas dificilmente um usuário não especialista vai saber o que é em *ordem de árvore* com relação ao título da mensagem

- **Reconhecimento ao invés de relembrança; consistência e padrão**- a pergunta mais freqüente dos usuários é em como fazer para alterar a ordem das mensagens. Dificilmente se apercebem que basta um clique no rótulo do campo da mensagem que a ordem é alterada. Deveria ser provida uma forma que tornasse visível essa funcionalidade. Também importante que outras ferramentas análogas (como o correio exemplificado anteriormente) consistentemente apresentassem a mesma funcionalidade (pois depois de aprendida na lista, a funcionalidade vira fonte de erro dada a inconsistência entre as ferramentas análogas).

Exemplo 3

O antivírus Norton 2000 para NT Server é um software projetado para proteger servidores de rede Windows NT de arquivos infectados por vírus. O software é executado no servidor NT e sempre que se tenta gravar/abrir algum arquivo infectado no servidor, o programa apresenta uma mensagem na tela do servidor avisando que o arquivo está infectado, mas os usuários em estações cliente NT não recebem esse aviso. Tem-se portanto a heurística **visibilidade do status do sistema** violada. Consequentemente, quando o usuário trabalhando em uma estação cliente tenta gravar um arquivo infectado no servidor o antivirus impede a gravação e não emite nenhuma mensagem para a estação cliente e o usuário pode então perder seu trabalho sem saber que isso está ocorrendo. O resultado é perda do arquivo, o que teria sido facilmente prevenido se alguma mensagem de alerta fosse enviada à estação cliente. Claramente a **heurística ajudar os usuários a reconhecer, diagnosticar e corrigir erros** também é violada.

Exemplo 4

No sistema Windows quando se quer instalar um novo componente de hardware é inciado um processo de busca e o indicador de detecção pode ficar parado por muito tempo, como indicado na janela de diálogo (Figura 4.4). O usuário fica perdido na maioria das vezes por não saber o que significa esse muito tempo e não sabe se deve ou não reiniciar o computador, mesmo porque usuários de sistemas semelhantes sabem quão pouco confiável é a relação da barra de deteção com o real andamento da operação (**visibilidade e status do sistema; ajudar os usuários a reconhecer, diagnosticar e corrigir erros**).

FIGURA 4.4 - DETECÇÃO DE HARDWARE DO SISTEMA WINDOWS

Exemplo 5

No Dia das Crianças o provedor ZAZ (http://www.zaz.com.br) fez uma página especial que dava entrada para uma página de jogos (Figura 4.5). Nessa página uma coleção de termos técnicos é encontrada e as crianças, potencialmente usuários novatos, não conseguem entender o que tem que fazer para atingir seu objetivo, que é jogar (**compatibilidade do sistema com o mundo real**).

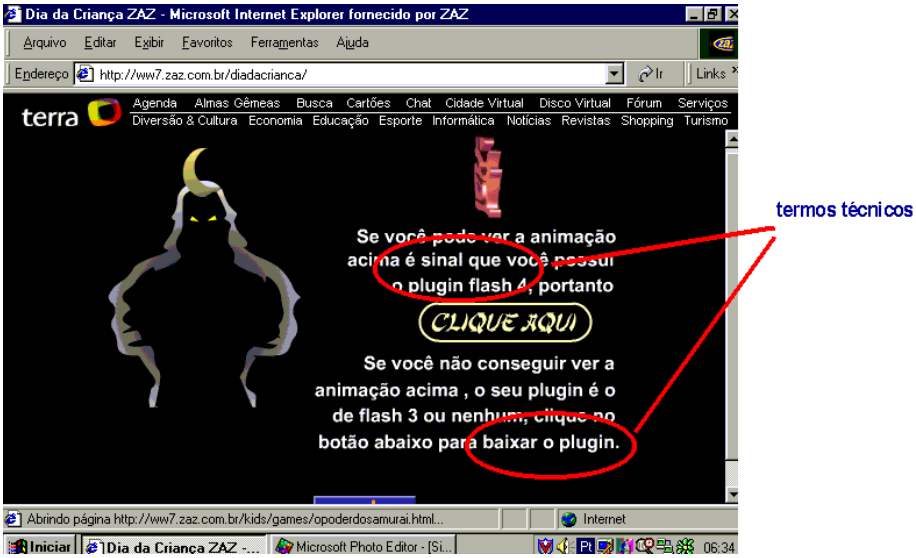


FIGURA 4.5 - SITE DO PORTAL ZAZ NO DIA DA CRIANÇA DE 1999

Exemplo 6

O software Winzip em uma mesma versão gratuita (não registrada) tem uma tela de abertura onde os botões aparecem em ordem aleatória a cada execução (Figura 4.6). Arbitrariamente os botões Accept e Quit aparecem em ordem trocada levando o usuário a errar sem saber porque (**consistência e padrões; prevenção de erros**).

FIGURA 4.6 - TELA DE ABERTURA DA CÓPIA PARA AVALIAÇÃO DO SOFTWARE WINZIP

Exemplo 7

No sistema de entregas da declaração de Imposto de Renda ao tentar desistir do programa de instalação o usuário recebe uma caixa de diálogo onde o Não está a direita do Sim (Figura 4.7). Isso foge completamente ao padrão de diálogo de toda aplicação Windows em caixas de diálogo do tipo Sim/Não sendo fonte constante de erro (**consistência e padrões; prevenção de erros**). Vale a pena observar que esse programa é da categoria de softwares de uso eventual e mesmo que internamente esteja sendo adotado o padrão de colocar a escolha “mais provável” em primeiro lugar, o padrão Windows deveria ter sido respeitado, pois o usuário não irá usar o software

muito tempo de forma a poder aprender esse padrão interno específico. Outro aspecto a ser observado é o uso indevido da palavra abortar na caixa de diálogo (**compatibilidade do sistema com o mundo real**).

FIGURA 4.6 - PÁGINA DE INSTALAÇÃO DO SOFTWARE DE ENVIO ELETRÔNICO DA DECLAÇÃO DE IMPOSTO DE RENDA

Exemplo 8

No Windows Explorer ao tentarmos excluir um arquivo que está em uso uma caixa de diálogo é aberta (Figura 4.7). Nessa caixa aparece a mensagem de que não foi possivel acessar o arquivo, e recomenda ao usuário que verifique se o disco está cheio ou protegido, e finalmente se o arquivo não está sendo usado. Não há usuário que não se confunda: o que tem a ver disco cheio com excluir um arquivo! (**ajudar os usuários a reconhecer, diagnosticar e corrigir erros**)

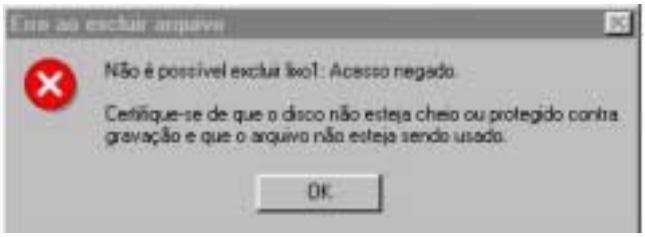


FIGURA 4. 7 MENSAGEM DE ERRO DO WINDOWS EXPLORER

Exemplo 9

A página de entrada do site do AltaVista[2] tem uma enorme redundância de links para a função de busca (Figura 4.8). O usuário efetivamente não sabe qual usar, quando na verdade todas conduzem ao mesmo resultado (**estética e design minimalista**).

FIGURA 4.8 - PÁGINA DE ENTRADA DO SITE ALTA VISTA EM 10/04/2000

Exemplo 10

O Microsoft Word possui, no canto inferior esquerdo da tela, 4 botões, que servem para selecionar o modo de exibição do texto: normal, *layout online*, *layout* da página, e estrutura de tópicos (Figura 4.9).

Através desses botões o usuário pode alterar facilmente o modo de exibição de seu documento. Mas quando o modo de *layout online* é selecionado, os botões desaparecem, e o usuário fica sem saber como retornar ao modo de exibição anterior (Figura 4.10). O retorno só pode ser feito indo no menu Exibir e selecionando um dos outros modos (**reconhecimento ao invés de relembrança**)

[2] http://www.altavista.com

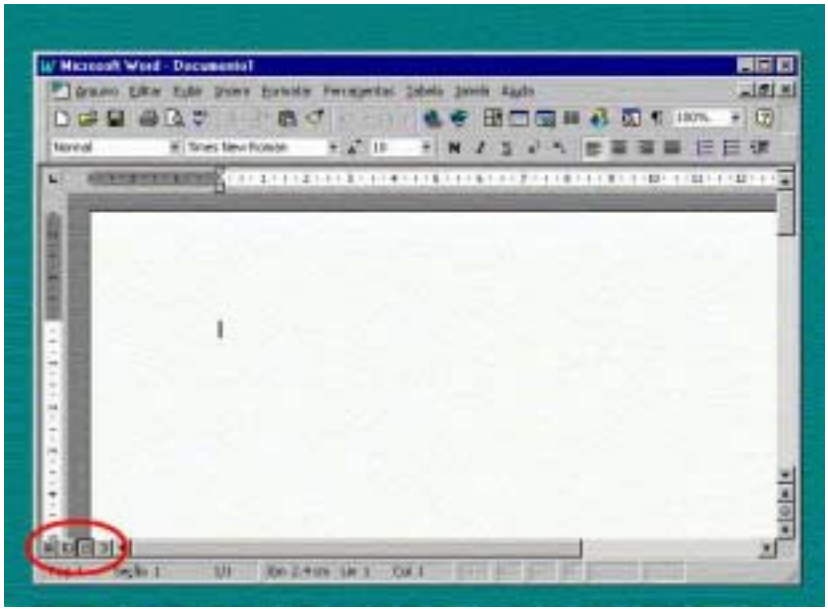


FIGURA 4.9 - TELA DO EDITOR DE TEXTOS WORD PARA WINDOWS

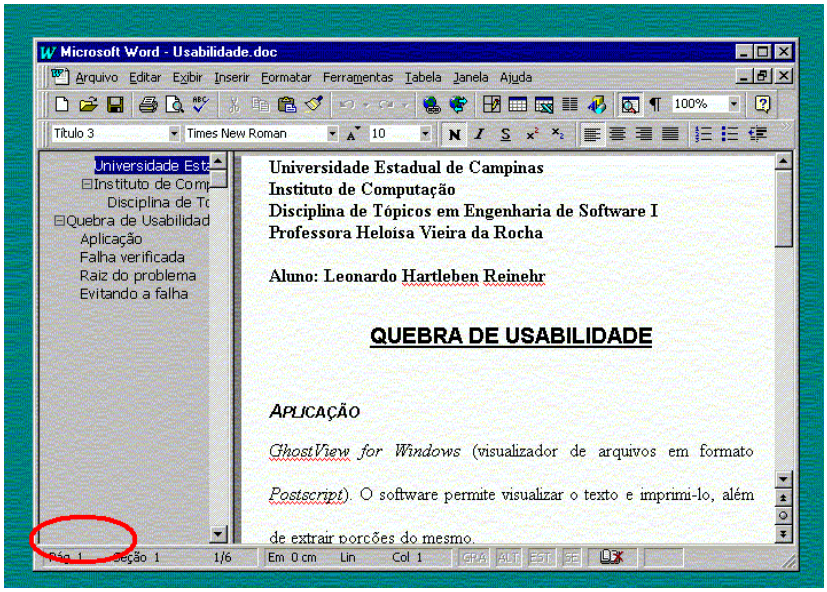


FIGURA 4.10 - TELA DO EDITOR DE TEXTOS WORD PARA WINDOWS MOSTRANDO TEXTO NO FORMATO *LAYOUT*

Exemplo 11

No Editor do Netscape quando se está editando um documento html, constantemente é necessária a visualização do documento no *browser*, para se testar links ou visualizar a forma do documento. O mesmo ocorre quando se está no *browser* e se deseja editar o documento (Figura 4.11). Portanto estas duas opções são bastante usadas por desenvolvedores de páginas, logo sempre deveriam ser oferecidos atalhos para essas opções (a partir do *browser* e a partir do editor), o que não ocorre no produto em questão (**flexibilidade e eficiência de uso**)

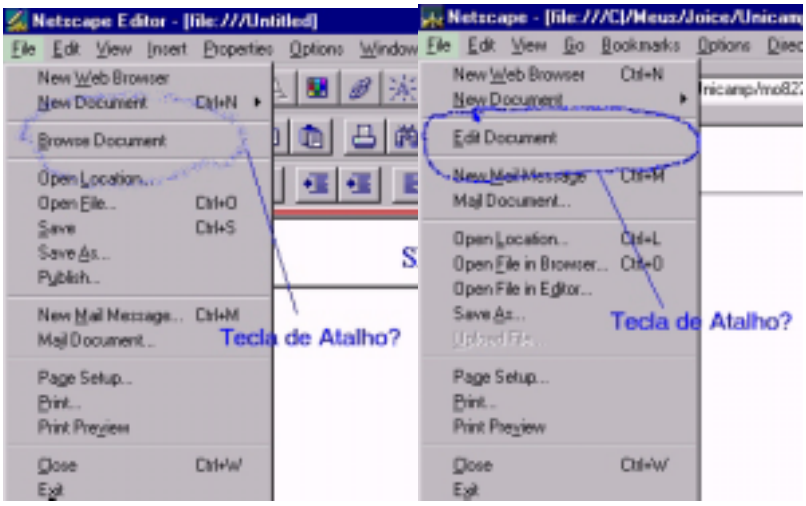


FIGURA 4.11 - TELAS DO BROWSER NETSCAPE COM MENUS DO MODO COMPOSER E NAVIGATOR

Exemplo 12

O Help do sistema Netscape não é muito extenso (Figura 4.12). Ele não preserva nenhum contexto, ou seja, sempre é aberta a mesma tela, independente do usuários estar no Navigator ou Mail, por exemplo (**help e documentação**). No índice à esquerda da tela, o usuário não sabe quando deve dar um clique na palavra ou na bola (que é o link) ao lado da palavra (três primeiras palavras da tela), ou seja, o usuário não identifica onde está o link, inclusive na parte de baixo da tela as palavras é que são os links (**consistência e padrões; prevenção de erros**). Ainda nessa tela tem-se a opção *Find* e a opção *Look* For que têm exatamente a mesma funcionalidade. O mesmo acontece com o X na barra em baixo da tela cuja funcionalidade é a mesma do X na barra inicial (padrão para fechar janelas). Isso confunde o usuário achando que o X em baixo é para sair de tudo e não simplesmente fechar a janela(**estética e design minimalista**).

FIGURA 4.12 - SISTEMA DE *HELP* DO *BROWSER* NETSCAPE

Com esses exemplos, podemos de forma mais clara perceber que o dignóstico de um problema associado com as heurísticas que foram consideradas fonte do problema efetivamente traz possibilidades concretas de redesign, apesar desse não ser o objetivo da avaliação.

Como dissemos, uma possibilidade para estender o método de avaliação heurística, de forma a prover efetivas soluções de redesign, é fazer uma sessão de discussão final envolvendo a equipe de avaliadores e representantes da equipe de desenvolvimento. Essa discussão não precisa ser formalmente conduzida e deve estar focalizada nos principais problemas de usabilidade. Outro ponto que fortalece a existência dessa reunião é que nela podem ser levantados os aspectos positivos do design, pois a avaliação heurística não trata desse importante aspecto. Aspectos positivos são todas as características importantes e boas em uma interface que de forma alguma deveriam ser alteradas ou eliminadas em um redesign.

GRAUS DE SEVERIDADE

Adicionalmente à lista de problemas de usabilidade detetados, a avaliação heurística pode ser usada para avaliar a gravidade de cada problema. Esta informação é importante no momento em que forem alocados recursos para corrigir os problemas mais sérios e se necessário deixar os menos graves para uma nova versão.

A gravidade de um problema é a combinação de três fatores:

- **a frequência** com que ele ocorre: se é comum ou raro
- **impacto** do problema quando ele ocorre: se é fácil ou difícil para o usuário superá-lo
- **a persistência** do problema: problema que ocorre uma única vez e que o usuário pode superar desde que saiba que ele existe, ou se os usuários serão repetidamente incomodados por ele

Finalmente, é claro que é preciso considerar o **impacto do problema no mercado**, pois muitos problemas simples de serem superados tem um efeito importante na popularidade de um produto.

Dada essa diversidade de componentes, usualmente se faz uma tabela onde todos aparecem combinados de modo bastante subjetivo como pode ser visto na Tabela 4.4.

1. eu não concordo que isso é um problema de usabilidade
2. é um problema cosmético somente - precisa ser corrigido somente se sobrar algum tempo no projeto
3. problema de usabilidade menor - corrigí-lo deve ter prioridade baixa
4. problema de usabilidade grave - importante corrigí-lo, deve ser dada alta prioridade
5. catástrofe de usabilidade - a sua correção é imperativa antes do produto ser liberado

TABELA 4.4 - LISTA DE GRAUS DE SEVERIDADE DE PROBLEMAS ENCONTRADOS EM UMA AVALIAÇÃO HEURÍSTICA (ADAPTADO DE NIELSEN, 1994, P. 49)

Dificilmente se terá uma tabela de valores que expresse de maneira objetiva os componentes de gravidade de um problema para todos os sistemas. Elas são construídas quase que caso-a-caso e será a experiência dos avaliadores que irá determinar a coerência na atribuição de valores. Nielsen (1994) apresenta uma série de resultados procurando certificar a confiança nos julgamentos de severidade. Ele conclui que a taxação de severidade feita por um único avaliador não é confiável. Quanto mais avaliadores julgam a gravidade de problemas a qualidade cresce rapidamente, e taxação feita por três ou quatro avaliadores é satisfatória para a maioria dos problemas práticos.

Como a atribuição de graus de severidade não é possível de ser feita enquanto não se tem uma relação completa dos problemas, depois da sessão de avaliação é apresentado ao conjunto de avaliadores uma lista de graus de severidade e a relação completa de todos os problemas encontrados e cada avaliador atribui individualmente graus de severidade aos problemas.

CARACTERÍSTICAS DE PROBLEMAS DE USABILIDADE ENCONTRADOS PELA AVALIAÇÃO HEURÍSTICA

Avaliação heurística tem se mostrado um bom método para determinar tanto problemas graves como problemas menores de usabilidade. Certamente os problemas mais sérios são mais fáceis e os mais importantes de serem identificados, mas um grande valor da avaliação heurística está na detecção dos chamados problemas menores - como inconsistência tipográfica entre componentes do diálogo - que dificilmente seriam detetados usando outros métodos de avaliação e que muitas vezes prejudicam a interação do usuário (tornando-a mais lenta por exemplo no caso da inconsistência tipográfica).

Problemas de usabilidade em um diálogo podem ser localizados de quatro maneiras diferentes: em um único local da interface, em dois ou mais locais que devem ser comparados para se detetar o problema, um problema da estrutura geral da interface e como algo que precisa ser incluido na interface. Não existem diferenças significativas na quantidade de problemas localizados de alguma das maneiras, o que indica que avaliadores têm igual facilidade de localizar qualquer um dos tipos de problemas. A diferença que existe é no estágio de implementação da interface e o tipo de problema que pode ser detectado. Por exemplo, componentes de diálogo que precisariam ser incluídas (que faltam) são difíceis de serem detectadas em interfaces em estágio muito primário de desenvolvimento (em papel, por exemplo).

Concluindo, pode-se dizer que avaliação heurística é o método básico da engenharia de usabilidade e é relativamente fácil de ser usado e de ser aprendido. A descrição que fizemos apresenta alguns aspectos como a avaliação de interfaces de domínio

muito específico e atribuição de graus de gravidade que podem parecer um pouco complicadas de inicio por exigirem uma certa expertise dos avaliadores. Mas o mais importante é que se pode começar a fazer avaliação heurística sem considerar esses aspectos e sem muita experiência com avaliação.

Segundo Nielsen (1994) os principais componentes de uma avaliação heurística podem ser resumidos como:

- avaliadores devem percorrer a interface pelo menos duas vezes. Na primeira vez devem se concentrar no fluxo e na segunda nas componentes individuais do diálogo.
- a interface deve ser inspecionada com base em uma lista de princípios de usabilidade, as denominadas heurísticas, e todos os problemas devem ser justificados e detalhados o máximo possível.
- combinar os problemas encontrados por 3 a 5 avaliadores e fazer com que trabalhem individualmente (sem que um influencie o outro)

Depois do trabalho individual o ideal é ter uma reunião final de discussão, incluindo representantes da equipe de desenvolvimento de forma a se ter sugestões para redesign. E graus de severidade podem ser atribuídos aos problemas caso haja necessidade de priorizar a correção de problemas.

Usar o método, não somente melhora a interface sob análise, como também beneficia futuros projetos, o que é um efeito colateral da inspeção que julgamos extremamente importante.

PERCURSO COGNITIVO

Percurso cognitivo (Lewis *et al.* 1990; Polson *et al*. 1992a) é um método de inspeção de usabilidade que tem como foco principal avaliar o design quanto à sua facilidade de aprendizagem, particularmente por exploração.

O foco na aprendizagem foi motivado por resultados de estudos que apontavam que usuários preferem aprender a usar um software por exploração (Carroll and Rosson, 1987; Fisher, 1991). Ao invés de investir tempo em treinamento formal ou leitura de extensivo material de apoio, usuários preferem aprender sobre um software enquanto trabalham em suas tarefas usuais, adquirindo conhecimento sobre as características do software à medida que delas necessitem. Esta abordagem de aprendizagem incremental de certa forma assegura que o custo da aprendizagem de uma determinada característica é em parte determinado pelo seu benefício imediato ao usuário.

O que estamos pretendendo nesta seção é prover uma descrição detalhada de como se faz um percurso cognitivo, em que fase do desenvolvimento do sistema, etc. Portanto, não estaremos explorando os aspectos teóricos que subsidiaram o desenvolvimento do método, que podem ser vistos em Polson *et al*. (1992a).

UMA PRIMEIRA DESCRIÇÃO

Percurso cognitivo é um processo de revisão no qual o autor de um aspecto do design apresenta uma proposta para um grupo de pares. Os pares então avaliam a solução usando critérios apropriados ao design específico.

Os revisores avaliam a interface proposta no contexto de uma ou mais tarefas do usuário. A entrada para uma sessão de percurso inclui uma descrição detalhada da interface (na forma de um protótipo executável ou uma maquete em papel), o cenário da tarefa, suposições explícitas sobre a população de usuários e o contexto de uso, e a seqüência de ações que o usuário terá que fazer para executar corretamente a tarefa. O processo de percurso pode ser dividido em duas fases básicas, fase preparatória e fase de análise, cujas características podem ser vistas resumidamente na Tabela 4.5.

Fase preparatória

- Analistas definem tarefas, seqüências de ações para cada tarefa, população de usuários e a interface a ser analisada

1. Quem serão os usuários do sistema?
2. Qual tarefa (ou tarefas) devem ser analisadas?
3. Qual é a correta seqüência de ações para cada tarefa e como pode ser descrita ?
4. Como é definida a interface?

Fase de análise

- Objetiva contar uma estória verossímil que informe sobre o conhecimento do usuário e objetivos, e sobre o entendimento do processo de solução de problemas que leva o usuário a "adivinhar" a correta solução. Analistas respondem 4 questões:

1. Os usuários farão a ação correta para atingir o resultado desejado?
2. Os usuários perceberão que a ação correta está disponível?
3. Os usuários irão associar a ação correta com o efeito desejado?
4. Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação a tarefa desejada?

- uma estória verossímil de fracaso será contada se algumas das questões acima tiver resposta negativa

TABELA 4.5 - PROCESSO DE PERCURSO COGNITIVO (ADAPTADO DE WARTON, C. *ET AL* . 1994, P.106)

Durante o processo de percurso o grupo de avaliadores considera, em seqüência, cada uma das ações necessárias para completar a tarefa. Para cada ação, os analistas tentam *contar uma estória* sobre interações típicas de usuários com a interface. Eles perguntam o que o usuário tentaria fazer nesse ponto a partir das ações que a interface deixa disponíveis. Se o design da interface for bom, a intenção do usuário fará com que ele selecione a ação apropriada e tenha conhecimento disso, ou seja, em seguida à ação a interface deverá apresentar uma resposta clara indicando que progresso foi feito na direção de completar a tarefa.

DESCRIÇÃO DETALHADA DO PROCEDIMENTO DE PERCURSO

Como já dissemos o percurso cognitivo tem duas fases principais: fase preparatória e fase de análise. Na fase preparatória, os analistas definem as condições de entrada para o percurso: as tarefas, seqüência de ações para cada tarefa, população de usuários e a interface que será objeto de análise. O principal trabalho analítico é feito na segunda fase, durante o qual os analistas trabalham em cada ação de cada uma das tarefas escolhidas. Os detalhes de cada fase, em particular as informações a serem registradas, dependem fortemente de como o percurso vai ser usado no processo de desenvolvimento.

Formalmente, o percurso cognitivo é um método de inspeção para avaliar um design quanto ao aspecto de facilidade de aprendizagem por exploração. Ele pode ser efetuado depois de uma especificação detalhada da interface do usuário, a qual acontece depois da análise de requisitos e definição da funcionalidade de uma aplicação. Um percurso também pode ser efetuado em uma simulação em papel da interface, ou em um protótipo mínimo construído com qualquer ferramenta de prototipação ou ainda em um protótipo completo de um design.

O percurso pode ser um processo individual ou em grupo. Em grupo, o designer apresenta a interface para um grupo de pares, tipicamente após fases significativas do design como a construção de um primeiro protótipo, e usa o feedback da avaliação para modificar ou fortalecer a próxima revisão. Os pares podem incluir outros designers, engenheiros de software, e representantes de outras unidades organizacionais como marketing, documentação e treiamento. Adicionalmente pode ser incluído ao grupo um especialista em avaliação de interfaces. Cada membro do grupo de avaliadores tem um papel específico: o escriba, o facilitador e os demais contribuem com um conhecimento específico: conhecimento do mercado potencial, análise das necessidades do usuário, etc.

Um indivíduo pode também usar o percurso cognitivo para avaliar um design pessoal. Como o percurso é baseado no modelo explícito do processo de aprendizagem por exploração, desenvolvedores participando de seu próprio (ou de outro grupo) percurso também têm a possibilidade de internalizar o conhecimento

sobre o processo associado ao modelo teórico que o embasa. Esse conhecimento pode influenciar em próximas decisões de design que o desenvolvedor tenha que tomar. Portanto, o processo de avaliação pode ser usado em fases bastante iniciais do processo de design por designers individuais, desenvolvedores ou grupos de designers.

De modo geral, percurso cognitivo tem um impacto positivo em todas as fases do design e do processo de desenvolvimento. Muitas vezes as tarefas do usuário, centrais na aplicação, escolhidas para avaliação são utilizadas como carros-chefe de teste de campo e propaganda.

DEFININDO AS ENTRADAS PARA O PERCURSO - FASE PREPARATÓRIA

Antes da fase de análise quatro aspectos devem estar plenamente acordados:

- *Quem são os usuários do sistema?*

Pode ser uma descrição simples e geral tal como, "*pessoas que usam LINUX*". Mas o processo de percurso é mais revelador se a descrição inclui mais especificamente a experiência e conhecimento técnico que podem influenciar os usuários na interação com uma nova interface. Por exemplo, usuários podem ser "*Usuários de Windows que trabalham com o Microsoft Word*". No processo de percurso são considerados o conhecimento do usuário com relação à tarefa e com relação à interface.

- *Qual tarefa (ou tarefas) será analisada?*

O percurso envolve a detalhada análise de uma ou várias tarefas. É possível fazer a análise de todas as tarefas associadas a um sistema com funcionalidade simples, como um sistema de compactação de arquivos, por exemplo. Também para um sistema uma única tarefa pode ser analisada, por exemplo a que se mostrou problemática em versões anteriores. Em geral, para sistemas com alguma complexidade, a análise deverá ser limitada a uma razoável, mas representativa, coleção de tarefas.

A questão crítica é como selecionar essas tarefas. Seleção de tarefas deverá ser baseada em resultados de estudo de mercado, análise de necessidades, análise de conceito e análise de requisitos. Algumas tarefas devem ser escolhidas como exemplo a partir da funcionalidade central da aplicação, isto é, operações básicas que se pretende que o sistema deva suportar. Adicionalmente, outras tarefas que requeiram uma combinação dessas funções básicas podem ser consideradas.

As tarefas selecionadas devem ser o mais concretas e realistas possível. A descrição da tarefa deve conter o contexto necessário, por exemplo, o conteúdo das bases de dados que se espera que o usuário vá utilizar. Esse contexto reflete condições típicas sob as quais o sistema irá operar. Por exemplo, em uma tarefa de recuperação de

dados de uma base de dados, a base de dados usada como exemplo deverá ser grande ou pequena, dependendo do uso pretendido do sistema.

- *Qual a correta seqüência de ações para cada tarefa e como é descrita?*

Para cada tarefa, deve haver uma descrição de como se espera que o usuário veja a tarefa antes de aprender sobre a interface. Também deve haver uma descrição da seqüência de ações para resolver a tarefa na atual definição da interface. Essas ações podem incluir movimentos simples como "*pressionar a tecla ENTER*" ou "*mover o cursor para o menu File*", ou, podem ser seqüências de diversas ações simples que o usuário típico pode executar como um bloco, tais como "*Logar no sistema*" para usuários experientes em UNIX, ou selecionar o "*Salvar como do menu File*" para usuários experientes em Windows. A decisão sobre a granularidade da descrição depende muito da expertise do usuário alvo. A grosso modo, pode-se definir que a descrição deve ser equivalente a descrição que seria colocada em um tutorial eficiente.

- *Qual a interface definida?*

A definição da interface precisa descrever os *prompts* que precedem cada ação requerida para completar as tarefas que estão sendo analisadas como também a reação da interface para cada uma de suas ações. Se a interface já está implementada, toda informação estará disponível na implementação. Entretanto, algumas características importantes do sistema são difíceis de serem apreciadas como, por exemplo, o tempo de resposta, cores, temporização em interfaces com fala, e interações físicas.

Para uma interface descrita em papel, o detalhamento da definição da interface irá depender da experiência pretendida dos usuários alvo com sistemas que já existem. Por exemplo, a preparação para analisar uma aplicação Windows dirigida para usuários experientes em Windows, não precisa prover uma descrição detalhada da aparência padrão de menus Windows; uma descrição simples de seus conteúdos é suficiente.

PERCORRENDO AS AÇÕES - FASE DE ANÁLISE

A fase de análise do percurso consiste em examinar cada ação do caminho da solução e tentar contar uma *estória verossímil* de como o usuário iria escolher aquela ação. Estórias verossímeis são baseadas em suposições sobre objetivos e conhecimento do usuário, e no entendimento do processo de solução de problemas que possibilitará ao usuário escolher a ação correta.

O processo de solução de problemas foi descrito por Polson and Lewis em sua teoria sobre aprendizagem exploratória (Polson and Lewis, 1990). Brevemente, este processo de solução de problemas estabelece que usuários: (1) iniciam com uma

descrição grosseira da tarefa que tem que efetuar (2) exploram a interface e selecionam as ações que eles imaginam as mais adequadas para efetuar a tarefa ou parte dela (3) observam a reação da interface para verificar se suas ações tiveram o efeito desejado e (4) determinam qual ação efetuar a seguir.
A teoria também aponta algumas heurísticas específicas que os usuários aplicam quando tomam suas decisões. Em particular, usuários freqüentemente seguem a estratégia de *"label-following"*, a qual os faz selecionar uma ação se o rótulo para essa ação combina com a descrição da tarefa (Engelbeck, 1986). Por exemplo, se o usuário deseja "imprimir um arquivo" poderá selecionar uma ação com o rótulo (ou ícone) "imprimir" ou "arquivo".

As caraterísticas críticas da interface, são portanto, aquelas que provêem uma ligação entre a descrição do usuário para a tarefa e a ação correta, e aquelas que provêem feedback indicando o efeito da ação do usuário. Conforme o percurso vai avançando o analista aplica essa teoria ao relatar e avaliar sua estória de como o usuário escolheria a ação prevista pelo designer em cada passo. Os analistas ao contarem suas estórias devem responder a quatro questões:

- *Os usuários farão a ação correta para atingir o resultado desejado?*

Suponha que em uma determinada aplicação antes de mandar imprimir um documento é preciso selecionar uma determinada impressora. O usuário irá saber que tem que fazer isso antes de executar a tarefa de impressão?

- *Os usuários perceberão que a ação correta está disponível?*

Se a ação estiver disponível no menu e for facilmente identificada não há problema. Mas suponha que para imprimir um documento seja necessário dar um clique em um ícone com o botão esquerdo do mouse. O usuário pode não pensar nunca nisso.

- *Os usuários irão associar a ação correta com resultado desejado?*

Se existe um item de menu claro e facilmente encontrado informando "Selecionar Impressora" então não há problemas, mas se no menu só tem a opção "ImpSis" aí as coisas talvez não sejam tão evidentes.

- *Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação a tarefa desejada?*

Se após a seleção o usuário tiver um feedback informando "Impressora Laser XXX da sala YY selecionada" então sem problemas. O pior caso é a ausência de resposta.

Essas questões servem de guia para construir as estórias, não sendo requisitos obrigatórios, mas são as mais usadas. Podem ser definidos pelo analista outros critérios que o levem a contar estórias verossímeis. No caso de seguir o critério das quatro questões acima, a falha em qualquer uma das questões implicará em problemas com a interface.

REGISTRO DA INFORMAÇÃO DURANTE A AVALIAÇÃO

Durante o percurso é importante registrar toda informação gerada. Para uma avaliação feita em grupo, muito comum no caso de percurso cognitivo, é recomendado que sejam usados materiais visíveis ao grupo. Também é muito conveniente gravar na forma de vídeoteipe todo o processo de avaliação, incluindo comentários dos avaliadores. O vídeo é uma importante fonte de referência para eximir dúvidas, verificando comentários e decisões tomadas durante o processo de avaliação. O material visível ao grupo serve para registrar e resumir todas as decisões e informações chave para o grupo.

Existem diversas informações importantes que devem ser registradas de forma visível ao grupo durante o processo de avaliação. Indispensáveis são as informações sobre o conhecimento esperado do usuário, suposições sobre a população de usuários, notas sobre mudanças de design e a estória verossímil desenvolvida durante o processo de percurso. São sugeridas três instâncias de registro: uma para os pontos chave da estória do grupo, outra para catalogar toda informação sobre o usuário e uma terceira que registre notas sobre mudanças de design e outras observações paralelas importantes.

Quanto às informações sobre os usuários deve-se ter o registro para cada classe de usuários, das seguintes informações:
que o usuário precisa conhecer antes de executar a tarefa
que o usuário poderá aprender enquanto executa a tarefa

ESTÓRIAS DE SUCESSO E ESTÓRIAS DE FRACASSO

Para se ter uma idéia mais clara do tipo de estórias que analistas geram durante um percurso, iremos apresentar alguns pequenos exemplos de estórias de sucesso e de fracasso verossímeis, de interfaces que existem ou existiram efetivamente.

EXEMPLOS DE ESTÓRIAS DE SUCESSO

Estória de sucesso 1:
Um usuário experiente em Windows inicia uma tarefa dando um clique no ícone da aplicação para abrí-la.

Defesa da Credibilidade

- usuário abre a aplicação porque ele sabe que deve abrir uma aplicação para usá-la
- usuário conhece por experiência que pode dar clique sobre o ícone da aplicação

- usuário sabe por experiência que o clique é a ação a ser usada
- mudanças na tela ou na barra de menu sinalizam o inicio da aplicação

Deve-se notar que as três primeiras partes dessa estória não seriam válidas para uma pessoa novata no uso de computadores, e a segunda e terceira não seriam válidas para pessoas inexperientes em Windows.

Estória de sucesso 2:
Um usuário experiente em Windows abre o menu Tabela para preparar uma tabela em um editor de textos.

Defesa da Credibilidade
- usuário está tentando preparar uma tabela pois essa é a tarefa
- usuário abre o menu Tabela pois o título Tabela é claramente relacionado com o que ele está querendo fazer
- usuário sabe que pode abrir um menu e essa é a ação a ser feita se o título parece bom, por experiência com o Windows
- usuário sabe que tudo está indo bem quando da leitura das opções do menu.

Estória de sucesso 3*:*
Um usuário de cartão de crédito está usando o sistema telefônico para obter informações sobre seu saldo. O sistema diz "entre com o número de seu cartão" e o usuário disca seu número.

Defesa de Credibilidade
- usuário entra com o número porque o sistema informou isso para ele
- usuário usa as teclas do seu telefone porque elas estão visíveis e não existe outra possibilidade para entrar o número
- usuário conhece o número do cartão pois é dele o cartão e ele tem então acesso ao número
- usuário sabe que as coisas estão indo bem quando o sistema começa a dizer os números de um menu de opções de serviço.

CARACTERÍSTICAS COMUNS DE SUCESSOS

Com esses exemplos em mente, podem ser resumidos os quatro pontos que os analistas consideram em cada passo e suas características mais relevantes:

1. Usuários irão conhecer **Qual resultado querem alcançar**:
   - Porque é parte da tarefa original, ou

- Porque eles têm experiência no uso do sistema, ou
- Porque o sistema diz a eles o que devem fazer

2. Usuários irão saber que **Uma ação está disponível**:
   - Por experiência, ou
   - Observando algum dispositivo, ou
   - Observando a representação de uma ação

3. Usuários irão saber **Qual ação é adequada** para o resultado que estão tentando obter:
   - Por experiência, ou
   - Porque a interface provê um *prompt* ou rótulo que conecta a ação ao que ele está tentando fazer, ou
   - Porque as outras ações não parecem corretas

4. Usuários irão saber que **As coisas estão indo bem** depois da ação:
   - Por experiência, ou
   - Por reconhecer a conexão entre a resposta do sistema e o que ele está tentando fazer

Muitos desses pontos enfatizam a importância de se conhecer como o usuário descreve a tarefa. Quando o sistema usa a mesma terminologia que o usuário, ele provavelmente irá escolher a ação correta. O mesmo acontece com a resposta do sistema (feedback) quando é dada na linguagem do usuário. Quando termos não usuais são utilizados pelo sistema, o usuário poderá encontrar dificuldades em obter sucesso sem um conhecimento adicional.

EXEMPLOS DE ESTÓRIAS DE FRACASSOS

Estórias de sucesso requerem sucesso em todos os quatro critérios de análise, enquanto estórias de fracasso tipicamente falham em apenas um dos quatro critérios implicando em que uma estória verossímil não possa ser contada.

Vamos então agrupar nossos exemplos de estórias de fracasso de acordo com o critério em que falham.

- **Critério 1**: *Os usuários farão a ação correta para atingir o resultado desejado?*

  Exemplo 1:
  Considerando um usuário de Windows com familiaridade com editores gráficos básicos, que ao usar um editor de *gifs* animados (Figura 4.13) tenta dar um *zoom in* na figura sob edição. Para isso ele tem que alterar o número que está na caixa de diálogo à esquerda, no topo da tela (inicialmente com 100%).

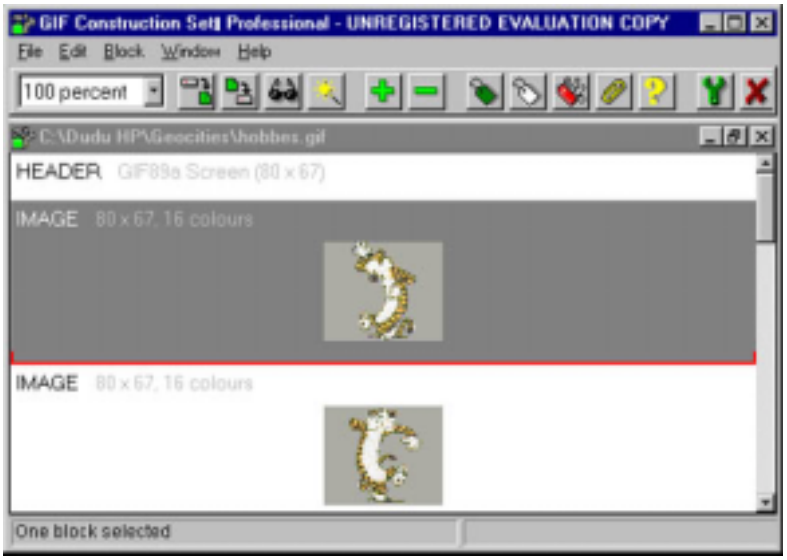


FIGURA 4.13 - PÁGINA DE SOFTWARE DE EDIÇÃO DE *GIFS* ANIMADOS

Estória de fracasso: Os usuários irão acionar diretamente o botão com o sinal de menos pois esse é o padrão de ícone usado na maioria dos editores para dar *zoom in*. Não obteriam portanto o resultado desejado pois o respectivo botão elimina o último *gif* editado da seqüência de *gifs*.

Exemplo 2*:*
Considerando um usuário experiente em Windows e com conhecimentos básicos do editor de textos Word. Ele quer numerar as páginas de um texto colocando a numeração no seguinte formato *pg – número*, centralizado no topo da página. Para fazer isso ele terá que ir ao menu View, selecionar a opção Header and Footer e aí editar o formato desejado da numeração em uma caixa de entrada que é aberta no texto, conjuntamente com uma caixa de ferramentas nomeada Header e Footer (figura 4.14)

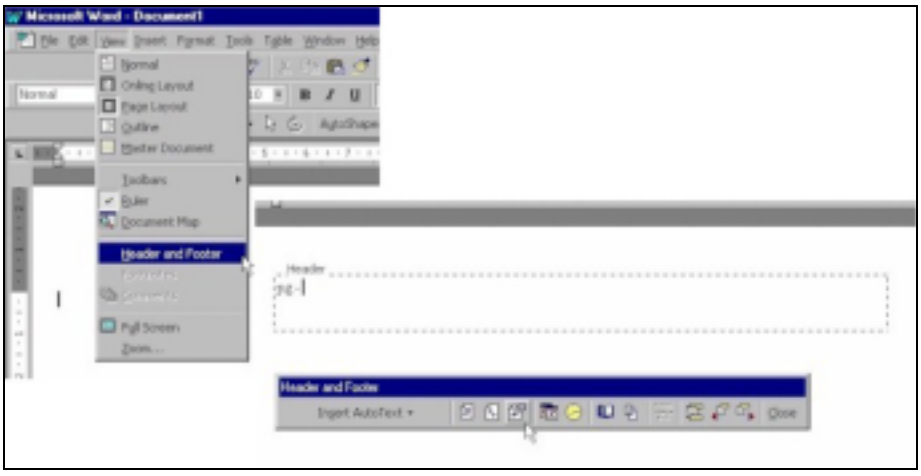


FIGURA 4.14 TELAS DO EDITOR DE TEXTOS WORD PARA WINDOWS COM MENUS VIEW E TELA DE EDIÇÃO DE HEADER

Estória de fracasso: Dificilmente o usuário irá associar a ação de numerar páginas em um determinado formato com a opção *View*. Ele irá certamente escolher o item de menu *Insert* com a opção *Page Numbers* e *Format Page Numbers* em seguida, falhando em sua ação.

- **Critério 2**: *Os usuários perceberão que a ação correta está disponível?*

  Exemplo 1*:*
  Em um particular programa que gera gráficos (pizza, barras, etc.), para se alterar a fonte e outras especificações do título de um gráfico deve-se dar um duplo clique no título do gráfico e somente dessa forma, irá abrir uma caixa de diálogo que permitirá a edição desejada. Considerando usuários experientes em interfaces gráficas.

  Estória de fracasso: os usuários freqüentemente não consideram a possibilidade de usar um duplo clique nesse contexto

  Exemplo 2*:*
  No sistema sistema operacional Windows 95 para se desligar o computador deve-se dar um clique no botão Iniciar. Considerando-se usuários novatos em Windows. (Figura 4.15)

  Estória de fracasso: Usuários principiantes não descobrem que essa é a ação a ser adotada para se desligar corretamente o computador.

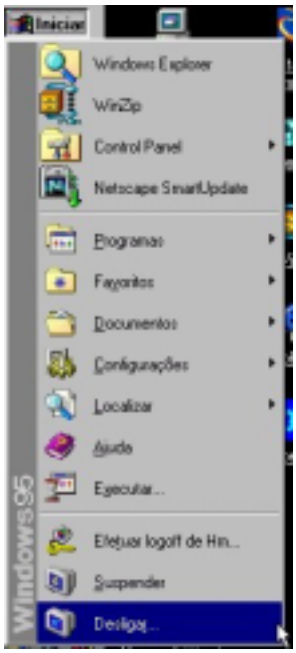


FIGURA 4.15 - MENU INICIAR DO WNDOWS 95

Com freqüência, sistemas orientados por comando têm falhas com relação a esse critério. Usuários geralmente sabem o que querem fazer (por exemplo, abrir um diretório, mudar a proteção de um arquivo, listar um arquivo, etc), mas eles muitas vezes não sabem, e nem sempre conseguem encontrar, o nome do comando.

- **Critério 3:** *Os usuários irão associar a ação correta com o efeito desejado?*

  Exemplo 1:
  Em um processador de textos, orientado a menus, existem dois menus. Um ítem de menu é chamado FORMAT e outro é chamado FONT. Estilo de fontes são parte do menu FORMAT. Considerando usuários iniciantes.

  Estória de fracasso: Os usuários não saberão qual menu escolher se quiserem colocar uma palavra em itálico.

  Exemplo 2:
  Considerando o exemplo mencionado anteriormente sobre a colocação de número de páginas no processador de textos Word do Windows no formato *pg - número da página*

Estória de fracasso: Mesmo considerando que os usuários saibam que esse formato de numeração tenha que ser colocado sob a forma de Header, dificilmente irão associar a operação de inserir um Header com o menu View.

- **Critério 4**: *Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação à tarefa desejada?*

  Exemplo*:*
  No sistema Windows NT, para desligar o computador deve ser acionado o menu iniciar da mesma forma que no sistema Windows (Figura 4.15). Depois de selecionada a opção correta aparece na tela uma caixa de diálogo para o usuário iniciar novamente o sistema.

  Estória de fracasso: Usuários muitas vezes não percebem que sairam do sistema com sucesso. Ao invés disso, alguns usuários automaticamente dão um clique ou enter e se vêem presos em um *loop*.

Alguns outros problemas que devem ser observados durante um percurso e que poderão resultar em estórias de fracasso:

Time outs: alguns sistemas, especialmente os baseados em telefone ou que requisitam senhas (como os caixas automáticos por exemplo), dão ao usuário um tempo determinado para tomar a ação. Tentar determinar se o tempo dado está adequado considerando o usuário alvo.

Ações difíceis fisicamente: apertar teclas simultaneamente, especialmente quando devem ser pressionadas ou soltas em uma determinada ordem, é difícil. Também a dificuldade em selecionar um componente muito pequeno na tela com o mouse ou toque. Problemas desse tipo existem em editores gráficos que muitas vezes exigem um controle motor muito fino para desenhar uma simples linha.

Terminadores de ação: usuários geralmente esquecem ações que indicam que alguma parte da tarefa está concluída, como colocar um ";" ou "." após um comando em uma linguagem de programação. Isso porque para o usuário é óbvio o término. Isso ocorre mesmo com usuários muito experientes que sabem o que deve ser feito e mesmo assim estão sempre tendo que efetuar correções.

COMO USAR OS RESULTADOS DO PERCURSO PARA CORRIGIR PROBLEMAS

Observamos que falhas são tipicamente associadas com um dos quatro critérios de análise; sugestões gerais de como corrigí-las também podem ser organizadas da mesma forma. Consideremos, portanto, cada um dos critérios:

- *Os usuários farão a ação correta para atingir o resultado desejado?*
  Se a interface falha nesse ponto - ou seja, o usuário não está fazendo a coisa correta - existem pelo menos três opções para tentar corrigir: (1) a ação precisa ser eliminada, ou o sistema a efetua (por exemplo, no caso de se ter obrigatoriamente que escolher uma impressora antes de efetuar uma impressão) ou deve ser combinada com outra ação mais adequada (2) um *prompt* deve ser dado ao usuário informando-o sobre qual ação deve ser feita (3) alguma outra parte da tarefa precisa ser mudada de forma a que o usuário entenda a necessidade da ação, talvez porque passe a ser consistente com alguma outra parte da seqüência de ações.

- *Os usuários perceberão que a ação correta está disponível?*
  Se o usuário tem os objetivos corretos mas não sabe que a ação está disponível na interface, a solução é associar a ação a um controle mais óbvio. Isso tipicamente envolve o uso de um menu ou *prompt*, ao invés de uma tecla de controle; ou pode envolver associar a ação a um controle mais escondido mas que pode ser facilmente descoberto por estar significativamente agrupado.

- *Os usuários irão associar a ação correta com o efeito desejado?*
  Para corrigir essa falha os designers precisam conhecer seus usuários e a forma como descrevem a tarefa. Com essa informação o designer pode prover rótulos e descrições para ações que incluam palavras que os usuários freqüentemente usam para descrever suas tarefas. Pode ser necessário redefinir rótulos ou outros controles que os usuários selecionam em preferência aos corretos. Também pode envolver um reagrupamento de funções em menus de forma mais significativa com relação à visão que o usuário tem da tarefa.

- *Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação à tarefa desejada?*
  Claramente, na maioria das situações qualquer feedback é melhor que nenhum feedback. E feedback que indica o que aconteceu é sempre melhor que um feedback que indica que alguma coisa aconteceu. Respostas são sempre mais efetivas quando usam termos (ou gráficos) relacionados à descrição do usuário para a tarefa.

De maneira geral, lidar com o problema eliminando ou reagrupando ações é sempre preferível a tentar corrigir os problemas usando *prompts* ou feedback. Quando a

interface apresenta uma série de problemas que indicam uma concepção de design que não confere com a descrição que o usuário tem da tarefa, o designer precisa avaliar a possibilidade de corrigir esses problemas efetuando uma reorganização global da interface ao invés de ficar corrigindo problemas locais.

ESCOPO E LIMITAÇÕES DO MÉTODO

Percurso cognitivo, como ja dissemos, enfoca apenas um atributo de usabilidade, a facilidade de aprendizagem. Mas deve ser considerado que outros atributos de usabilitade são fortemente conectados à facilidade de aprendizagem, como funcionalidade e facilidade de uso.

Entretanto, o uso de percurso cognitivo como o único método para avaliar uma interface pode conduzir o design à um forte compromisso com a facilidade de aprendizagem. Por exemplo, o processo de percurso pode conduzir à uma avaliação negativa de características que objetivem o aumento de produtividade se essas características tornarem mais difícil decidir como efetuar uma determinada tarefa.

Percurso cognitivo avalia cada passo necessário para a realização de uma tarefa com o objetivo de descobrir erros de design que podem dificultar a aprendizagem por exploração. O método encontra os conflitos entre designers e usuários sobre a concepção da tarefa, escolhas pobres de palavras de menus e rótulos de botões, e respostas inadequadas sobre conseqüências de ações. O método também mostra as suposições explícitas e implícitas feita pelos desenvolvedores sobre o conhecimento do usuário sobre a tarefa e das convenções da interface. O procedimento de avaliação toma a forma de uma série de questões que são feitas sobre cada passo dentro de uma tarefa derivadas da teoria de aprendizagem por exploração (Polson *et al*. 1992a). Não existem relatados dados estatísticos que informem o número de avaliadores a serem usados de forma a se obter confiabilidade nos resultados das avaliações. Por ser uma avaliação muito detalhada ela quase sempre é realizada durante o processo de design avaliando sempre pequenos cenários de interação que se apresentem críticos com relação a característica de aprendizagem. Resultados muito satisfatórios são conseguidos com avaliações feitas com grupos de 3 a 5 avaliadores trabalhando conjuntamente. Existem software, especificamente desenvolvidos para o processo de percurso cognitivo, que apresentam formulários pré-formatados e que são usados durante as avaliações, tanto na fase preparatória como durante a avaliação.

Mais uma vez, reforçamos que todos os métodos têm suas características fortes e fracas. O forte compromisso do percurso cognitivo com a facilidade de aprendizagem sacrifica a obtenção de informação válida sobre outros importantes atributos de usabilidade como, por exemplo, consistência global da interface. Isso reforça a necessidade de se usar diversos métodos complementares de forma a garantir uma ampla cobertura da interface em todas as suas características relevantes.

Portanto, todo método de avaliação não pode ser considerado completo e sim complementar suas características com outros métodos.

TESTE DE USABILIDADE

Teste com usuário é um método fundamental de usabilidade. Desenvolvedores tradicionais resistem à idéia, dizendo que teste de usabilidade sem dúvida alguma é uma boa idéia, mas limitações de tempo e de recursos os impedem de fazê-lo. Mas esse cenário está mudando muito rapidamente. Gerentes de desenvolvimento estão percebendo que ter agendado testes de usabilidade é um poderoso incentivo para o término da fase de design. E a surpresa é que resultados práticos têm demonstrado que testes de usabilidade não somente têm acelerado muitos projetos como também têm produzido uma significativa redução em seus custos (Gould and Lewis, 1985; Gould *et al.*, 1991; Karat, 1994).

O movimento em direção aos testes de usabilidade estimulou a construção de laboratórios de usabilidade (Dumas and Redisch, 1993; Nielsen, 1993). A IBM fez em Boca Raton, Flórida, uma sofisticada construção com 16 laboratórios dispostos em círculo com uma base de dados centralizada para registrar a performance e o *log* de uso de produtos testados. Muitas outras empresas seguiram o exemplo e abraçaram a idéia com bastante força. Isso demonstra a crescente preocupação com o usuário, que está permeando a maioria dos desenvolvimentos atuais.

Um laboratório de usabilidade geralmente abriga uma pequena equipe de pessoas com experiência em teste e design de interface de usuário. A equipe do laboratóro geralmente entra em contato com representantes da equipe de desenvolvimento no início de um projeto, de forma a estabelecer um plano de teste com datas definidas e custos alocados. Ela também participa na fase inicial de análise da tarefa e revisão de design, fazendo sugestões e provendo informações, e ajudando no desenvolvimento do conjunto de tarefas para o teste de usabilidade.

A disponibilidade de um laboratório não deve ser considerada condição para a realização de um teste de usabilidade e sim como uma grande facilitação. Quase todas as formas de teste podem ser feitas nos mais diversos locais, desde que devidamente preparados. Também não deve ser considerada uma condição a existência de avaliadores experientes para se efetuar um teste. Bons resultados têm sido obtidos com experimentadores novatos que aprendem o método de teste (Nielsen, 1992; Wright and Monk, 1991)

OBJETIVOS E PLANO DE TESTE

Antes de qualquer teste ter inicio é preciso estabelecer seus objetivos pois isso tem um impacto significativo no tipo de teste a ser feito. A principal distinção é se o teste tem como objetivo obter uma ajuda no desenvolvimento ou é um teste que visa avaliar a qualidade global de uma interface. No primeiro caso interessa saber em detalhe quais aspectos da interface estão bons ou ruins, e como o design pode ser melhorado. É uma forma mais gradual de analisar a interface, e nesse caso usualmente se aplica o teste denominado *pensar em voz alta* (*thinking- aloud test*) que veremos a seguir. No segundo caso, como se quer uma visão mais global de uma interface em fase final de definição geralmente se utiliza testes que dêem medidas de performance que apresentaremos em seções a seguir. Em qualquer uma das situações deve ser desenvolvido um plano detalhado de teste onde, dentre outras mais específicas, as seguintes questões devem ser respondidas:

- O objetivo do teste: o que se deseja obter?
- Quando e onde o teste irá acontecer?
- Qual a duração prevista de cada sessão de teste?
- Qual o suporte computacional necessário?
- Qual software precisa estar a disposição?
- Qual deverá ser o estado do sistema no inicio do teste?
- Quem serão os experimentadores?
- Quem serão os usuários e como serão conseguidos?
- Quantos usuários são necessários?
- Quais as tarefas que serão solicitadas aos usuários?
- Qual critério será utilizado para definir que os usuários terminaram cada tarefa corretamente?
- Quanto o experimentador poderá ajudar o usuário durante o teste?
- Quais dados serão coletados e como serão analisados uma vez que tenham sido coletados?
- Qual o critério para determinar que a interface é um sucesso? (p. ex: *nenhum problema de usabilidade novo com severidade maior ou igual a 3)*

Deve-se sempre estar atento a dois problemas vinculados a um teste de usabilidade: a **confiabilidade** e a **validade**. Como confiabilidade entendemos o grau de certeza de que o mesmo resultado será obtido se o teste for repetido; e como validade, o fato dos resultados de teste refletirem os aspectos de usabilidade que se deseja testar.

No quesito **confiabilidade** deve-se estar atento às diferenças individuais entre os usuários. Por exemplo, cuidar do grau de confiança que se dá a afirmações do tipo:

> *"usuário A usando a interface X executa uma tarefa 40% mais rápido que o usuário B usando a interface Y"*

que não necessariamente significam que a interface A tem melhor qualidade, pois não é incomum ter-se um grupo de usuários onde o melhor usuário é 10 vezes mais

rápido que o mais lento e que os 25% melhores são 2 vezes mais rápidos que os 25% piores (Egan, 1988)

Quanto à **validade**, o que se gostaria de assegurar é que o resultado obtido tenha realmente significado considerando-se o produto real em uso e fora da situação de laboratório. Deve-se então nesse ponto estar atento à escolha dos usuários, à escolha das tarefas e à diferença entre equipamentos (situação de teste e situação real)

A regra principal para se efetuar a **escolha dos usuários** é que sejam tão representativos quanto possível com relação aos usuários reais do sistema. O ideal seria envolver usuários reais do sistema, mas isso nem sempre é possível. Se o grupo de sujeitos não é composto de usuários reais, ele deve ter idade e nível educacional similar ao grupo de usuários alvo. Também similar deve ser sua experiência com computadores, com o tipo de sistema que está sendo testado e o conhecimento do domínio da tarefa. Certamente não é conveniente testar uma interface voltada para o público em geral e utilizar estudantes de computação como grupo de teste: eles certamente não são representativos da população de usuários alvo.

Os usuários devem ser tratados com respeito e principalmente serem informados de que é a interface e não eles que estão sendo testados. Geralmente se sentem sob muita pressão na situação de teste e isso os leva a aprenderem mais lentamente e a fazerem mais erros, sentindo-se estúpidos quando experimentam dificuldades.

Os experimentadores devem ser preparados no sentido de terem conhecimento extenso sobre a aplicação e a respectiva interface de usuário. Não precisam saber como o sistema foi implementdo, mas devem estar prontos a lidar com problemas que afetem o teste, por exemplo, problemas que levem o sistema a cair. Nada impede que sejam os próprios designers desde que esses estejam preparados no sentido de manter uma certa isenção no sentido de não mascarar os resultados do teste. Geralmente eles tendem a ajudar muito os usuários e a antecipar situações de erro, dado seu extenso conhecimento da interface.

As tarefas a serem feitas durante um teste devem ser as mais representativas possíveis e devem dar uma cobertura razoável das partes mais significativas da interface. Devem poder ser completadas no tempo definido para uma sessão de teste (de 1 a 3 horas). Devem ter grau de dificuldade gradativa para dar mais confiança ao usuário e devem ser planejadas para que possam ser interrompidas a qualquer tempo, caso o usuário assim o deseje. A descrição de cada tarefa a ser efetuada deve ser feita por escrito e deve ser tão realista quanto possível e inserida em um cenário de uso.

Geralmente, um teste piloto é efetuado com um pequeno grupo (de 1 a 3) de usuários, para refinar todos os procedimentos definidos.

ETAPAS DE UM TESTE

Basicamente um teste é composto de quatro etapas:

- *Preparação*
  Nessa etapa se garante que tudo estará pronto antes do usuário chegar. Muito cuidado deve ser tomado com relação aos equipamentos que serão utilizados, devem estar "limpos" (de resultados de outros teste, alarmes sonoros, etc).

- *Introdução*
  É uma fase muito importante, onde os usuários são apresentados à situação de teste e de alguma forma colocados a vontade. Alguns pontos que devem ser falados aos usuários nessa introdução podem ser destacados:
  - O propósito do teste é avaliar o sistema e não o usuário
  - Não devem se preocupar em ferir sentimentos dos experimentadores (designers) com suas observações
  - Os resultados do teste servirão para melhorar a interface do usuário
  - Relembrar que o sistema é confidencial e não deve ser comentado com outros (que inclusive podem vir a ser futuros usuários em outros testes)
  - A participação no teste é voluntária e podem parar a qualquer tempo
  - Os resultados do teste não serão colocados publicamente e o anonimato do participante estará garantido
  - Explicar sobre o uso de gravações de vídeo ou audio que estarão sendo feitas (o ideal é não gravar a face do usuário)
  - Explicar que podem fazer qualquer pergunta durante o teste, mas que nem sempre o experimentador irá ajudá-los ou responder suas questões
  - Instruções específicas sobre a forma do teste (p. ex.: falar em voz alta, ou fazer as atividades o mais rápido que puder, etc.)

- *Teste*
  Durante o teste deve ser escolhido somente um experimentador para falar com o usuário, para evitar confusão, e é importante que:
  - evite qualquer tipo de comentário ou expressões sobre a performance ou observações do usuário
  - evite ajudar o usuário, a não ser que ele esteja realmente em dificuldades muito graves

- *Sessão final*
  Depois do tempo definido para completar as tarefas - usualmente de 1 a 3 horas - os participantes são convidados a fazerem comentários ou sugestões gerais, ou a responderem um questionário específico.

Gravar em vídeo os participantes efetuando as tarefas é sempre um recurso valioso para uma posterior revisão. Analisar vídeos é um trabalho extremamente difícil e

tedioso, portanto sempre devem ser feitas cuidadosas anotações ou coletados *logfiles* durante o teste de modo a reduzir o tempo dispendido em encontrar acontecimentos críticos (Harrison, 1991). A reação de designers vendo esses vídeos de usuários cometendo erros é muito poderosa e motivadora. Quando designers vêem usuários repetidamente acessando o menu errado, eles se convencem que rótulos e posições devem ser mudadas.

Uma técnica efetiva durante um teste de usabilidade é solicitar que os usuários *pensem em voz alta* sobre o que estão fazendo. A atmosfera informal de uma sessão que usa essa técnica é extremamente agradável, e freqüentemente leva à muitas sugestões espontâneas de melhorias.

PENSANDO EM VOZ ALTA

É uma técnica muito valiosa utilizada originalmente como um método de pesquisa psicológico. Solicita-se ao usuário que verbalize tudo que pensa enquanto usa um sistema e a expectativa é que seus pensamentos mostrem como o usuário interpreta cada item da interface (Lewis, 1982). Certamente é uma técnica não adequada quando se deseja medidas de performance. Geralmente os usuários ficam mais lentos e cometem menos erros quando pensam em voz alta.

O experimentador tem que ser bem preparado no sentido de levar o usuário a falar sempre e nunca interferir no uso do sistema pelo usuário. Formas de questionamento usuais podem ser relacionadas:

- O que você está pensando agora?
- O que você acha que essa mensagem significa (depois do usuário notar a mensagem)?
- Se o usuário pergunta se pode fazer alguma coisa: O que você acha que vai acontecer se fizer isso?
- Se o usuário se mostra surpreso: Era isso que você esperava que iria acontecer? O que esperava?

Os comentários dos usuários devem ser criteriosamente analisados e nunca aceitos indiscriminadamente pois podem dar falsa impressão das razões de um determinado problema. Os usuários têm teorias nem sempre verdadeiras.

A principal força dessa técnica é mostrar *o que* os usuários estão fazendo e *porque* estão fazendo *enquanto* estão fazendo, evitando as racionalizações posteriores.

No sentido de incentivar o pensar em voz alta muitas vezes se coloca usuários trabalhando aos pares de forma a produzirem mais conversas a medida que um

participante explica para o outro seus procedimentos, sem a inibição de estar falando com alguém que "sabe mais" sobre o sistema. Essa alternativa é muito usada em testes que envolvem crianças como sujeitos do teste.

Outra alternativa ao pensar em voz alta é fazer com que o usuário comente depois suas ações gravadas em vídeo. Isso auxilia quando se está também interessado em obter dados qualitativos de performance, mas deve ser levado em conta que todo o teste demora pelo menos o dobro do tempo.

MEDIDAS DE PERFORMANCE

Estudos de medidas quantitativas formam a base de muitas pesquisas tradicionais em fatores humanos, como visto no Capítulo 2. Também são importantes em usabilidade para avaliar se os objetivos de usabilidade foram efetivamente atingidos e também para comparar produtos competitivos.

Em usabilidade tem-se o critério de eficiência de uso como uma das *guidelines* de usabilidade. Dentro desse, são fundamentais algumas medidas de performance na forma de tomadas de tempo, por exemplo. Como e quando marcar tempos deve ser decidido *a priori* de acordo com os dados necessários na coleta. Por exemplo, se se quer saber quanto o usuário demora fazendo uma determinada tarefa é preciso primeiro definir quando começa e quando termina a tarefa e depois se o tempo será cronometrado pelo próprio usuário, pelo computador, pelo experimentados, etc.

Medidas típicas de usabilidade que são quantificáveis incluem:

- O tempo que o usuário gasta para fazer uma determinada tarefa
- O número de tarefas de diferentes tipos que são completadas em determinado limite de tempo
- A razão entre interações de sucesso e de erro
- O número de erros do usuário
- O número de ações errôneas imediatamente subseqüentes
- O número de comandos (ou diferentes comandos) ou outras características que foram utilizados pelo usuário
- O número de comandos ou outras características nunca utilizados pelo usuário
- O número de características do sistema que o usuário consegue se lembrar na sessão subseqüente ao teste
- A freqüência de uso de manuais ou do sistema de *help* e o tempo gasto usando esses elementos do sistema
- Quão freqüentemente o manual/sistema de *help* resolveu o problema do usuário
- A proporção entre comentários do usuário favoráveis e críticos com relação

ao sistema
- O número de vezes que o usuário expressou frustração (ou alegria)
- A proporção de usuários que disse preferir o sistema a outro sistema competidor
- A proporção de usuários utilizando estratégias eficientes e ineficientes
- A quantidade de "tempo morto" - quando o usuário não está interagindo com o sistema (ou esperando resposta ou pensando)
- O número de vezes que o usuário desviou do objetivo da tarefa

Certamente somente um pequeno subconjunto de medidas pode ser coletado em uma particular situação de teste.

A maioria dos testes de usabilidade são feitos em laboratórios onde os usuários são observados diretamente pelos avaliadores. Entretando, a localização remota e distribuída dos usuários - freqüentemente na rede, atualmente o principal ambiente para distribuição e uso de aplicações de software, tais como os CSCW - dificulta, e até impede, a possibilidade da observação direta em testes de usabilidade. Também deve ser considerado que a rede em si e o trabalho remoto têm se tornado parte intrínseca de padrões de uso. Adicionalmente, desenvolvedores muitas vezes têm dificuldade em conseguir usuários representativos que possam participar de testes de usabilidade nos laboratórios; e o contexto de trabalho do usuário dificilmente consegue ser reproduzido em situação de laboratório. Todos esses fatores muitas vezes tornam o custo de um teste de usabilidade proibitivo.

Essas barreiras para o teste de usabilidade têm levado à uma extensão do teste para além dos limites dos laboratórios e começam a surgir métodos de teste de usabilidade remotos, tipicamente usando a rede como uma ponte de acesso aos usuários em seu ambiente natural de trabalho. Define-se portanto um teste de usabilidade remoto como aquele em que o observador que efetua a observação e análise e o usuário estão separados em tempo e espaço. Alguns resultados preliminares apontam para a efetividade desses métodos que usam tipicamente uma combinação dos mecanismos de comunicação eletrônicos, como tele-conferência por exemplo, e software especificamente construídos para coletar dados de uso (Hartson *et al*, 1996)

Temos também como modalidade de teste de usabilidade os denominados **testes de campo** que objetivam colocar novas interfaces em ambientes reais de uso por um determinado período de tempo. Testes de campo dão melhor resultado quando se pode dispor de software que geram arquivos *log* que capturam erros, comandos usados, freqüência de acesso a *helps* e mais algumas medidas de produtividade. Quando se conta somente com a resposta do usuário os resultados não são muito satisfatórios. Somente cerca de 20% dos usuários inscritos participam ativamente do teste; os demais estão mais interessados em obter uma primeira versão do produto. Um dos maiores testes de campo já realizados foi o efetuado pela Microsoft na avaliação da versão Beta do Windows 95, onde cerca de 400.000 usuários em todo o

mundo receberam versões preliminares do software e foram convidados a enviar comentários.

CONSIDERAÇÕES FINAIS

Designers e profissionais de IHC procuram por métodos rápidos e baratos de avaliação de interfaces em substituição aos testes de laboratório que geralmente são caros e muitas vezes sem possibilidade de serem realizados por falta de condições estruturais. Em virtude dessa situação, as técnicas de avaliação denominadas métodos de inspeção de usabilidade foram propostas com a promessa de oferecer informação de usabilidade de modo mais rápido e barato que os tradicionais testes de usabilidade. Os mais populares desses métodos incluem avaliação heurística e percurso cognitivo vistos anteriormente em detalhe neste capítulo.

Esses métodos, cada qual com seus procedimentos próprios, provêem dados que podem ser usados quando testes de usabilidade não são possíveis ou em conjunção com eles. Mas resultados de experimentos comparativos confirmam que até o momento eles não substituem os testes com usuário. Existe certamente um fator econômico a considerar na adoção de métodos de inspeção mas a relação custo-benefício precisa ser bem analisada.

Um amplo estudo (Desurvire, 1994) comparando testes de usabilidade, avaliação heurística e percurso cognitivo apresenta alguns resultados interessantes:

- Os resultados dos métodos de inspeção são melhores quando os avaliadores são especialistas em avaliação. Mesmo assim, não substituem o teste de usabilidade: nos experimentos relatados os melhores avaliadores, usando o método de melhor performance, não detetaram em média 56% dos problemas encontrados no teste de usabilidade.

- Avaliação heurística permite uma avaliação global da interface faciltando a identificação de melhorias na interface. Foi a mais eficaz na detecção de erros e principalmente na identificação da maioria de erros sérios. Além disso é a de menor custo.

- Teste de usabilidade é o mais eficaz em detetar erros, mas o mais caro. O custo de um teste de usabilidade é da ordem de 50 vezes o custo dos métodos de inspeção (isso está mudando com a introdução da metodologia de testes remotos). Todos os problemas sérios são encontrados mas perde na detecção de consistência

- Percurso cognitivo falha em identificar problemas gerais e recorrentes pois observa detalhes menores diretamente ligados à execução de uma tarefa. Foi

o que apresentou melhores resultados quando utilizado por não especialistas em usabilidade mas desenvolvedores de software.

Cada método tem pontos fortes e pontos fracos e ainda não se tem resultados substanciais de pesquisas comparando os métodos para que se possa efetivamente dizer qual é o melhor e em qual situação. Certamente cada situação de projeto irá requerer uma forma de avaliação. Acreditamos que o principal problema é a não disponibilidade de especialistas em usabilidade em situações de pequenas empresas desenvolvedoras de software (Chan e Rocha, 1996). Mas tendo como parâmetro que qualquer avaliação é melhor que nenhuma, ela deve ser feita de qualquer forma e a conseqüente geração de conhecimento em usabilidade será uma das melhores conseqüências.

Quanto à Web não existem ainda metodologias específicas para avaliação de Web *sites*. Conforme visto do Capítulo 1 *guidelines* começam a ser definidas (Nielsen, 1999) e irão possibilitar a definição de métodos de avaliação que levem em conta as especificidades desses design. Certamente os parâmetros de usabilidade permanecem e no decorrer do livro muitos exemplos extraídos de sistemas da Web foram mencionados, mas existe o componente da informação que certamente deve ser considerado em um processo de avaliação. Spool *et al*. (1999) relata alguns estudos de caso de avaliações de Web *sites*. Eles consideram que informação é o tema central e focalizam seus estudos em avaliar quanto um *site* é bom em prover informação às pessoas de forma a auxiliá-las a tomarem decisões. E concluem que quanto mais um *site* ajuda uma pessoa a encontrar a informação que está procurando mais usável ele é. De modo geral, são ainda resultados parciais quanto a usabilidade de sistemas baseados na Web e que ainda deverão ser mais aprofundados e formalizados.

REFERÊNCIAS:

Carrol, J. M., e Rosson, M. B. (1987) The paradox of the active user. Em J.M. Carrol (ed.) *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. Cambridge: Bradford Books/MIT Press

Desurvire, H. W. (1994) Faster, Cheaper!! Are usability Inspection Methods as Effective as Empirical Testing? Em J. Nielsen (ed.) *Usability Inspection Methods.* John Wiley, New York

Dix, A., Finlay, J., Abowd, G., Beale, R. (1998) *et al.* . *Human-Computer Interaction*. Prentice Hall Europe

Dumas, J. and Redish, J. (1993) *A Practical Guide to Usability Testing*. Ablex, Norwood, NJ

Egan, D. E. (1988) Individual differences in human-computer intercation. Em M. Helander (ed.). *Handbook of Human-Computer Interaction*. Amsterdam: Elsevier Science Publishers

Engelbeck, G. E. (1986) *Exceptions to generalizations: Implications for formal models of human-computer interaction*. Master Thesis. Department of Psychology, University of Colorado, Boulder

Fisher, G. (1991) Supporting learning on demand with design environments. *Proceedings of the International Conference on Learning Sciences* (Evanston, IL, August): 165-172

Gould, J. D., e Lewis, C. (1985) Designing for usability: Key principles and what designers think. *Communications of the ACM* 28, 3:300-311

Gould, J., Boies, S. J., e Lewis, C. (1991) Making usable, useful, productivity-enhancing computer applications. *Communications of the ACM* 34, 1: 74-85

Greenbaum, J., Kyng, M. eds. (1991) *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Lawrence Erlbaum Assoc.

Hartson, H. R., Castilho, J. C., Kelso, J. (1996) Remote Evaluation: The Network as an Extension of the Usability Laboratory. Disponível na Web em http://www.acm.org/sigchi/chi96/proceedings/papers/Hartson/hrh_texto.htm. Consulta 09/03/2000

Hix, D. and Hartson, H. R. (1993) *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley and Sons, New York

Kahn, M.J., Prail, A. (1994) Formal usability Inspections. Em J. Nielsen (ed.) *Usability Inspection Method.* John Wiley, New York

Karat, C. (1994) A Comparison of user Interface Evaluation Methods. Em J. Nielsen (ed.) *Usability Inspection Methods.* John Wiley, New York

Lewis, C. (1982) Using the 'thinking-aloud' method in cognitive interface design. *IBM Research Report* RC9265 (#40713), IBM Thomas J. Watson Research Center, Yorktown Heights, NY

Lewis, C., Polson, P., Wharton, C. and Rieman, J. (1990) Testing a walkthroug methododology for theory-based design of walk-up-and-use interfaces. *Proceedings ACM CHI'90 Conference* (Seattle, WA, April 1-5):235-242

Molich, R. and Nielsen, J. (1990) Improving a human-computer dialogue. *Communications of the ACM* 33,3: 338-348

Monk, A., Wright, O., Haber, J., Davenport, L. (1993*) Improving your Human-Computer Interface: A Practical Technique*. New York: Prentice-Hall

Nielsen J. (1989) Usability engineering at a discount. Em G. Salvendy *et al*. (eds.). *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*. Amsterdan:Elsevier Science Publishers

Nielsen, J. (1992) Finding usability problems through heuristic evaluation *Proceedings ACM CHI'92 Conference* (Monterey, CA, May 3-7): 373-380

Nielsen, J. (1993) *Usability Engineering*. Academic Press, Cambridge, MA

Nielsen, J. (1994) Heuristic Evaluatin. Em J. Nielsen (ed.) *Usability Inspection Methods.* John Wiley, New York

Nielsen, J. (1999) *Design Web Usability*. New Riders Publishing

Polson, P. e Lewis, C. (1990) Theory-based design for easily learned interfaces. *Human Computer Interaction* 5, 2&3:191-220

Polson, P., Lewis, C., Rieman, J. e Wharton, C. (1992a) Cognitive Walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36:741-773

Preece, J., Sharp, H., Benyon, D., Holland, S., Carey, T. (1994) *Human-Computer Intercation*, Addison-Wesley

Spool, J., Scanlon, T. Schoeder, W. Snyder, C. e DeAngelo, T. (1999) *Web Site Usability: A Designers Guide.* Morgan Kaufmann Publishers

Romani, L. S. e Baranauskas, M.C.(1998) Avaliação heurística de um sistema altamente dependente do domínio. *Relatório Técnico* IC-98-26 , July. Disponível na Web em: http://www.dcc.unicamp.br/ic-tr-ftp/ALL/Titles.html

Chan, S. e Rocha, H.V. (1996) Estudo comparativo de métodos para avaliação de interfaces homem-computador.*Relatório Técnico* IC-96-05, September. Disponível na Web em: http://www.dcc.unicamp.br/ic-tr-ftp/ALL/Titles.html

Schneiderman, B. (1998) *Design the User Interface: Strategies for Human-Computer Interaction*. Addison Wesley Longman, Inc.

Wharton, C., Rieman, J., Lewis, C., Polson, P. (1994) The Cognitive Walkthrough: A Practitioner´s Guide. Em J. Nielsen (ed.) *Usability Inspection Methods.* John Wiley, New York

Whitefield, A., Wilson, F., and Dowel, J. (1991) A framework for human factors evaluation. *Behaviour & Information Technology* 10, 1 (January-February), 65-79

Wright, P. C. and Monk, A. F. (1991) The use of think-aloud evaluation methods in design. *ACM SIGCHI Bulletin* 23, 1:55-71

CAPÍTULO 5

PERSPECTIVAS DAS INTERFACES HUMANO-COMPUTADOR - O ADVENTO DE UMA NOVA COMPUTAÇÃO

The real question before us lies here: do these instruments further life and enhance its values, or not?
Munford, *Technics and Civilization*, 1934
apud Schneiderman, 1998

INTRODUÇÃO

Tem-se muito a aprender, olhando as visões de futuro ocorridas no passado não muito distante, especialmente na área de tecnologia. Cientistas e futurólogos cometeram erros grosseiros em sua visão de futuro das tecnologias que moldaram o século XX (Veja, 1999):

> *O mercado mundial terá lugar para cinco computadores. (Thomas Watson, fundador da IBM, 1943);*
>
> *O fonógrafo não tem nenhum valor comercial. (Thomas Edison, inventor do toca-discos, em 1880);*
>
> *É uma invenção maravilhosa, mas não passa de um brinquedo (Gardiner Hubbard, sogro de Alexander Graham Bell, o inventor do telefone, em 1876);*
>
> *Em seis meses a televisão some do mercado. As pessoas vão se cansar de ficar sentadas diante de uma caixa de madeira. (Darryl F. Zanuck, presidente da 20th Century Fox, em 1946);*
>
> *Não existe nenhuma razão que justifique uma pessoa ter um computador em casa. (Ken Olson, fundador da Digital Equipment Corporation, a maior competidora da IBM em 1977).*

Essas lições nos ensinam que, muito mais do que um exercício de adivinhação, pensar o futuro é um exercício de construção coletiva de visão de mundo. Ao contrário do que somos ingenuamente levados a pensar a técnica como um instrumental a nosso serviço, Pierre Lévy coloca a técnica como *um dos mais importantes temas filosóficos e políticos de nosso tempo* (Lévy, 1993, p.7).

O telégrafo e o telefone serviram para pensar a comunicação; os servomecanismos, a teoria da informação e a visão cibernética do mundo. Os produtos da técnica moderna não servem apenas ao uso instrumental, mas participam da instituição de mundos percebidos. O computador tornou-se hoje, um desses dispositivos técnicos pelos quais percebemos o mundo, no plano empírico e também no plano transcendental. Para Lévy, técnica, política e projetos culturais misturam-se de forma inextrincável. Certas técnicas de armazenamento e de processamento das representações tornam possíveis e condicionam evoluções culturais.

Neste capítulo estaremos analisando a História dos avanços tecnológicos da última metade do século de modo a podermos pensar em o que queremos construir em termos de desenvolvimento computacional para o próximo século e nesse contexto

reafirmar a relevância de estarmos atentos aos resultados advindos dos estudos sobre o processo de interação humano-computador.

UM POUCO DE HISTÓRIA

Em 1945, Vannevar Bush (1945) apresentou a proposta da Memex, uma biblioteca para estender a memória através do acesso a um amplo conjunto de patentes, artigos científicos, citações legais. Memex é uma máquina que teria a capacidade de armazenar informação tanto textual quanto gráfica de tal forma que qualquer peça de informação poderia ser arbitrariamente ligada a qualquer outra peça. Nas próprias palavras de Bush(1945):

[...] He[the user] can add marginal notes and comments, taking advantage of one possible type of dry photography, and it could even be arranged so that he can do this by a stylus scheme, such as is now employed in the telautograph seen in railroad waiting rooms, just as though he had the physical page before him. All this is conventional, except for the projection forward of present-day mechanisms and gadgetry. It affords an immediate step, however, to associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the Memex. The process of tying two items together is the important thing.

Memex também oferecia ao usuário a capacidade de criar informação, que poderia ser recuperada posteriormente, relativa a todos os *links* percorridos. Os trechos seguintes extraídos do artigo de Bush (1945)[1] dão uma melhor idéia de sua proposta:

> *The real heart of the matter of selection, however, goes deeper than a lag in the adoption of mechanisms by libraries, or a lack of development of devices for their use. Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. When data of any sort are placed in storage, they are filled alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. It can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.*

[1] Preferimos deixar no original em inglês dado o valor histórico que creditamos ao artigo

The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature.

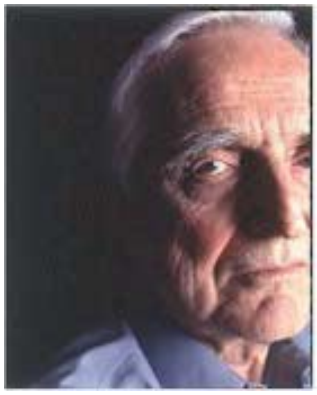
Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. In minor ways he may even improve, for his records have relative permanency. The first idea, however, to be drawn from the analogy concerns selection. Selection by association, rather than by indexing, may yet be mechanized. One cannot hope thus to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage.

Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, ``memex'' will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

It consists of a desk, and while it can presumably be operated from a distance, it is primarily the piece of furniture at which he works. On the top are slanting translucent screens, on which material can be projected for convenient reading. There is a keyboard, and sets of buttons and levers. Otherwise it looks like an ordinary desk.

In one end is the stored material. The matter of bulk is well taken care of by improved microfilm. Only a small part of the interior of the memex is devoted to storage, the rest to mechanism. Yet if the user inserted 5000 pages of material a day it would take him hundreds of years to fill the repository, so he can be profligate and enter material freely.

Vinte anos depois de Bush, Licklider (1965) trouxe a idéia de biblioteca digital e reconheceu o potencial da teleconferência como meio de aproximar as pessoas.

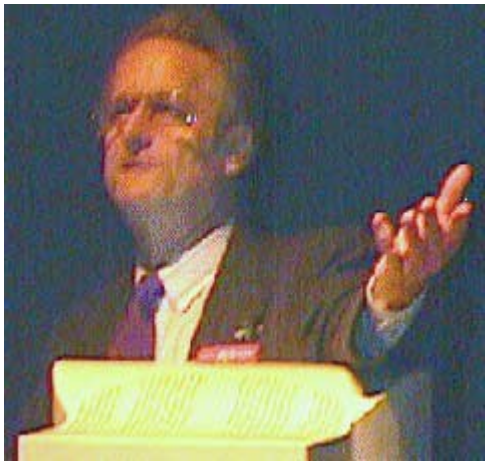


Engelbart (1968), o inventor do mouse, vê o computador como um meio de amplificar o intelecto humano, e propõe seu sistema NLS (oN Line System) que mantinha um "jornal distribuído" com mais de 100.000 artigos, reportagens, memoriais e referências cruzadas.

Pela ampla polêmica gerada e pelas idéias propostas o projeto Xanadu merece destaque.
Theodor Holm Nelson, um escritor, diretor de cinema, e designer de software concebeu a idéia do Xanadu em 1981. Em suas próprias palavras ele explica o que é Xanadu (Gromov):

…….

1. *Xanadu is a system for the network sale of documents with automatic royalty on every byte.*
2. *The transclusion feature allows quotation of fragments of any size with royalty to the original publisher.*
3. *This is an implementation of a connected literature.*
4. *It is a system for a point-and-click universe.*
5. *This is a completely interactive docuverse.*



Analisando Xanadu, Andrew Pam (Pam) explica *transclusion* como:

> *"Transclusion" is a term introduced by Ted Nelson to define virtual inclusion, the process of including something by reference rather than by copying. This is fundamental to the Xanadu design; originally transclusions were implemented using hyperlinks, but it was later discovered that in fact hyperlinks could be implemented using transclusions! Transclusions permit storage efficiency for multiple reasonably similar documents, such as those generated by versions and alternates as discussed above. "*

No esquema do Xanadu, uma base universal de documentos (*docuverse*) deveria permitir endereçar qualquer substring de um documento por qualquer outro documento - *"This requires an even stronger addressing scheme than the Universal Resource Locators used in the World-Wide Web."* (De Bra)

Adicionalmente, Xanadu poderia permanentemente guardar todas as versões de todos os documentos, eliminando a possibilidade de quebra de um link e a tão familiar e indesejada mensagem: *404-Document Not Found error*.

Xanadu somente manteria por inteiro a última versão do documento. Versões anteriores poderiam ser dinamicamente reconstruídas a partir da versão atual utilizando para isso um sofisticado sistema de versões que manteria um registro das modificações de cada geração do documento.

Em um poema de Samuel Taylor Coleridge, intitulado Kubla Khan, Xanadu é um *"magic place of literary memory"* onde nunca nada é esquecido (Gromov , Zeltser)

Xanadu nunca foi implementado (bem como Memex). Wolf (Wolf) escreve:

> *Xanadu, a global hypertext publishing system, is the longest-running vaporware story in the history of the computer industry. It has been in development for more than 30 years. This long gestation period may not put it in the same category as the Great Wall of China, which was under construction for most of the 16th century and still failed to foil invaders, but, given the relative youth of commercial computing...*

Mas, além da crítica, ele acrescenta:

> *"Nelson's writing and presentations inspired some of the most visionary computer programmers, managers, and executives - including Autodesk Inc. founder John Walker - to pour millions of dollars and years of effort into the project. Xanadu was meant to be a universal library, a worldwide hypertext publishing tool, a system to resolve copyright disputes, and a meritocratic forum for discussion and debate. By putting all information within reach of all people, Xanadu was meant to eliminate scientific ignorance and cure political misunderstandings."*

Depois de anos de frustração, Ted Nelson aceitou um convite do Japão em 1994, e fundou o Sapporo HyperLab onde continuou sua pesquisa no Xanadu. Atualmente ele é professor de Informação Ambiental na Shonan Fujisawa Campus of Keio University.

Hoje em dia a WWW usa hipertexto para relacionar milhões de documentos e tornou-se um paradigma de interfaces. Na Web, o básico da visão de Vannevar Bush e Ted Nelson e dos pesquisadores que aceitaram seus desafios têm se tornado realidade. Mas certamente a Web que conhecemos hoje ainda não captou a amplitude das idéias de seus pioneiros embora inovações apareçam com muita frequência.

Ao contrário do que se é levado a pensar, as revoluções tecnológicas são rápidas apenas do ponto de vista da linha de tempo da civilização. Todas as novas tecnologias levam um tempo longo até afetar a vida das pessoas comuns, conforme mostra Norman (1998): a imprensa levou 100 anos a espalhar-se pela Europa; não é muito diferente o tempo de impacto das tecnologias da aviação, do telefone, do fax, etc. na vida das pessoas. O computador tem 50 anos e o computador pessoal mais de 20. O inicio da Internet aconteceu há 30 anos atrás, por incrível que pareça. Inovações tecnológicas são, de certa forma, limitadas no tempo pelas mudanças sociais, organizacionais e culturais que provocam.

O CICLO DE VIDA DA TECNOLOGIA

Norman (1998) explica o desenvolvimento de todo tipo de tecnologia por um ciclo de vida que evolui do nascimento à maturidade, alterando suas características em função do tipo de consumidor dessa tecnologia. Durante esse ciclo de vida, a categoria de usuário varia começando com o que ele chamou "*early adopters*" até os "*late adopters*". No início do ciclo de vida de uma nova tecnologia, seus usuários são entusiastas da tecnologia que ajudam o novo produto a ganhar poder e aceitabilidade. A engenharia do produto é a base dessa fase; a cada produto novo lançado, o que conta são melhorias tecnológicas: rapidez, maior poder. A tecnologia é o motor dessa fase, guiada pelo marketing dirigido às inovações tecnológicas introduzidas no produto. Em sua maturidade, a estória muda dramaticamente. A tecnologia é um pressuposto, é considerada infra-estrutura, e outra categoria de usuários dirige o desenvolvimento do produto. Os *late adopters*, são a grande maioria de usuários, que esperam o amadurecimento da tecnologia para se apropriarem dela e, ao contrário dos *early adopters*, buscam conveniência em lugar de superioridade tecnológica.

A fase madura de um produto deve ser dirigida, portanto, pelas necessidades do usuário comum, pessoas que querem os benefícios da tecnologia sem, entretanto, serem aborrecidas por ela. O desenvolvimento do produto, agora, deve passar de centrado na tecnologia para centrado no usuário. Como resultado disso, as organizações que desenvolvem essa tecnologia devem mudar também. A transição da indústria dirigida à tecnologia para a indústria dirigida para as necessidades do *late adopter* não é trivial e é esse o momento em que a indústria de software se encontra atualmente.

A literatura que estuda a maneira como idéias e produtos inovadores chegam à sociedade, classifica as pessoas que são o alvo da inovação em cinco categorias: os inovadores, os que logo adotam, a maioria inicial, a maioria final e os que "ficam para trás" (Rogers e Moore, apud Norman, 1998). A Figura 5.1 ilustra a mudança de categorias de usuários ao longo do ciclo de vida da tecnologia.

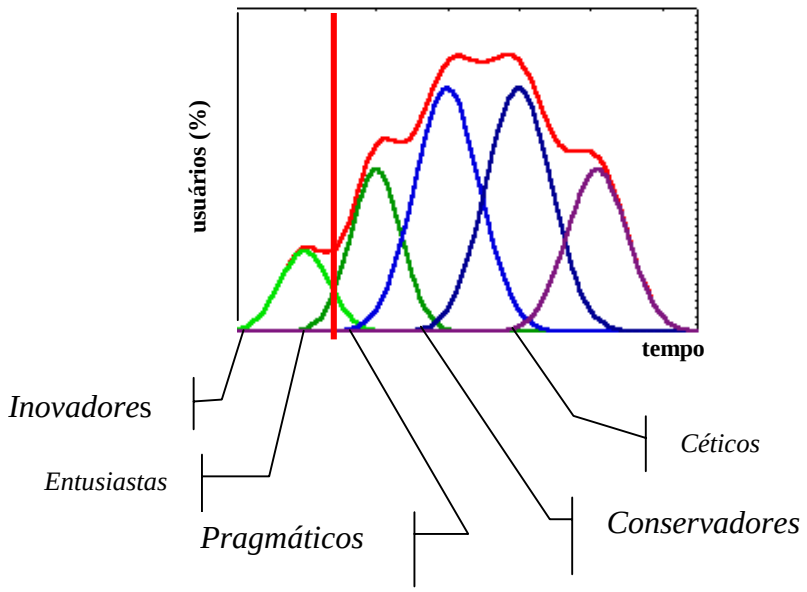


FIGURA 5.1 AS CATEGORIAS DE USUÁRIOS AO LONGO DO CICLO DE VIDA DA TECNOLOGIA

A grande maioria das pessoas é pragmática e conservadora; elas têm uma visão realista do mundo, esperam até que os preços caiam e a tecnologia se estabilize. Esses usuários querem conveniência, confiabilidade e valor. Eles não querem grandes rupturas com sua prática corrente. À medida que a tecnologia caminha para sua maturidade, a natureza do produto, não apenas o seu marketing, deve mudar. As estratégias para lidar com os consumidores iniciais – os entusiastas da tecnologia – é muito diferente das estratégias para lidar com a maioria na fase madura da tecnologia.

Para Moore, conforme a Figura 5.1 ilustra, há uma diferença bastante significativa entre os dois tipos de consumidores, que marca uma ruptura na prática corrente das indústrias com relação à concepção do produto. A indústria dos computadores digitais está agora cruzando essa região de ruptura no gráfico. Em conseqüência ela tem que passar pela transformação da indústria movida pela tecnologia para a indústria movida pelas necessidades e interesses da grande maioria das pessoas.

Na fase madura de determinado produto, tecnologia e conhecimento do usuário devem dar sustentação ao seu desenvolvimento. Tecnologia deve estar apropriada às funções e desempenho requeridos e a um custo razoável; conhecimento do usuário requer considerável atenção às suas necessidades, habilidades e processos de pensamento. Um produto tecnológico não será bem sucedido se for muito lento ou limitado em suas funcionalidades e capacidades mesmo que seja fácil de usar e

entender. Nesta categoria se encaixam muitos dos "*toy problems*", protótipos desenvolvidos em escala pequena, que não se ajustam à complexidade dos problemas reais. Por outro lado, o produto pode possuir uma tecnologia maravilhosa, mas ser muito difícil de usar na prática corrente dos usuários do domínio. Sistemas avançados de informação como, por exemplo, os GIS (Geographical Information Systems) passam por este tipo de dificuldade. Finalmente, um produto tecnológico poderá falhar se for muito caro, mesmo que tenha tecnologia adequada e seja fácil de usar.

Seguramente, o conceito de interface bem como as metodologias e métodos de design têm evoluído como um reflexo do ciclo de vida da tecnologia de computadores. Este entendimento do ciclo de vida da tecnologia dos computadores digitais fornece contexto para novos entendimentos do conceito de interface, conforme será discutido na próxima seção.

DA COMPUTAÇÃO PARA A COMUNICAÇÃO

De modo a se ter alguma segurança com relação ao futuro é preciso tirar vantagens do passado e do futuro: identificar algumas trajetórias que estão atualmente sob questionamento, e olhar para onde o futuro aponta. Certamente esse método não funciona sempre, haja vista o quanto estamos sempre nos surpreendendo. Mas pelo menos ele nos dá alguns pontos de partida.

Escolhemos os "visionários" citados no início deste capítulo porque eles, com suas propostas, nos apontavam a direção da computação para a comunicação, em uma época em que o computador era visto como simplesmente uma máquina para computação. O computador poderia fazer pouco trabalho de uma tarefa tal como calcular trajetórias de balísticas ou quebrar códigos, com um trabalho prévio de cálculos feitos por uma equipe de "computadores humanos" (Winograd, 1997). Quando a Internet surgiu, a rede foi vista primeiramente como uma ferramenta para facilitar a computação remota - a tarefa computacional poderia ser feita usando um computador distante fisicamente da pessoa que necessitava o trabalho e que controlaria sua execução.

Com o recente desenvolvimento de aplicações baseadas na Internet, tornou-se mais que claro que o computador não é uma máquina cujo principal propósito é conseguir que uma tarefa de cálculo complexo seja feita. O computador, com seus novos periféricos e redes, é uma máquina que provê novo meio para as pessoas se comunicarem com outras pessoas. O grande movimento atual na área computacional advém da exploração de novas capacidades de manipular e comunicar toda espécie de informação em todos os tipos de mídias, atingindo novas audiências de formas impensadas antes do computador. Memex e Xanadu certamente apontavam essa direção.

Também se considerarmos os computadores pessoais a história é a mesma. O conjunto de aplicações que dominam o mercado atualmente consiste primariamente de ferramentas de comunicação: processadores de texto, programas de apresentação, *email*, compartilhamento de arquivos, etc. Até mesmo uma aparente exceção - a planilha de cálculo - é usada prioritariamente para comunicar resultados que para calculá-los (Winograd, 1997).

De alguma forma isso não precisa causar surpresa, se observamos a natureza humana. Pessoas são primariamente interessadas em outras pessoas, e são altamente motivadas a interagir com elas em qualquer mídia que esteja disponível. Novas tecnologias, do telégrafo a Web, têm expandido nossa habilidade para nos comunicarmos amplamente, com flexibilidade e eficiência. Essa urgência de comunicação continua dirigindo a expansão da tecnologia com o advento da conectividade sem fio, a banda larga, imagem 3-D, e muito mais ainda não imaginado.

Dentro da indústria de computação estamos também vendo uma nova ênfase em comunicação, refletida na preocupação com o conteúdo. As companhias que fizeram extensivos investimentos no desenvolvimento de sistemas computacionais estão mudando seu olhar da direção do que a "máquina faz" para o que a "máquina comunica". Como exemplo, podemos considerar a trajetória da Microsoft, que começou com sistemas operacionais, expandiu para o universo de aplicações de software e atualmente está se movendo para a arena do conteúdo juntando esforços com a NBC e no Brasil com a gigante Rede Globo. Fabricantes de estações de trabalho como a Silicon Graphics estão indo na direção da indústria do entretenimento, e fabricantes de chips como a Intel abriram recentemente um novo laboratório de pesquisa com interesse em questões de mais alto-nível, como comunicação humana e uso de computadores domésticos.

Outro ponto relevante a ser considerado é a distância que usuários atualmente têm da máquina. Se perguntarmos a usuários comuns que tipo de computador estão usando, não é raro ouvirmos a resposta - Windows. Ao mesmo tempo, se perguntamos a um usuário mais especializado se ele tem um computador com transistor NMOS ou CMOS ele também não saberá responder. Com a Web, o distanciamento da máquina caminha a passos largos. A experiência deixa de ser a máquina, ou o programa, e sim a entrada no denominado *cyberspace* habitado por textos, gráficos, vídeos, animações compondo catálogos, propagandas, estórias animadas e galerias de arte.

A palavra *cyberspace* denota uma nova computação e tem se tornado um novo cliché. Na verdade *cyberspace* como um termo derivado de espaço reflete uma metáfora. Um espaço não é somente um conjunto de objetos e atividades, é um meio no qual a pessoa atua, experimenta e vive.

A idéia tradicional de interface nos leva a focar em duas entidades, a pessoa e a máquina, e em um espaço que reside entre elas. Mas além da interface, nós operamos dentro de um *"interspace"* que é habitado por múltiplas pessoas, estações de trabalho, servidores, e outros dispositivos em uma complexa rede de interações. Portanto, no design de novas aplicações e sistemas, nós não estamos somente provendo novas ferramentas para trabalhar com objetos dentro de um mundo que já existe - estamos criando novos mundos. Sistemas computacionais estão se tornando um meio para a criação de virtualidades: mundos nos quais usuários de software percebem, atuam, e respondem a novas experiências.

Dentro dos próximos anos, em uma onda já iniciada, a crescente importância do design de espaços para comunicação e interação humana irá levar à expansão de aspectos na computação que irão focar nas pessoas e não na máquina. Métodos, habilidades e técnicas com essa preocupação em aspectos humanos são geralmente estranhos às linhas centrais da Ciência da Computação e muitos autores advogam que um novo campo de atuação será criado com base nos princípios de IHC, descritos neste livro. Esses pontos de vista serão discutidos no final deste capítulo.

Ao pensarmos em futuro, também gostaríamos de questionar a importância de nos atermos aos benefícios sociais que desejamos com essa nova computação que desponta. Certamente, uma boa medida do progresso alcançado com a nova tecnologia computacional, além de medida de sua efetiva relevância, é a porcentagem da população com acesso aos serviços oferecidos pela tecnologia computacional, por exemplo os serviços oferecidos pela Web: *email* eletrônico, educação a distância, redes de interação, etc. (Anderson *et al*., 1995). Possibilitar o acesso universal à essa nova computação e todo o seu potencial deve ser a premissa básica de qualquer futuro desenvolvimento. E sem dúvida esse é um dos princípios básicos de toda a área de IHC.

ACESSO UNIVERSAL À TECNOLOGIA COMPUTACIONAL

Estamos sendo inundados na imprensa cotidiana por noticias sobre a Internet, seus benefícios, os serviços que oferece, as pessoas que estão enriquecendo desenvolvendo *sites*, etc. Além disso, começa a aparecer no Brasil o acesso grátis à Internet, o que vem causando muita polêmica, mas sem dúvida está revolucionando a área. Nos perguntamos: *Qual o alcance dessas notícias? Quantas pessoas não têm nem idéia do que se está falando?*

Então sem dúvida, um esforço tem que ser feito na direção do acesso universal. Prover eletricidade, hardware e comunicação é somente o começo. Aplicações e serviços devem ser repensados para atender às diferentes necessidades dos usuários excluídos. É preciso pensar, por exemplo, em como um sistema de *email* precisaria

ser implementado para atender aos usuários com deficiências de escrita ou leitura ao mesmo tempo em que os ajudaria a superar essas dificuldades.

Como deveria ser um sistema de votação eletrônica, declaração de imposto de renda, registro de automóveis, registro de crimes, se o acesso universal fosse assumido e toda a diversidade de uma população precisasse ser considerada?

Para saber se as interfaces atuais são adaptadas aos excluídos, poder-se-ia colocar a seguinte questão: um excluído (por exemplo, um idoso, um cego, ou simplesmente um sujeito que nunca teve contato com essas novas tecnologias de informação e comunicação) poderia facilmente aprender a usar um editor de textos, um navegador na Internet ou simplesmente retirar dinheiro de um caixa automático? Nos dias de hoje, a resposta seria quase sempre negativa, pois as especificidades dessas populações somente em raríssimos casos foram consideradas para o projeto de interfaces humano-computador (Cybis e Michel,1999).

A interface que temos hoje é essencialmente dependente do bom funcionamento de nossos sistemas perceptual cognitivo e motor. Fazemos uso principalmente da visão – para leitura da tela, e do sistema motor – para uso do teclado e do mouse. Pessoas portadoras de deficiências, nesses sistemas tem o acesso à informação tremendamente dificultado. Algumas pessoas podem não ser capazes de ver, ouvir, mover-se ou processar certos tipos de informação; podem não ser capazes de operar o teclado ou o mouse. Graças a alguns esforços isolados têm surgido artefatos de software e hardware especiais para categorias de necessidades especiais dessas pessoas. Podemos citar, por exemplo, alguns artefatos criados para deficientes visuais (cegos ou portadores de visão subnormal): o DOSVOX[2] é um sistema de software comercial constituído de um conjunto de 60 programas que "falam" ao usuário durante o uso de determinadas aplicações como *telnet*, *ftp*, navegadores, editores de texto, etc. Enquanto a comunicação do usuário para com o computador continua sendo feita via teclado, a saída de informação do computador para o usuário é falada em língua portuguesa. ViaVoice[3] é um outro exemplo desse tipo de sistema, com entrada por voz ou teclado e saída através de textos falados.

Sistemas para deficientes visuais envolvendo hardware, software e outros tipos de equipamento podem ser classificados em 3 tipos: sistemas amplificadores de telas, sistemas de saída de voz (como nos exemplos citados), e sistemas de saída em Braille – impressoras e terminais de acesso. Outras tecnologias despontam, envolvendo reconhecimento de voz, *scanners* e amplificadores de imagem. A falta de padronização entre fabricantes, herança da tecnologia vigente, é um problema a enfrentar especialmente nesses artefatos para necessidades especiais.

[2] UFRJ (http://nce.ufrj.br/aau/dosvox)

[3] IBM (http://www.ibm.com/speech/demo)

Nos últimos cinco anos, alguns fabricantes têm mostrado preocupação em aumentar a acessibilidade a seus produtos, especialmente depois da popularização da Internet. Acessibilidade, nesse contexto, é sinônimo de facilidade de aproximação. Diversas formas de tratar necessidades especiais do usuário têm sido incorporadas ao hardware ou software ou nas principais aplicações, facilitando o acesso de todos, ou têm sido criadas através de utilitários que modificam o sistema, ou aplicações especiais para alguns tipos de deficiências. Mesmo tendo seus mecanismos perceptuais, motores e cognitivos funcionando perfeitamente, muitas vezes o usuário pode encontrar-se em situações onde o uso dos olhos, ouvidos e mãos estejam comprometidos executando outras funções, ou o usuário pode estar usando um monitor que não processa imagem, ou simplesmente pode estar usando uma versão antiga de navegador.

Aspectos de acessibilidade em páginas Web consideram a variedade de contextos de interação que podem estar relacionados a diversos tipos de situações dos usuários com deficiência ou não. Entre esses cidadãos encontra-se também a população de idosos. Com o avanço da idade o cristalino do olho torna-se amarelado e opaco; como conseqüência menos luz entra nos foto receptores e a sensibilidade ao contraste diminui. Regras simples podem contemplar as dificuldades dessa categoria de usuários, como por exemplo:

Fazer o texto maior ou ajustável e usar cores de muito contraste;
Evitar uso de fontes de linhas finas ou com muito detalhe.

Adaptações na apresentação dessas páginas quase não oneram o custo final e alcançam um maior número de pessoas, melhorando também o desempenho de usuários não deficientes. Já existem normas disponíveis na Internet com recomendações de acessibilidade que atendem tanto aos usuários de computadores padrão como usuários que estejam interagindo a partir de um sintetizador de voz, de um mostrador Braille ou sem monitor de vídeo. Algumas alterações simples como fornecer equivalentes textuais para recursos multimídia, colocando legendas nas imagens, por exemplo; ou assegurar que o esquema de cores utilizado não cause dificuldades à visualização (aumentando o contraste) podem facilitar o acesso de todos, independentemente dos que podem mais ou menos em função do estado de seus sistemas perceptual, cognitivo ou motor. Diretrizes para a confecção de páginas Web-acessíveis tem sido divulgadas pelo W3C *The World Wide Web Consortium.*

A avaliação de acessibilidade de páginas Web pode ser feita através da ferramenta Bobby[4] que fornece um relatório indicando problemas de acessibilidade ou de incompatibilidade de navegadores, encontrados na página analisada. É um serviço gratuito cuja missão é expandir oportunidades para pessoas com necessidades especiais através de usos inovadores da tecnologia de computadores. Em vários países como EUA, Austrália, Portugal, país de origem do teste, todos os *sites* de

[4] http://www.cast.org/bobby/

órgãos públicos devem satisfazer os parâmetros de acessibilidade. Um símbolo especial é usado no *site*, para indicar a aprovação dele para testes de acessibilidade.

O surgimento de novas tecnologias trás consigo alto poder de inclusão ou exclusão das pessoas no seu meio. A acessibilidade de páginas Web representa esforços para tornar a sociedade da informação e do conhecimento acessível também aos cidadãos com necessidades especiais.

Um exemplo, que merece ser mencionado a título de ilustração da problemática de exclusão, é o projeto brasileiro de criação de urnas eletrônicas que foi concebido especificamente para suprimir as possibilidades de fraudes e diminuir a duração do processo de contagem de votos. Cybis e Michel (1999) relatam resultados de uma pesquisa para verificar o impacto da utilização da urna em sete importantes cidades do estado de Santa Catarina, nas eleições de 1996. O autor afirma que o sistema de urna eletrônica só permite aos cegos treinados em Braille (15% dos cegos) a realização do voto sem erros. Considerando-se as pessoas de baixa visão, onde a maior parte não conhece Braille, foi impossível utilizar a interface da urna eletrônica. Para os idosos alfabetizados e conhecedores do processo de voto, foi possível constatar que uma boa parte deles concentravam toda sua atenção no teclado e que a tela era praticamente ignorada. Dado o número de informações alfa numéricas tanto sobre o teclado como na tela, a chance de sucesso para o idoso analfabeto era evidentemente nula: todos os idosos que obtiveram sucesso no voto tiveram ajuda dos mesários. Pode-se, portanto, concluir que os problemas enfrentados por certos eleitores desabituados à operação de sistemas eletrônicos e com dificuldades especiais deveriam ter levado os projetistas da urna eletrônica a tomarem cuidados adicionais no projeto de sua interface. Tanto para cegos como para os idosos, o número elevado de erros constatados e a duração média dos votos permitem concluir que a urna eletrônica não proporcionou a realização normal de seu direito de cidadãos.

Votar deve ser um ato civil natural, e a tecnologia não deve se colocar como obstáculo. Uma urna eletrônica deveria facilitar essa tarefa ao eleitor, garantindo-lhes votarem em seus candidatos, com o mínimo de erros e incidentes. Certamente esses objetivos não foram alcançados pela urna eletrônica, o que pode ser explicado pela abordagem puramente tecnológica de sua concepção (Cybis e Michel, 1999).

Talvez se pudesse iniciar fazendo o redesign de interfaces para simplificar tarefas comuns. Poder-se-ia prover novos métodos de treinamento e ajuda de forma que usar o computador seja uma oportunidade de satisfação ao invés de um desafio frustrante. Aprendizado evolutivo, com interfaces estruturadas por nível de experiência e na qual os principiantes teriam sucesso em tarefas simples e que teriam um caminho de crescimento para o uso em tarefas mais complexas. Com milhões de novos usuários, estratégias para filtrar mensagem eletrônica, encontrar informação e conseguir assistência *online* serão necessárias.

Interfaces com facilidades de internacionalização onde através de controles, os usuários possam especificar sua língua, suas unidades de medida, seu nível de habilidade, etc. Portabilidade para hardware não padrão, adaptação a diferentes tamanhos e resoluções de telas ou diferentes velocidades de modems, e o design para portadores de deficiências ou idosos deverão ser práticas comuns.

Acesso universal é uma decisão política. Políticas regulamentadoras para telefones, televisão, satélites, etc., têm tido sucesso em criar acesso quase universal a essas tecnologias, mas o design e serviços computacionais e suas implicações econômicas aparentemente precisam de revisão de modo a alcançar uma audiência mais ampla.

E o suporte para garantir a universalidade do acesso à educação? Computadores vêm alterando a educação profundamente e aplicações educacionais inclusive as que visam educação a distância precisam ser analisadas.

Shneiderman (1998) faz uma abordagem bastante interessante onde combina educação com benefício social e experiências autênticas para ensinar estudantes a como participar em grupos de trabalho, sistemas políticos e comunidades. Tecnologia de informação é poderosa quando possibilita aos estudantes colaborar efetivamente no sentido de construir resultados significativos que beneficiem pessoas fora da sala de aula. Ele empresta de Denning (1992) a abordagem *relate-create-donate* que empurra os estudantes para aprenderem fundamentos relevantes, e os encoraja a perseguir objetivos práticos.

Quanto à educação a distância, muita tecnologia tem sido desenvolvida somente para oferecer suporte a educação a distância baseada na Web (Harasim *et al.*, 1995). Inúmeros ambientes foram e estão sendo desenvolvidos (AulaNet, Webct, Teleduc) mas seus design privilegiam facilitar o oferecimento de conteúdo em detrimento à cooperação e colaboração. Com isso, e sem uma análise crítica e redesign de tais ambientes, sofremos o sério risco da educação dar um passo atrás em sua atual evolução que prega o aprender a pensar, o aprender fazendo, o aprender a aprender - centrado no aluno e não somente em um conteúdo a ser transmitido.

Concluindo, em comunidades com problemas de moradia, fome, analfabetismo, certamente telefones ou computadores não são necessidades primárias, mas a tecnologia continua sendo uma necessidade como parte de um plano geral de desenvolvimento. Adaptar o design de sistemas usados em comunidades mais desenvolvidas para as mais carentes requer, sem dúvida alguma, uma engenharia criativa, além de recursos financeiros. Daí a mencionada decisão política ser tão relevante.

A PROBLEMÁTICA DA TECNOLOGIA ATUAL

Certamente é ingenuidade supor que o largo uso da tecnologia, universalmente acessível, somente trará benefícios. Existem razões legítimas para nos preocuparmos que a crescente disseminação dos computadores poderá levar à uma série de opressões - pessoais, organizacionais, políticas e sociais. Pessoas que têm medo de computadores têm boas razões para isso. Designers de sistemas computacionais precisam estar cientes desses problemas ao tomarem decisões e os estudos na área de IHC apontam direções para a prevenção da grande maioria deles.

Shneiderman (1998) enumera o que ele denomina das dez pragas da era da informação, dentre elas destacamos:

- **Ansiedade**
  Muitas pessoas evitam o computador ou o usam com grande ansiedade - medo de quebrar a máquina, medo de parecer tolo ou incompetente, ou mais geral, medo do novo. Essas ansiedades são reais e não podem ser ignoradas, e podem ser superadas com experiências positivas. *Pode-se construir interfaces de usuários e sistemas que venham a reduzir ou eliminar o atual alto grau de ansiedade experimentado por muitos usuários?*

- **Discriminação social**
  Pessoas sem habilidades com computadores têm novas razões para não terem sucesso na escola ou em conseguir um emprego. Existe uma grande disparidade na disponibilidade de computadores nas escolas, especialmente entre escolas privadas e públicas. Conseqüentemente o acesso à fontes de informação também é desproporcional. *Pode-se construir sistemas computacionais de tal modo que trabalhadores menos habilitados possam atuar de forma semelhante a expertos? Pode-se possibilitar treinamento e educação para todo membro capaz da sociedade?*

- **Impotência do indivíduo**
  Pessoas que tentam conseguir explicações sobre discrepâncias em seus extratos bancários, por exemplo, enfrentam sérios problemas, especialmente se têm problemas visuais ou auditivos, ou são portadores de outras deficiências físicas ou cognitivas. Sistemas computacionais interativos devem ser usados para aumentar a importância do indivíduo, ou seja, para prover tratamento personalizado, mas sua aplicação tem sido na direção contrária. *Como podemos fazer design de forma a que pessoas se sintam capazes e seguras?*

- **Fragilidade organizacional**
  Quanto mais dependentes de uma tecnologia complexa, mais frágeis se tornam as organizações. Recente exemplo é o amplo temor do *bug* do milênio onde

poderíamos ter uma parada geral, desde um simples elevador até todo o sistema telefônico e elétrico. *Como produzir design mais robusto que saibam lidar com os perigos?*

- **Invasão de privacidade**
  Quão seguro é fazer a declaração de imposto de renda pela Internet? Sem dúvida sistemas bem projetados têm o potencial de serem mais seguros que sistemas em papel, desde que exista alta preocupação com a proteção da privacidade. Isso nos remete ao conceito de aceitabilidade social que discutimos no Capítulo 1. *Pode-se projetar sistemas que aumentem ao invés de reduzir a proteção à privacidade?*

- **Falta de responsabilidade profissional**
  A complexidade da tecnologia provê amplas oportunidades para que organizações passem a responsabilidade de problemas para o computador. *Listagens emitidas por computadores podem se tornar mais confiáveis que a palavra de uma pessoa ou um julgamento profissional?* Sistemas complexos e confusos possibilitam a usuários e até designers culparem a máquina, mas com design melhorado, responsabilidade e crédito serão conseguidos, e serão aceitos por usuários e designers.

Dertouzos (1998), em recente livro, tenta fazer predições sobre quanto nossas vidas serão afetadas pela tecnologia no futuro. Ele avalia as novas tecnologias da computação e seus efeitos sobre nossas vidas, procurando entender a extensão, o significado e a profundidade da Revolução Informática para a humanidade como um todo. Para fazer isso ele primeiro examina uma série de "defeitos" - modos como a tecnologia dos computadores é mal empregada na atualidade, em função das falhas tecnológicas ou humanas. Afirma que corrigir esses defeitos é o primeiro passo para facilitar o uso dos computadores.

A análise de Dertouzos é profundamente econômica e tem portanto como princípio básico que a produtividade é a medida para avaliarmos as revoluções sócio-econômicas.

> *… a produtividade crescerá, quando os computadores e comunicações forem usados no Mercado de Informação para aliviar o trabalho cerebral das pessoas, assim como as máquinas industriais aliviaram o trabalho braçal.* (Dertouzos, 1998, p. 316).

> *A produtividade aumentará na Era da Informação assim como aumentou na Era Industrial, e pelas mesmas razões de antes: a aplicação de novos instrumentos para aliviar o trabalho humano. Ignorar a capacidade fundamental dos computadores, que é ajudar os seres humanos a fazer seu*

> *trabalho intelectual é na melhor das hipóteses maldade, e na pior, irresponsabilidade.* (Dertouzos, 1998, p. 318).

Dertouzos (1998) resume o que ele considera de errado com a tecnologia em algumas categorias de problemas dentre os quais destacamos:

- **Problema do vício**
  Relacionado com o fato das pessoas continuarem a fazer as coisas da maneira como se acostumaram a fazer antes dos computadores. Um exemplo é as pessoas fazerem uso de uma agenda eletrônica, mas ela ser somente o segundo passo no agendamento de compromissos, que continua sendo feito com base nas antigas agendas de papel e contatos pessoais entre os envolvidos. Segundo o autor, esse problema não é diretamente relacionado com a tecnologia e sim com a forma equivocada que as pessoas a usam. Não concordamos completamente com essa opinião, certamente grande parte desses são decorrentes da falta de confiabilidade, usabilidade e aceitabilidade dos produtos - as necessidades dos usuários e suas tarefas geralmente não são considerados no design. Falta adequação da ferramenta ao usuário e suas necessidades.

- **Problema do aprendizado excessivo**
  A quantidade de informação que acompanha qualquer software é enorme e tem a cada dia se tornado maior. Por que é esperado que tenhamos que fazer uso de um manual de quase 1000 páginas para usar um processador de textos? Esse é o problema do aprendizado excessivo - a expectativa de que as pessoas possam aprender e guardar um volume de conhecimento muito superior aos benefícios que podem obter com a utilização desse conhecimento. Usabilidade é o fundamental:

  > *Não tenho dúvidas de que passaremos a primeira metade do século XXI lutando para nos libertarmos dos manuais monstruosos, e tornando o uso dos computadores mais fácil e natural. A real facilidade de uso é algo básico na busca do aumento da produtividade* (Dertouzolos, 1998, p. 320)

- **Problema da perfeição**
  Tantos são os recursos oferecidos que qualquer um de nós se vê obrigado a gastar um tempo enorme, por exemplo, ajustando margens, mudando fontes e estilos, escolhendo cores diferentes, enfim, cuidando dos detalhes da aparência da informação. E o tempo é ainda maior, pois todas essas funcionalidades não são fáceis de serem usadas. Às vezes, chega a ser contraproducente para o autor criar uma carta bonita demais, ou uma planilha maravilhosa, ou um slide incrível, quando a versão mais simples transmitiria a mesma informação, exigindo metade do tempo para ser feita. É claro que a boa aparência é fundamental. Mas é preciso uma busca do equilíbrio entre a estética e a utilidade. O que efetivamente for útil deve ser fácil de ser usado e a pirotecnia

que deslumbra abandonada (lembremo-nos do problema da usabilidade na Web).

- **Problema da falsa inteligência**
  Os editores de texto que tentam adivinhar nossas intenções ou o formato do documento que pretendemos fazer, ou as planilhas que adivinham o conteúdo de um campo assim que digitamos a primeira letra, etc. Ótimo se realmente adivinhassem, ou se fosse fácil para o usuário comum desligar essa "função inteligente". O que vemos, no entanto, é que mais atrapalham que auxiliam, sendo fonte constante de erros. Na verdade, ainda não se sabe fazer programas com capacidade cognitiva, bom senso e outros atributos que efetivamente os categorizassem de inteligentes.

- **Problema da máquina autoritária**
  Esse problema, certamente relacionado com o anterior, é o do balanço entre poder e controle. Soluções autoritárias são usadas em larga escala porque aumentam a produtividade do programador, mesmo que diminuam a produtividade do usuário! Certamente, o fim da era do "usuário burro" ou da eterna desculpa para programas que não funcionam "a culpa é do usuário que não entende nada". Quanto antes essas muletas de programação forem abandonadas, permitindo controle do usuário, melhor para os humanos, que controlarão as máquinas e não o contrário.

- **Problema do excesso de complexidade**
  Não precisamos de grandes explicações quanto à natureza deste problema. Ele aparece todos os dias quando tentamos ligar o computador. Por que demora tanto e são necessários tantos cliques? E a terrível mensagem SEU SISTEMA NÃO FOI DESLIGADO CORRETAMENTE PORTANTO…., vamos ter que esperar mais uns bons minutos, responder mais uma série de mensagens que não fazem sentido para finalmente (se tudo der certo) o computador se recuperar e finalmente poder ser usado. É indesculpável, na beirada do século XXI, projetar sistemas para seres humanos que sejam tão complexos e trabalhosos de usar, mesmo nas tarefas mais simples - ligar e desligar.

A maioria dos profissionais da área de tecnologia da computação afirma que essas dificuldades são "inevitáveis"; em particular essa é uma visão bastante comum entre os estudantes de Ciência da Computação. Norman (1998) apresenta um contra-argumento bastante interessante para a dificuldade intrínseca do uso da tecnologia de computadores fazendo um paralelo com a Teoria da Relatividade (TR). É aceitável que o entendimento da Teoria da Relatividade seja difícil e envolva estudar matemática e física; mas não precisamos entender da TR para usar objetos físicos. Da mesma maneira não deve ser necessário ser um experto em Computação para usar um computador na tarefa do dia a dia.

Há também o argumento de que o computador é difícil apenas para as gerações mais velhas. Esse argumento é também discutível; as gerações mais novas se acostumam com as dificuldades impostas pela tecnologia sem ter consciência de que haveria uma solução melhor.

Como enfrentar e minimizar esses problemas?

Ambos os autores, Shneiderman e Dertouzos, apesar das diferentes óticas com que analisam o estado atual da tecnologia computacional, apontam como soluções, direções bastante semelhantes.

Dertouzos (1998) advoga a volta da facilidade de uso e como caminho a implementação do que ele denomina de "sistemas de subida uniforme" que deverão ter algumas propriedades básicas: oferecerão resultados palpáveis para o esforço despendido; serão capazes de automatizar tarefas repetitivas; serão “gentis”, na medida em que ações incompletas ou "erros" de usuários não resultem em catástrofes; e serão tão fáceis de compreender como uma receita culinária. De forma análoga à Nardi (1993) ele advoga a necessidade de se facilitar a atividade de programação desenvolvendo uma nova geração de sistemas de software que abandonem a **postura generalista** que direciona a construção de software preocupados essencialmente com estruturar informações, como bancos de dados, planilhas, editores de texto, *browsers* e linguagens de propósito geral. Com isso se tem ferramentas comuns que podem ser igualmente utilizadas em muitas aplicações diferentes, da engenharia à arte. Com o abandono da postura generalista se poderá ter ambientes de programação especializados oferecendo mais informações e operações básicas de sua especialidade, possibilitando às pessoas se concentrarem no significado da informação, e não na sua estrutura. Este posicionamento conduzirá ao desenvolvimento de sistemas de software que possam ser alterados de modo a melhor se adequarem às necessidades particulares de uma pessoa ou empresa e a programação (especializada e diferente do padrão que conhecemos hoje) como o meio de efetuar essa adequação. Na verdade, isso já vem acontecendo em escala reduzida, com milhões de usuários de planilhas de cálculo e alguns sistemas CAD.

Shneiderman (1998) aponta algumas estratégias para prevenir os problemas, dentre elas podemos citar:

- **Design centrado no humano**
  Já amplamente discutido neste livro, significa concentrar a atenção no usuário e em suas tarefas, ou seja, fazer dos usuários o centro da atenção e com isso construir sentimentos de competência, proficiência, clareza e predição.

- **Suporte Organizacional**
  Além do design do software, empresas produtoras também precisam dar suporte ao usuário. Explorar estratégias de design participativo (discutido no Capítulo 3) e conduzir avaliações freqüentes de usabilidade. Grupos de usuários devem ser

mais que observadores passivos. Precisam saber que serão ouvidos ao apontarem falhas e conseqüentes revisões deverão ser efetuadas.

- **Educação**
  Educação é crucial dada a complexidade do mundo atual. Especial atenção deve ser dada à educação continuada, treinamento no trabalho, e formação de professores.

- **Pesquisa avançada**
  Indivíduos, empresas e governo precisam dar suporte a pesquisa no desenvolvimento de novas idéias, que minimizem os problemas e perigos e disseminem as vantagens de sistemas interativos. Teorias sobre o comportamento cognitivo do usuário, diferenças individuais, aquisição de habilidades, percepção visual, mudanças organizacionais, etc., conforme discutido nos Capítulos 2 e 3 serão fundamentais aos designers e implementadores.

Além dessas, ele menciona também a necessidade do nascimento de uma consciência pública que "force" a solução de problemas e o desenvolvimento de legislação específica relativa à privacidade, direito de acesso à informação, crimes computacionais, de forma a estimular o desenvolvimento e prevenir abusos.

HAVERIA UMA SOLUÇÃO "MÁGICA" ?

Há ainda os que esperam por uma "cura mágica" para as dificuldades através da própria tecnologia. Várias soluções têm sido apontadas para o problema da dificuldade das interfaces de software atuais, entre elas o reconhecimento da fala, visualização 3D, agentes inteligentes, redes de computadores e equipamentos portáteis.

O reconhecimento da fala está longe de ser a solução, uma vez que, usar um software que descobre que palavras foram ditas, para o usuário seria o equivalente a teclar um comando; estaríamos falando de interfaces orientadas a comando. Obviamente isso é muito diferente de reconhecimento de linguagem natural pelo software. Para avaliarmos as dificuldades da área de entendimento de linguagem, basta pensarmos nos fenômenos de que somos capazes com a língua natural e que são difíceis de reproduzir, como vimos no capítulo 2. Um exemplo bastante simples é o chamado *cocktail party:* o fenômemo de extrairmos e focarmos a atenção em determinada conversa, quando muitas acontecem em paralelo, ou nosso entendimento derivado de coisas que não foram explicitamente ditas, que a herança cultural dos falantes permite comunicar. Mesmo que o problema da linguagem natural estivesse resolvido, para muitas tarefas que fazemos, a linguagem não é a forma mais

apropriada de descrição; tente descrever em palavras os nós dos marinheiros ou como dar nó em gravata ou ...

Nossos mecanismos perceptuais se adaptaram ao espaço 3D; nos lembramos do lugar no livro onde lemos algo interessante, do lugar na sala onde nos sentamos, etc. Somos levados a pensar que uma solução para o problema da complexidade da interface seria alcançada se tivéssemos a mesma facilidade na interface. O problema, segundo Norman (1998) é que o que muitos propõem como solução, não é realmente representação espacial. Mover figuras 3D enquanto ficamos parados é diferente de nos movermos enquanto o mundo fica parado. Somos levados a perguntar, então, e quanto à Realidade Virtual? Também não seria a solução e o contra-argumento é o fato de que mesmo o mundo real ser 3D, não nos impede de perder os objetos, especialmente quando o guardamos em um lugar especial para não perdê-lo.

E a solução dos "agentes inteligentes"? Sistemas que entenderão o que queremos, mesmo antes de pensarmos no que queremos! Alguns trabalham sem serem notados, oferecendo sugestões que o usuário pode ignorar ou explorar. Outros poderão estar no controle tomando decisões em nosso nome, sem que percebamos. Outros poderão simular comportamento humano, poderão ser confundidos com pessoas reais e levar a problemas sociais. De qualquer forma, os assistentes, guias e *wizards* encontrados nos software comerciais atualmente são tentativas de automatização do *help* e são colocados como paliativos ao problema da complexidade de interação com o software e não como solução.

Os computadores em rede seriam a solução moderna para a proliferação de sistemas diferentes que não "conversam" entre si. Tentam combinar as virtudes dos sistemas de tempo compartilhado com os computadores pessoais, mantendo o software no computador central (administrado por especialistas) que seria executado nas máquinas locais. Computadores em rede, ao mesmo tempo em que tomam o controle do usuário, removem o "pessoal" do computador pessoal. Não parecem conduzir à solução, portanto.

Os portáteis, a exemplo do Newton da Apple e do 3ComPalm organizer, são ferramentas especializadas, algumas vezes chamados PDAs (*Personnal Data Assistants*) e representam os primeiros passos em direção ao que Norman (1998) aponta como solução à complexidade do software: os *Information Appliances* (IAs).

Norman (1998) conta que em 1918 a Sears Roebuck vendia o "motor elétrico doméstico", que era complementado por uma série de anexos utilitários como por exemplo o ventilador, a máquina de costura, o aspirador de pó, etc. Hoje a maioria dos eletro-domésticos (que os americanos chamam de *appliance*) têm um motor embutido que, além de não aparecer no nome do aparelho, o usuário não "vê"; isto é, a ferramenta é específica para uma certa tarefa e a tecnologia de motores é "invisível" ao usuário. Essa analogia pode ser feita com o que ele chama de *information appliance*: um artefato ou ferramenta criado para realizar uma função

específica, um utensílio especializado em informação: conhecimento, fatos, gráficos, imagens, vídeo ou som. Uma característica diferenciada dos IAs é a habilidade de compartilhar informação entre si. São exemplos de IAs: a câmera digital, que permite visualizar imediatamente a imagem, freqüentemente de mais valor do que uma impressora, uma calculadora ou agenda eletrônica. Todas podem compartilhar informações com outros artefatos.

Outros autores, em outros domínios do conhecimento apontam na mesma direção de Norman. Engenhofer (1999, p.3), por exemplo, sugere como solução para a complexidade crescente dos GIS (Geographical Information Systems), os SIAs: *Spatial Information Appliances:*

> *[...] Devices that combine a hand-held computer with a GIS receiver, a cellular phone, and a digital camera will enable users to integrate spatial analysis into their daily lives, opening geographic information systems (GIS) to the mass markets of day-to-day use.*

Segundo esse autor, os SIAs representam a próxima geração de Sistemas de Informação Geográfica e diferirão significativamente dos atuais sistemas de propósito múltiplo, constituindo famílias inteiras de SIAs orientados a aplicações específicas.

O computador pessoal tenta ser todas as coisas para todas as pessoas; hoje ele é mais complexo do que os main-frame que substituiu. No modelo dos IAs, cada artefato é especializado na tarefa, de modo que não se distingue o aprender a usá-lo de aprender a tarefa. Há dois requisitos, portanto, que definem um IA: a ferramenta deve ajustar-se à tarefa e deve haver comunicação e compartilhamento de dados entre eles. O poder desse conceito acontece quando se vê os IAs como um sistema de componentes interconectados. Já não são produtos isolados, então, mas famílias de produtos estruturados de forma a trabalharem juntos sem esforço adicional para o usuário.

Esse conceito parece depender de tornar invisível a infra-estrutura (chip, sistema operacional, etc.). Os IAs dependeriam do estabelecimento de uma padronização universal, aberta, para troca de informação. Três axiomas de design são colocados para os IAs: 1. Simplicidade – a complexidade deve ser ditada pela tarefa e não pela infra-estrutura. 2. Versatilidade – o artefato deve encorajar interação criativa. 3. Prazer – deve ser prazeroso de usar.

A combinação da infra-estrutura da comunicação e da computação caracterizará, certamente, o desenvolvimento dos artefatos tecnológicos do próximo século. Em conseqüência a informação estará mais e mais disponível às pessoas, independentemente de onde se encontrarem; os IAs parecem ajustar-se nesse cenário e vêem na mesma direção da não generalidade discutida anteriormente.

Resumindo, acreditamos que qualquer uma das soluções apontadas envolve primeiramente um repensar das práticas atuais de desenvolvimento de produtos de software, centradas na construção do produto e no projeto de seus componentes. E mais que isso, o total desconhecimento por parte dos projetistas do seu usuário enquanto o principal agente do sucesso do software em desenvolvimento.

POR UMA DISCIPLINA DE DESIGN DE SOFTWARE OU DESIGN DA INTERAÇÃO

Kapor (1996), o projetista do Lotus 1-2-3, expõe o que considera a "vergonha secreta" da indústria de software: a dificuldade de uso e os projetos medíocres de seus produtos.

Fazendo um paralelo com a Arquitetura, Kapor cita Vitruvius, crítico romano que propôs o conceito de que construções bem projetadas são aquelas que exibem estabilidade, comodidade e satisfação e sugere esses mesmos atributos para qualificar um bom produto de software: a estabilidade, na ausência de falhas; a comodidade, no atendimento aos requisitos de funcionalidade e a satisfação, ao tornar prazerosa a utilização do programa. Para Kapor (1996) a precariedade atual dos produtos de software deve-se à absoluta inexistência de preocupações com o seu design e à ênfase dada pelos desenvolvedores aos aspectos internos dos programas, em detrimento daqueles relacionados com a interface com os usuários, muito embora até 75% do código produzido para um programa moderno estar relacionado com tais interfaces. Kapor (1996) conclui propondo que a atividade de design de software seja reconhecida como uma área profissional própria, no mesmo nível da Engenharia de Software.

De modo geral os resultados advindos dos estudos de IHC por um lado sugerem uma clara divisão no trabalho de desenvolvimento de software, onde designers definem o comportamento do produto, suas características externas e os modelos subjacentes principais, e os engenheiros os constróem (Winograd, 1996). Por outro lado, sugerem a possibilidade de design de software vir a deixar de existir como uma atividade autônoma, fundindo-se com a engenharia de software, seja através da formação de superdesigners com múltiplas habilidades, ou mais provavelmente, no trabalho em equipes multidisciplinares interagindo no desenvolvimento do produto a partir de uma visão unificada do mesmo (Laurel, 1994).

Winograd (1997) na mesma linha de Kapor (1996) advoga a favor da criação de uma nova área que ele denomina de Design da Interação. Como já mencionamos no início deste capítulo ele analisa a trajetória da indústria de computação - da máquina para o software para a comunicação para o conteúdo. E como o foco de interesse prático e comercial vem mudando, o mesmo acontece com o papel das pessoas envolvidas no trabalho. Ele afirma que o trabalho será norteado por disciplinas que focalizam

pessoas e comunicação - Psicologia, Comunicações, Design Gráfico, Lingüística - da mesma forma que em disciplinas que dão suporte às tecnologias de comunicação e computação.

Winograd (1997) questiona: Se a computação ampliou-se como um empreendimento social e comercial, o que irá acontecer com a Ciência da Computação como uma disciplina profissional? Deverá ser estendida para incluir design gráfico, lingüística e psicologia? O que poderá significar termos uma disciplina com tal amplitude?

Responde, ser mais realista imaginarmos que a Ciência da Computação não irá ampliar suas fronteiras, e pelo contrário irá se contrair ao podar seus ramos. Muito do sucesso comercial da indústria computacional é dirigido por fatores fora do escopo técnico da ciência da computação, como a conhecemos hoje, mas sempre existirão novas teorias, descobertas e avanços tecnológicos tanto na área de hardware como na de software que compõem o núcleo central da disciplina tradicional.

Como paralelo, podemos pensar na indústria automobilística e no papel da engenharia mecânica e a teoria termodinâmica dentro dela. Claramente, o sucesso do mercado automotivo é determinado por fatores que muito pouco tem a ver com a ciência ou a engenharia - como posicionamento do veículo no mercado, habilidade de associá-lo a um apelo emocional de imagem e estilo, propaganda, etc. Engenharia continua importante e relevante, mas não é o maior fator de sucesso, e não é a força dominante na indústria automobilística.

Espera-se o mesmo tipo de dissociação no mundo computacional. A indústria computacional irá utilizar o trabalho de muitas profissões diferentes dentre as quais a Ciência da Computação que irá continuar focalizando os aspectos da computação que tenham a ver com teorias formais e métodos de engenharia, mas perderá seu aspecto centralizador.

Interação Humano-Computador é por excelência uma área com preocupações interdisciplinares, fato largamente marcado ao longo deste livro, e no meio dessa colisão interdisciplinar Winograd (1997) vislumbra o início de uma nova atividade profissional, que ele denomina de "design da interação". Uma disciplina desenvolvida a partir de diversas outras, mas com um conjunto distinto de preocupações e métodos - elementos de design gráfico, design de informação, e conceitos de IHC - que consistirão na base para o design da interação com sistemas baseados no computador.

Também um paralelo pode ser feito com a Arquitetura, considerando sua disciplina co-irmã a Engenharia Civil. O arquiteto se preocupa com as pessoas e suas interações com e dentro de um espaço a ser criado tentando responder a perguntas do tipo: Será que o projeto provê o tipo de espaço que se encaixa no estilo de vida da família ou no negócio para o qual ele está sendo concebido? Qual é o fluxo de trabalho no escritório, e de que caminhos de comunicação ele depende? As áreas comuns devem

ser ignoradas, ou elas irão levar a um acréscimo de discussão informal? Quais são as diferenças chave entre o design de um banco e uma barbearia, de uma catedral e de um bar?

O engenheiro, por outro lado, está preocupado com aspectos estruturais, métodos de construção, custo e durabilidade. A formação de arquitetos e engenheiros é também diferente. Arquitetos são engajados em um processo que enfatiza a criação e crítica de design. Engenheiros enfatizam a habilidade de aplicar conhecimento formal acumulado na área de modo a ser capaz de calcular possibilidades técnicas e recursos de forma a decidir o que deverá ser construído.

Da mesma forma que uma casa ou um escritório, um software não é somente um mecanismo com o qual o usuário interage; ele é também um gerador de espaço dentro do qual o usuário atua (Winograd, 1997). Design da interação é relacionada com a engenharia de software da mesma forma que a Arquitetura é relacionada com a engenharia civil. Embora não exista uma fronteira clara entre design e engenharia, existe uma diferença fundamental de perspectiva (Winograd, 1996). Em engenharia tradicional, compromissos podem ser quantificados: custos, resistência de materiais, etc. Em disciplinas de design, os compromissos são mais difíceis de serem identificados e quantificados pois residem em valores, necessidades e desejos humanos. O designer tem um pé na tecnologia e outro no domínio de preocupações humanas, e esses dois mundos não são facilmente mensuráveis.

Um exemplo atual é o design de páginas para a Web. O que é preciso conhecer para desenhar uma página? Primeiro, o nome “página” não é muito adequado, pois pressupõe que a WWW é uma coleção de páginas e que, portanto, o conhecimento mais relevante é o do designer gráfico ou do designer de informação. Mas uma página atualmente é muito mais uma interface gráfica que uma página impressa - não é algo só para se ver e sim algo para se interagir. O designer precisa também dominar técnicas de computação e linguagens de programação, como *Perl* ou *Java*. Mas nem o grupo de designers gráficos e o grupo de especialistas em programação são formados no sentido de entender a interação como processo central - não existe uma estrutura de conhecimentos que dê subsídios para fazer um design efetivo de interações entre pessoas e máquinas e entre pessoas usando máquinas. E essa é a proposta da disciplina que Winograd (1997) apresenta.

Concluindo, reafirmamos todos os pontos que analisamos no decorrer deste livro. Um design de interação que tenha sucesso requer que se mude o olhar das máquinas para a vida das pessoas que as usam. Nesta dimensão humana, os fatores relevantes tornam-se difíceis de serem quantificados, e até mesmo identificados. E esta dificuldade aumenta quando se tenta olhar as conseqüências sociais, como assinalamos neste capítulo final.

Design de Software ou Design da Interação ou simplesmente a disciplina de IHC, como parte ou não da Ciência da Computação, nos próximos anos terá um caminho a

delinear, combinando preocupações e benefícios de suas áreas de origem. Como a engenharia, necessita ser prática e rigorosa (como pretende a Engenharia de Usabilidade). Como as disciplinas de design, precisa colocar as necessidades e preocupações humanas como centrais; e como as disciplinas sociais necessita ter uma ampla visão das responsabilidades e possibilidades sociais. O desafio é grande, como também o são os benefícios. E alguns passos já foram dados, como os que apresentamos neste livro.

Referências:

Bush, V. (1945) As We May Think, *Atlantic Monthly* (July). Disponível na Web em http://www.isg.sfu.ca/~duchier/misc/vbush, Consulta em: 13/02/2000

Cybis, W.A. e Michel, G.(1999) A interferência das novas tecnologias e os perigos de sua generalização: uma avaliação ergonômica do voto eletrônico no Brasil, *Atas do II Workshop em Interação Humano-Computador*, Campinas, São Paulo, Brasil

De Bra, P. History of Hypertext and Hypermedia. Disponível na Web em http://www.win.tue.nl/win/cs/is/debra/cursus/history.html Consulta em: 20/02/2000

Dertouzos, M. (1998) O que Será: Como o Novo Mundo da Informação Transformará Nossas Vidas. Companhia das Letras. São Paulo, Brasil

Engelbart, D. C. et al.(1968) A research center for augmenting human intellect. *AFIPS Proc. Fall Joint Computer Conference*, 33, 395-410

Engenhofer, M. J. (1999) Spatial Information Appliances: A Next Generation of Geographic Information Systems. *Proceedings of the I Brazilian Workshop on GeoInformatics*, p. 1-3, Campinas, SP, Brazil

Gromov, Gregory R. History of Internet and WWW: View from Internet Valley. Disponível na Web em http://www.internetvalley.com/intval.html. Consulta em: 13/02/2000

Harasim, L. et al. (1995), *Learning Networks*. MIT Press

Kapor, M. (1996) A Software Design Manifesto. Em T. Winograd (ed.) *Bringing Design to Software*, ACM Press, New York

Laurel, B. (ed.)(1994) *The Art of Human-Computer Interface Design*. Addison-Wesley Pub. Co.

Lévy, P. (1993) *As Tecnologias da Inteligência*. Editora 34 Ltda., SP.

Licklider, J. C. R. (1965) *Libraries of the Future*. MIT Press, Cambridge, MA

Nardi, B. (1993) *A Small Matter of Programming.* The MIT Press, Cambridge, MA

Norman, D. A. (1998) *The Invisible Computer*. The MIT Press, Cambridge, MA.

Pam, A. Where World Wide Web Went Wrong. Disponível na Web em http://www.glasswings.com.au/GlassWings/attendants.html.Consulta em: 13/02/2000

Winograd, T. (1996) *Bringing Design to Software*. ACM Press, New York

Winograd, T. (1997) From Computing Machinery to Interaction Design. Em P. Denning, e R. Metcalfe (eds.) *Beyond Calculation: The Nest Fifty Years of Computing*, Springer-Verlag, 149-162

Wolf, G. The Curse of Xanadu. Disponível na Web em http://wired.lycos.com/wired/3.06/features/xanadu.html .Consulta em: 13/02/2000

Zeltser, Lenny. The World Wide Web: Origins And Beyond. Disponível na Web em http://homepage.seas.upenn.edu/~lzeltser/WWW/#About_WWW. Consulta em: 13/02/2000