INTERFACES USUÁRIO-MÁQUINA

Muito tem se falado a respeito da preocupação com a interface por parte dos profissionais ligados ao projeto de sistemas. E realmente existe esta necessidade de se focar a atenção no planejamento e desenvolvimento da interface desses sistemas.

Na realidade, até determinado tempo atrás, essa preocupação inexistia, uma vez que os usuários de programas de *software* eram os seus próprios desenvolvedores. Sendo assim estes julgavam não haver a necessidade de "gastar" tempo com a estrutura da interface do programa, pois conheciam a fundo o *software* que iriam trabalhar. Mais tarde, com o início da introdução de computadores em variados tipos de organizações, estes programas passaram a ser destinados a um pequeno público de usuários externos, que recebiam treinamento pesado sobre o sistema que utilizariam. Até este ponto tudo ia relativamente bem com as interfaces humano-computador, mas quando os programas de computadores passaram a ser destinados a um público mais amplo e menos treinado, e os sistemas passaram a ser propostos como produtos, destinados ao mercado consumidor, a forma como vinha sendo tratada essa questão passou a não mais funcionar. Isso aconteceu por volta de duas décadas atrás, sendo mais intenso a partir da década de 90, com a "explosão" da informática.

Até então, o desenvolvedor tinha muito mais sucesso construindo programas de aplicação, do que interfaces com o usuário. Este profissional tinha um conhecimento profundo sobre métodos e técnicas e possuía ferramentas que o auxiliavam na construção de um sistema eficaz. Porém já não possuía as mesmas facilidades em relação ao desenvolvimento de uma interface com o usuário, tarefa que exige abordagens, métodos, conhecimentos e treinamento que estes profissionais não recebiam na sua formação e por conseqüência não julgavam necessários. A falta de interesse pela lógica de utilização, fazia com que as interfaces com os usuários fossem sempre deixadas como última coisa no desenvolvimento de um *software*. O usuário, e o trabalho que este efetivamente realizava, era praticamente desconsiderado e as informações necessárias para a estruturação do sistema eram baseadas em dados estritamente técnicos, passados por gerentes e responsáveis pelos sistemas. Fundamentalmente, deixavam de envolver o usuário na concepção e teste de protótipos.

Assim, com a mudança deste pensamento, sobretudo nos anos 90, foram desenvolvidas as primeiras abordagens, métodos, técnicas e ferramentas destinadas a apoiar a construção de interfaces intuitivas, fáceis de usar e produtivas. A Engenharia de Usabilidade, até então desconhecida, saía dos laboratórios das universidades e institutos de pesquisa e começava a ser implementada, como função nas empresas desenvolvedoras de *software*.

O desenvolvimento desta área, assim como o da informática em geral, tem crescido cada vez mais, de maneira irreversível e considerável. Hoje é indissociável a aplicação de estudos baseados em interface na estruturação de um sistema. O usuário passou a ser valorizado, ganhou mais poder e liberdade. Resta aplicar os resultados dos recentes estudos da área no desenvolvimento de *software*s, fornecendo ao usuário este poder, desde que associado à funcionalidade e facilidade de uso do sistema, pois para o usuário, a interface é o sistema.

1. A Interação Humano-Computador

1.1. Introdução

Para dar início a esse conteúdo há a necessidade de se conhecer conceitos que são a base dessa disciplina. Outros conceitos, não menos importantes, serão apresentados posteriormente, de acordo o desenvolvimento do conteúdo.

O primeiro destes conceitos é o conhecimento do que vem a ser **interação**. Interação pode-se definir sucintamente como a atuação de um agente em outro, ou seja, a "ação entre"; segundo a própria estrutura da palavra – *inter* ação. Este agente pode ser ativo em determinada situação e, em outra, deixar-se agir pelo outro, isto é, submeter-se à ação do outro. A interação então é o princípio fundador da linguagem, sendo que o sentido do que se quer transmitir depende da relação entre sujeitos, ou agentes.

Semelhante ao termo "ação", o termo "atividade" significa a qualidade ou estado do que é ativo, isto é, representa a possibilidade e a faculdade de operar. Assim, "atividade" comportaria múltiplas e variadas ações, mas não estaria restrita a elas, pois em uma atividade também está agregado um conjunto de estratégias e escolhas para as ações. Quando se une o termo "inter", surge a "atividade entre" e não a "ação entre". Forma-se então a idéia de que **interatividade** é o processo que favorece a participação ativa de agentes ou sujeitos.

Aplicando este conceito à informática, a interatividade digital pode ser entendida como um diálogo homem-máquina, através de uma zona de contato chamada de **interface** gráfica. A interface seria então, a princípio, o meio (*hardware*, *software*) no qual se dá o processo de interação e interatividade; uma espécie de "conversação" entre o homem e a máquina por meio de um ambiente.

1.2. Definição de interface

Quando o conceito de interface começou a surgir, esta era geralmente entendida como o *hardware* e o *software* com o qual o homem e o computador podiam se comunicar. Hoje em dia, quando se pensa no conceito de interface, imediatamente se visualiza janelas, menus, ícones e barras de rolagem, mas certamente não é só isso. O que aconteceu é que, o aumento do interesse e da necessidade de se melhorar a interface e seu conceito, levou à inclusão dos aspectos **cognitivos** e **emocionais** do usuário durante esta comunicação com a máquina.

Reestruturando então esta idéia, chega-se à descrição de CYBIS (2000) que diz que,

> "a interface com o usuário é formada por apresentações, de informações, de dados, de controles e de comandos. É esta interface também que solicita e recepciona as entradas de dados, de controles e de comandos. Finalmente, ela controla o diálogo entre as apresentações e as entradas. Uma interface tanto define as estratégias para a realização da tarefa, como conduz, orienta, recepciona, alerta, ajuda e responde ao usuário durante as interações".

Interface homem-computador compreende todos os comportamentos do usuário e do computador que são observáveis externamente. Há uma linguagem de entrada, uma de saída para refletir os resultados e um protocolo de interação. Veja a representação deste conceito na figura abaixo.

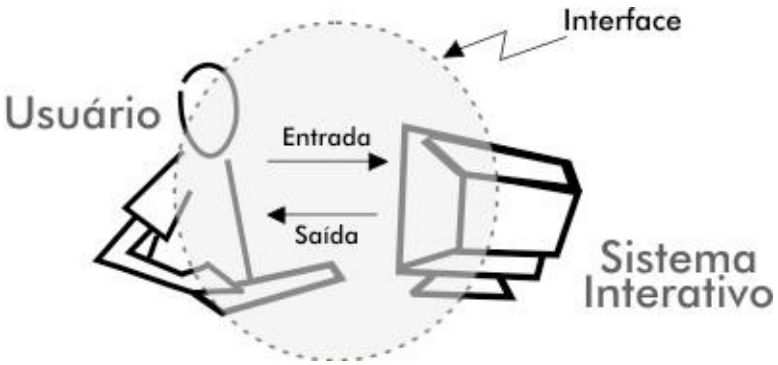


Figura 1 – Funcionamento da interface usuário-máquina

Ou ainda no conceito de LAUREL, citado por ROCHA(2000):

> **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.

1.3. Importância de um projeto otimizado de interface

Acredita-se que a grande dificuldade no desenvolvimento de interfaces otimizadas se deve ao fato delas constituírem fundamentalmente, sistemas abertos, sujeitos as influências do ambiente e as interpretações dos diversos tipos de usuários. Suas entradas e saídas podem significar coisas diferentes para pessoas diferentes, em função de seu conhecimento, do momento, do ambiente que as cercam. Pode-se afirmar então que a experiência da interação humano-computador é individual e única, no sentido de que cada pessoa é única em seu nível de conhecimento e experiência. Por isso é comum que o projetista de determinada interface tenham uma interpretação diferente acerca do sistema quando comparada com a interpretação do usuário.

Uma boa interface torna a interação com o sistema mais fácil de aprender e usar (amigável). Em outras palavras, a interface pode influir na produtividade do usuário, que nem sempre prefere um sistema com mais recursos ou eficiência do ponto de vista computacional.

Uma das justificativas para a atenção atualmente voltada às interfaces é a de que o uso de computadores tem crescido continuamente. Hoje em dia, desde a infância as pessoas já mantém contato com os recursos da informática, prevendo-se assim que, praticamente todo ser humano irá utilizar computadores no futuro de uma ou de outra forma.

Além disso o mercado tem mostrado que, nas vendas entre produtos similares, sobressai o que melhor permite o acesso do usuário à funcionalidade fornecida pelo sistema. É realidade que em alguns casos a funcionalidade e o desempenho não são suficientes

para agradar o usuário, que faz opção por outro sistema com interface mais atrativa. Ou seja, se um produto deseja ser competitivo, necessariamente sua interface deve ser considerada como fator de alta importância nesse processo.
Outro fator a se observar é que uma boa interface interfere economicamente na utilização de um sistema, pois quanto mais fácil de se utilizar um sistema, menor a necessidade de gastos com treinamento ou cursos de aprendizado. A interação com o sistema inclui, muitas vezes, o consumo de tempo desnecessário de operação se realizada de forma inadequada, ou ainda na correção de erros freqüentemente cometidos pelos usuários. Estes custos devem estar envolvidos em uma análise custo/benefício no desenvolvimento de uma interface amigável. Pode-se concluir então que a qualidade da interface tem grande influência no sucesso comercial de um sistema. Os melhores programas podem se tornar ineficazes devido a uma interface imprópria com o usuário.

LUCENA & LIESENBERG (2005) tratam alguns outros fatores que reiteram a justificativa da importância da interface para um sistema:

- o custo de um sistema computacional não se limita a *hardware* e *software*. É preciso treinar usuários. Quanto mais difícil de aprender mais oneroso é o treinamento e quanto mais difícil de usar, menor é o desempenho do usuário através de erros constantes, lentidão de operação do sistema e outros;
- *softwares* que apresentam dificuldades como as acima citadas tendem a ser rejeitados pelos usuários. Comercialmente, o sucesso de vendas de *software* interativos está intimamente relacionado à facilidade de uso e aprendizado do produto, adjetivos que acompanham praticamente toda propaganda de *software* hoje em dia;
- o desenvolvimento de interfaces é um processo caro, difícil, demanda tempo e que ainda há muito a ser empreendido;
- o número de usuários de computadores está se expandindo e com ele a demanda por sistemas interativos. As vendas, a descrição adequada e correta de tarefas, e inclusive a segurança de tais sistemas são influenciados pela interface, que consome 50% dos recursos de desenvolvimento de um sistema.

1.4. A multidisciplinaridade em IHC

O termo **Interação Humano-Computador** (**IHC**) começou a ser adotado como um meio de descrever esse novo campo de estudo. Este vem da necessidade de mostrar que o foco de estudo é mais amplo que somente o *design* de interfaces, abrangendo também todos os aspectos relacionados com a interação entre usuários e computadores.

Portanto, **IHC** é a disciplina que se ocupa do estudo do *design*, avaliação e implementação de sistemas computacionais interativos para uso humano e os principais fenômenos ao redor deles. Trata do *design* de sistemas computacionais que auxiliem as pessoas de forma a que possam executar suas atividades de maneira produtiva e com segurança.

A IHC tem papel no desenvolvimento de todo e qualquer tipo de sistema, podendo ser desde sistemas mais complexos até sistemas mais simples e direcionados para tarefas de menor complexidade, incluindo também jogos e aplicativos ou sistemas *web*.

Quanto à perspectiva disciplinar seria incoerente dissociar a IHC de alguns fatores como: segurança, eficiência, produtividade, aspectos sociais e organizacionais, aspectos comportamentais etc.

A figura abaixo mostra alguns desses conteúdos envolvidos em IHC.

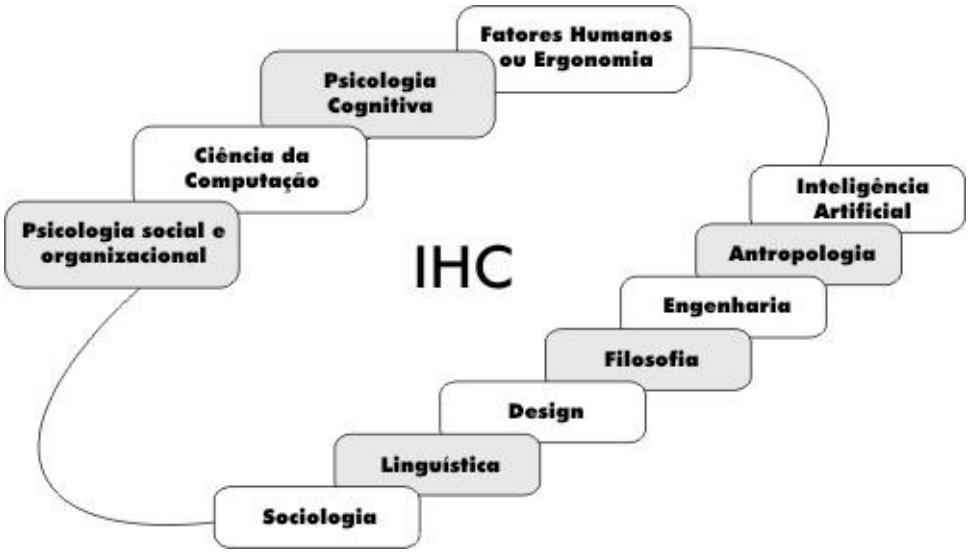


Figura 2 – Disciplinas que contribuem em IHC (PREECE cit. em ROCHA, 2000)

Ciência da Computação: participa provendo conhecimento sobre as possibilidades da tecnologia e oferecendo idéias sobre como explorar todo o este potencial. Também oferece o apoio no desenvolvimento de ferramentas de *softwares* auxiliares ao *design*, implementação e manutenção de sistemas: linguagens de programação, ferramentas de prototipação, sistemas de gerenciamento de interfaces, ambiente de *design* de interfaces de usuário etc.

Psicologia Cognitiva: tem como preocupação principal entender o comportamento humano e os processos mentais subjacentes. Compreende o estudo da percepção, atenção, memória, aprendizagem, solução de problemas etc, bem como a caracterização desses processos em termos de suas capacidades e limitações.

Psicologia Social: estuda a natureza e causas do comportamento humano no contexto social. São preocupações básicas: 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 de diferentes grupos; relacionamento entre estrutura e atividades de diferentes grupos.

Psicologia Organizacional: conhecimento sobre estruturas organizacionais e sociais e sobre como a introdução de computadores influencia práticas de trabalho. 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.

Fatores Humanos ou Ergonomia: tem por objetivo 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. Também tem o

objetivo de maximizar a segurança, eficiência e confiabilidade de da performance do usuário, tornando as tarefas mais fáceis e aumentando os sentimentos de conforto e satisfação. As contribuições dessa área iniciaram com o *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 informações na tela do vídeo.

Lingüística: o uso 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. Utilizado também na internacionalização de interfaces. Internacionalização é a preocupação em isolar os fatores culturais de um produto (textos, ícones, datas etc.) de outros que podem ser considerados genéricos culturalmente. Da mesma maneira é utilizada a lingüística no processo de colocar os aspectos culturais em um produto previamente internacionalizado.

Inteligência Artificial: ramo da ciência da computação cujo objetivo é desenvolver sistemas computacionais que exibam características que associamos com inteligência no comportamento humano. A principal preocupação é com o desenvolvimento de estrutura de representação do conhecimento que são utilizadas pelo ser humano no processo de solução de problemas. Atualmente tem-se dado ênfase no desenvolvimento de agentes de interfaces inteligentes, que auxiliam os usuários na navegação, busca de informação, organização da informação etc. Esses agentes reduzem 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: ênfase 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 antimão.

Engenharia: ciência direcionada à construção e testes empíricos de modelos. A grande influência da engenharia em IHC tem sido via Engenharia de *Software*.

Design: a maior contribuição nessa área tem sido o *design* gráfico, principalmente com o advento da *web*.

1.5. Princípios de *design*

Muitas vezes, pela complexidade de determinados sistemas, a IHC tende-se a se tornar precária. Por esse motivo alguns princípios básicos foram definidos por estudiosos da área, de maneira a garantir uma boa interação homem-computador.

Quatro princípios básicos são observados, segundo Norman (1988) citado em ROCHA (2000):

5.1.1. Visibilidade e *affordances*

Apenas as coisas necessárias têm que estar visíveis, indicando quais as partes podem ser operadas e como o usuário interage com um dispositivo. Visibilidade indica o mapeamento entre ações pretendidas e ações reais, além de indicar também distinções importantes. A visibilidade do efeito da operação mostra se esta foi executada como

pretendida. O que torna muitos dispositivos difíceis de serem operados é justamente a falta de visibilidade. *Designers* devem prover sinais que claramente indiquem ao usuário como proceder diante de uma determinada situação.

Affordances é o termo definido para se referir às propriedades percebidas e propriedades reais de um objeto, que deveriam determinar como ele pode ser usado. Quando em um sistema se tem a predominância da *affordance*, o usuário sabe que ação tomar somente olhando, sem a necessidade de figuras, rótulos ou instruções.

5.1.2. Bom modelo conceitual

Um bom modelo conceitual permite prever o efeito das ações. Sem este modelo bem definido o usuário efetua as operações solicitadas e não sabe que efeito esperar ou, o que fazer caso aconteça algo errado. O modelo conceitual é portanto claro, e até óbvio, e exige um efetivo uso de *affordances*.

5.1.3. Bons mapeamentos

O relacionamento entre duas entidades é denominado de mapeamento. Especificamente em interfaces computacionais, indica o relacionamento entre os controles e seus movimentos e os resultados. Mapeamentos naturais levam ao entendimento imediato, desde que aproveitem analogia física e padrões culturais. Um objeto é fácil de ser usado quando existe um conjunto visível de ações possíveis, e os controles exploram mapeamentos naturais.

5.1.4. *Feedback*

É princípio básico de sistemas computacionais o retorno ao usuário sobre as ações que foram executadas e seus resultados obtidos. Sem um *feedback* o usuário sente-se “perdido”, sem a certeza de que sua solicitação foi executada, se ocorreu algum erro, se há a necessidade da execução de outro procedimento para continuidade do processo. É de grande importância a preocupação com o retorno da ação ao utilizador do sistema, transmitindo confiança, segurança e transparência do sistema.

2. Fatores humanos e interatividade

Como já discutido anteriormente, o estudo de projetos de interfaces não está relacionado apenas com as possibilidades e limitações tecnológicas; é muito mais abrangente que somente um elemento técnico. O estudo do ser humano e seu comportamento também está diretamente relacionado ao desenvolvimento de interfaces. Envolve o conhecimento sobre o humano, sobre a tecnologia e sobre as maneiras como um influencia e é influenciado pelo outro.

Algumas questões precisam ser respondidas e assimiladas para a aplicação nessa concepção.

- Quem é o usuário?
- Quem faz o usuário interagir com o sistema informatizado?
- Como este usuário interpreta as informações produzidas pelo sistema?
- O que o usuário espera do sistema?
- Como o usuário avalia o resultado obtido?

Quando se trata de usuário, ou fatores humanos, de sistemas informatizados, deve-se procurar entendê-los sob diferentes óticas. Em um nível fundamental procura-se entender a percepção visual, a psicologia cognitiva de leitura, memória humana e raciocínio dedutivo e indutivo. Também deve-se procurar entender o usuário e seu comportamento e, por fim, as tarefas que o sistema executa para o usuário e as tarefas que são exigidas do usuário (interação humano-computador).

PRESSMAN (1995) trata a questão do envolvimento do fator humano em sistemas informatizados quando descreve que a interface com o usuário é o mecanismo por meio do qual se estabelece um diálogo entre o programa e o ser humano. Se tiverem sido levados em consideração os fatores humanos, esse diálogo será harmonioso e um ritmo será estabelecido entre o usuário e o programa; caso não aconteça essa valorização, o sistema quase sempre será tratado por “não-amigável”.

2.1. Modelo de processamento de informação humano

Da mesma maneira que se tem definido um sistema de processamento de informações em termos de memória, processadores, parâmetros e interconexões, propõem-se um modelo de processador de informação humano como uma descrição aproximada para ajudar a prever a interação humano-computador, relacionado a comportamentos. É constituído por um conjunto de memórias e processadores e um conjunto de princípios de operação. A esse modelo, Card, citado em ROCHA (2000) dá o nome de MPIH, que é composto por mais três subsistemas:

- o sistema perceptual (SP);
- sistema motor (SM);
- sistema cognitivo (SC).

Veja a representação do modelo na figura abaixo.

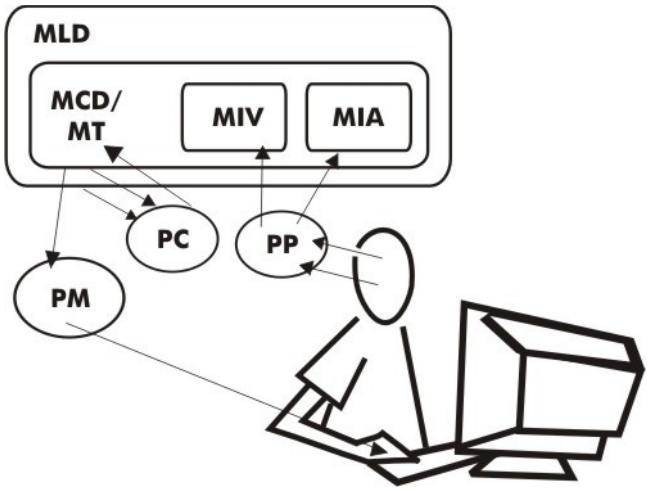


Figura 3 – Modelo do Processador de Informação Humano – MPIH e seus componentes principais (CARD cit. em ROCHA, 2000)

Nesse modelo a informação sensorial é captada pelos órgãos do sentido (aqui pela visão e audição) através do Processador Perceptual (**PP**) e flui para a Memória de Trabalho (**MT**) ou Memória de Curta Duração (**MCD**). A Memória de Trabalho consiste da ativação de partes da Memória de Longa Duração (**MLD**), também chamada 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, fazendo agir certos 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.

2.1.1. Sistema Perceptual

Este sistema transporta sensação do mundo físico, detectadas por sistemas sensoriais do corpo e os transforma em representações internas. Após a apresentação de um estímulo visual, uma representação aparece na MIV. Caso o estímulo seja auditivo é utilizada a MIA. Essas memórias sensoriais guardam informação codificada fisicamente.

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.

2.1.2. Sistema Motor

Logo após o processamento perceptual e cognitivo, o pensamento é traduzido em ação pela ativação de padrões de músculos voluntários que são arranjados em pares antagônicos disparados um após o outro em seqüência. São exemplos deste modelo, quando refere-se a sistema informacional, os sistemas braço-mão-dedo e cabeça-olho, pois são capazes de responder a impulso nervoso.

2.1.3. Sistema Cognitivo

Tem a função de conectar entradas do Sistema Perceptual para saídas corretas do Sistema Motor, isso em tarefas mais simples. Mas a maioria das tarefas realizadas pelo ser humano envolve de forma complexa aprendizado, recuperação de fatos e resolução de problemas. A Memória de Trabalho, ou Memória de Curta-Duração, e a Memória de Longa Duração, ambas associadas ao Sistema Cognitivo, formam as bases para o entendimento de estratégias e teorias em IHC.

De maneira geral, pode-se dizer que a Memória de Curta Duração é usada para armazenas informação sob consideração no momento de determinada atividade e a Memória de Longa Duração armazena informação a ser acessada a longo prazo.

2.2. Mecanismos da percepção humana

A percepção da informação apresentada na interface é feita através dos sinais que a constituem. Em sistemas informatizados torna-se clara a necessidade de entendimento de outras modalidades perceptuais e não somente “ver” o elemento.

Algumas teorias tentam explicar a percepção humana, dentre elas: a teoria construtivista, que acredita que a visão do mundo é construída de forma ativa por informação obtida do ambiente somada ao conhecimento previamente armazenado; a teoria ecologista, que defende que percepção é um processo direto que envolve a detecção de informação do ambiente e não requer quaisquer processos de construção ou elaboração, ou seja, os objetos carregam certas características que dirigem nossa percepção sobre eles. Como exemplo da linha de pensamento ecologista, pode-se citar o conceito de *affordance*, apresentado anteriormente.

A proximidade, similaridade, fecho, continuidade, simetria, são exemplos de fatores que explicam a forma como características no sinal apresentado nos levam a perceber ou não determinada informação.

A quantidade de informação disponível para representar determinada situação pode ser reduzida e mesmo assim o processo se torne aparente e possível de ser interpretado, mesmo que de maneira mais difícil, conforme ilustrado nas figuras 4.1 e 4.2.

Figura 4.1 – Imagem oculta de um cachorro

Figura 4.2 – Imagem oculta da face de Jesus Cristo

Outro artifício consiste em colocar organizações competitivas nas imagens, como é mostrado nas figuras abaixo.

Figura 5.1 – Ilustração de um saxofonista e do rosto de uma mulher em uma mesma imagem

Figura 5.2 – Ilustração de um esquimó e do rosto de um índio em uma mesma imagem

Figura 5.3 – Ilustração do rosto de um velho senhor e um casal de namorados em uma mesma imagem

Figura 5.4 – Ilustração de uma jovem, de uma velha senhora e de um senhor de bigode em uma mesma imagem

Deve-se tomar cuidado também para que as informações apresentadas não se tornem conflitantes. (Figuras 6.1 e 6.2)

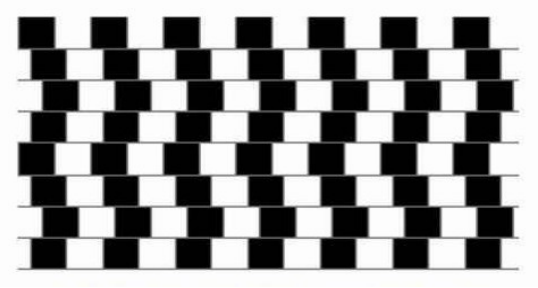


Figura 6.1 – Linhas paralelas que nos passam a impressão de inclinadas

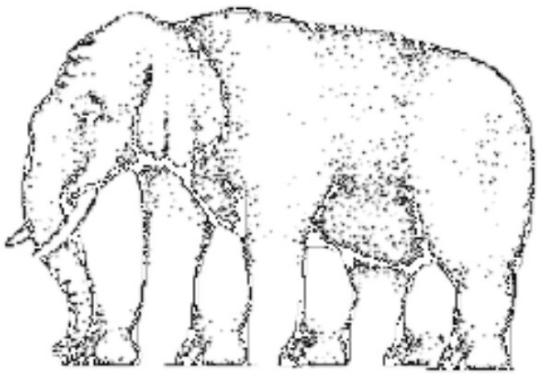


Figura 6.2 – Imagem que confunde a quantidade de patas do elefante

2.3. Modelos mentais

Modelos mentais são representações criadas pelos seres humanos da realidade percebida por esses. Esses modelos, que condicionam totalmente o comportamento do indivíduo, constituem a sua visão da realidade, que é modificada e simplificada pelo que é funcionalmente significativo para ele. Isso significa que a pessoa valoriza os elementos que julgar pertinente e elimina os conseqüentemente avaliados como secundários, estando a representação resultante ligada aos conhecimentos já adquiridos e à compreensão que o indivíduo tem de um problema.

Portanto, deduz-se que os modelos mentais, principalmente aqueles relativos a um sistema interativo, variam de pessoa para pessoa, em função de suas experiências passadas e conhecimentos, evoluindo então em função da aprendizagem e utilização daquele sistema.

Também é correto afirmar que modelos mentais são incompletos. A habilidade das pessoas em executar e/ou criar seus modelos mentais é limitada pelos mecanismos perceptual e cognitivo. Esses modelos também caracterizam-se como instáveis, pelas próprias restrições e interferências da memória, como o esquecimento de determinadas funções ou detalhes do sistema em uso, ou mesmo confusão com operações semelhantes.
Distingui-se então, numa determinada situação de trabalho informatizada, as seguintes conseqüências:

- os modelos mentais relativos a uma interface correspondem a um conjunto de conhecimentos semânticos (conceitos) e procedurais (procedimentos) que é particular a cada usuário.
- os modelos mentais desenvolvidos por projetistas e por usuários se diferenciam grandemente;
- os modelos mentais desenvolvidos por indivíduos, que exercem diferentes funções com o sistema, gestão ou de operação, por exemplo, se diferenciam grandemente. Neste caso são evidentes as diferenças nas representações mentais de quem opera um sistema assídua e freqüentemente, de quem o faz de maneira esporádica ou intermitente;
- os modelos mentais desenvolvidos por usuários novatos e por experientes se diferenciam grandemente;

Salienta-se a importância do conhecimento do conceito de modelos mentais em sistemas informatizados interativos na elaboração da interface desse. Tal interface deve ser flexível o suficiente para adequar-se aos diferentes tipos de usuários, ao mesmo tempo em que possa adaptar-se à evolução das características de um usuário específico durante seu processo de aprendizagem com o sistema.

Segundo as teorias cognitivas existem dois tipos principais de modelos mentais: o estrutural e o funcional. O modelo mental estrutural atua como um substituto da coisa real; é assumido que o usuário internalizou a estrutura de como o sistema funciona. Já no modelo mental funcional o usuário internaliza conhecimento procedimental sobre como usar o sistema; se desenvolve a partir de um conhecimento anterior de um domínio similar. Ambos os modelos contribuem igualmente para o entendimento do sistema. Daí a necessidade dos textos de ajuda explorarem estas duas perspectivas de um software interativo; como funcionam e como se operam suas funções.

Para o projeto de interfaces humano-computador, além da variabilidade, nos indivíduos e no tempo, é importante saber o que favorece ou limita a elaboração, armazenagem e a recuperação destas representações em estruturas de memória e por meio da percepção da realidade.

A operacionalização do sistema será mais simples se tiver um bom modelo conceitual. É tarefa do *designer* construir um modelo conceitual para o artefato, adequado ao uso. São três os modelos associados ao artefato: o modelo do *designer*, o modelo do usuário e a imagem do sistema. Os modelos do *designer* e do usuário são 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, juntamente com o *help* de manuais de instrução formam a imagem do sistema.

2.4. Componentes da interação humano-computador

Baseado nos estudos de Jakob NIELSEN, os componentes da interface usuário-máquina foram agrupados de maneira a organizar a estrutura dessas interfaces e os conhecimentos para selecionar, configurar e avaliar tais componentes.

O modelo propõe classes de elementos organizados a partir de diálogos, objetos de interação, sistemas de significados e primitivas.

2.4.1. Diálogos

São seqüências de interações entre o homem e o sistema. Analisados segundo perspectivas de função, forma e estrutura. As funções dos diálogos definem as classes de tarefas, e representam o nível pragmático das interações homem-sistema. Elas estão associadas às maneiras de apoiar os objetivos práticos dos usuários nas interações com o sistema. O modelo de características de interfaces humano-computador propõe alguns tipos de tarefas genéricas definidas nas relações com diversos tipos de programas aplicativos. O componente elementar de classes de tarefa é uma “ação”.

2.4.1.1. A ação

Corresponde à uma interação elementar. Esta compreende a menor entrada significativa do usuário acompanhada de uma resposta também significativa do sistema.

Em uma ação o sistema deve sempre aguardar pelo término da entrada e fornecer feedback imediato e significativo para ela. Se necessário, o sistema deve considerar como equivalentes as letras maiúsculas e minúsculas, além de preencher automaticamente zeros decimais e vírgulas. O sistema deve também avisar o usuário sobre os erros nas entradas através de um sinal sonoro.

Nas ações de entrada que envolvem tratamento demorado pelo sistema, deve ser dada atenção redobrada às questões de feedback, informando ao usuário sobre a indisponibilidade do sistema; o tempo esperado do tratamento; o estado atual do sistema; o resultado alcançado. Além disso, uma opção para a interrupção do tratamento deve estar disponível ao usuário.

2.4.1.2. Tarefa
A tarefa é uma seqüência de ações ou interações elementares.
Em algumas tarefas o objetivo do usuário é o de elaborar um diagnóstico visando a recuperação de incidentes relacionados ao sistema de produção. O sistema de controle informatizado deve apoiar o usuário, apresentando-lhe os dados críticos de modo diferenciado e lhe propondo ajuda on-line, com orientações exibidas em linguagem simples e objetiva.

Outro fator importante relacionado à tarefa é a qualidade das mensagens de erro. É aconselhável que as mensagens tenham um nível de detalhe configurável, de modo a que sejam adaptadas ao tipo de usuário, e que tenham um conteúdo dinâmico, variando no caso da reincidência de erros.

2.4.2. Objetos de interação
É um objeto de software cujo processamento gera uma imagem que é apresentada ao usuário e com a qual ele pode interagir. Esses objetos ocupam as telas das interfaces e podem se basear em metáforas de objetos do mundo não informatizado, representando botões, janelas, menus, interruptores etc.

As partes de um objeto variam de ambiente para ambiente, mas geralmente são definidos como primeiro plano, plano de fundo e bordas. O primeiro plano recebe as palavras e ícones, o plano de fundo recebe os motivos e sombras.

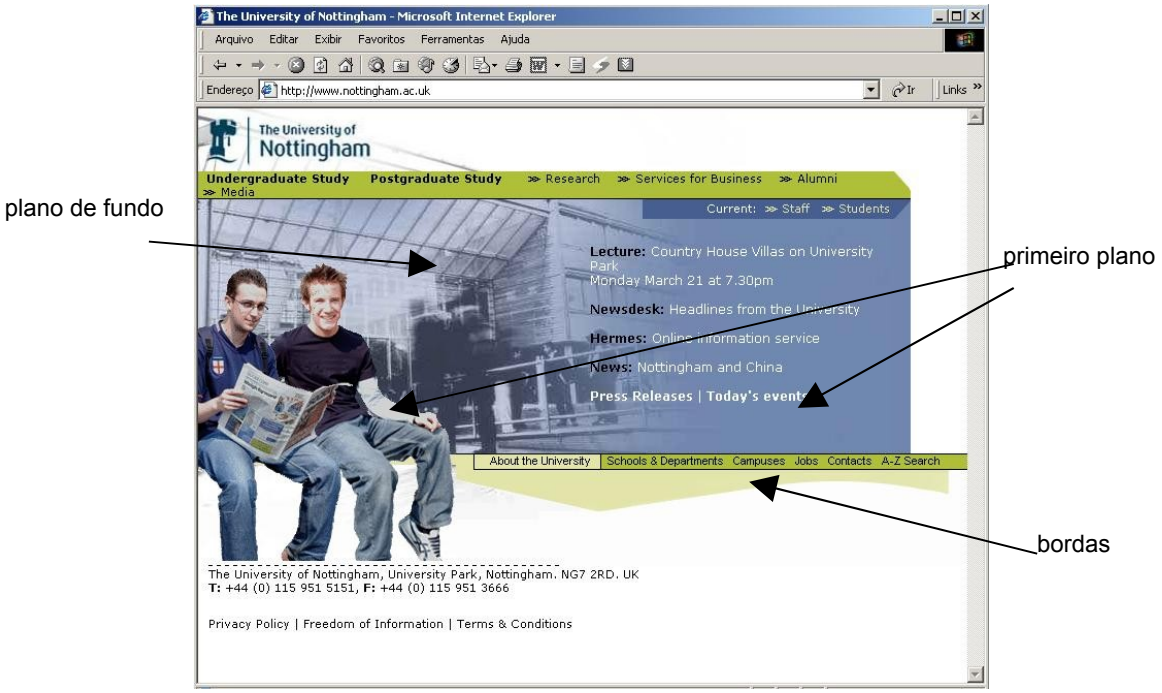


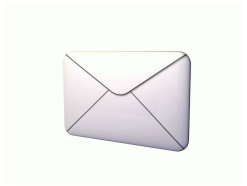
Figura 7 – Identificação de “Primeiro Plano”, “Plano de Fundo” e “Bordas” em uma interface

2.4.3. Sistema de significado
Esses sistemas referem-se as relações simbólicas estabelecidas na transmissão de um conteúdo de informação por meio de uma expressão perceptível e tratável pelo sistema cognitivo humano. Essas relações referem-se a entidades como símbolos e sinais.

Em um sinal a relação entre a forma de conteúdo e forma de expressão pode ser arbitrária ou dependente do conhecimento mútuo das regras de codificação. Os sinais usados tanto em uma linguagem natural como na álgebra ou na matemática tem assim uma capacidade de transmitir um conhecimento mais ou menos objetivo. Em um símbolo, a homogeneidade entre expressão e conteúdo estabelece uma representação motivada ou concreta, onde o caráter espontâneo da interpretação é essencial.

2.4.3.1. Ícones

Pode corresponder a diferentes tipos de representações. É recomendável utilizar sempre ícones prontos, respeitando seus significados.



Figur plos de íco de correio etrônico

Figura 8.2 – Exemplos de ícones de seção de compra

Os ícor ser signi tivos, apropriados, coe sistente os, simples e definidos em pequeno número. Seu tamanho deve ser econômico em relação ao espaço de tela. Dependendo de sua utilização aconselha-se a adoção de bordas bem definidas.

2.4.3.2. Código de formas

Envolvem os sinais geométricos construídos a partir de primitivas gráficas (linha, arco, retângulo etc.). Os círculos, quadrados, triângulos e retângulos são utilizados por exemplo, para codificar classes de eventos em gráficos estatísticos.

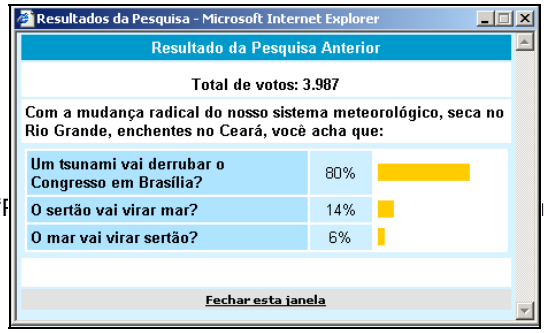


Figura 9 – Identificação de " ma interface

2.4.3.3. Denominações

Constituem códigos de expressão textual cujos termos são retirados da linguagem articulada pela população de usuários em sua tarefa com o sistema. As regras de codificação desses termos são definidas no ambiente de trabalho dos usuários. Assim, a linguagem textual da interface deve ser constituída de termos empregados no contexto de trabalho, portanto significativos e familiares para o usuário.

Figura 10 – Interface do *site* LicitaNET, que traz termos comumente utilizados entre órgãos públicos, citando como exemplo o termo “licitação”

2.4.3.4. Abreviaturas

Devem ser utilizadas somente quando absolutamente necessário. São diminutivos das denominações. As abreviaturas devem ser distintas entre si, claras, curtas e significativas.

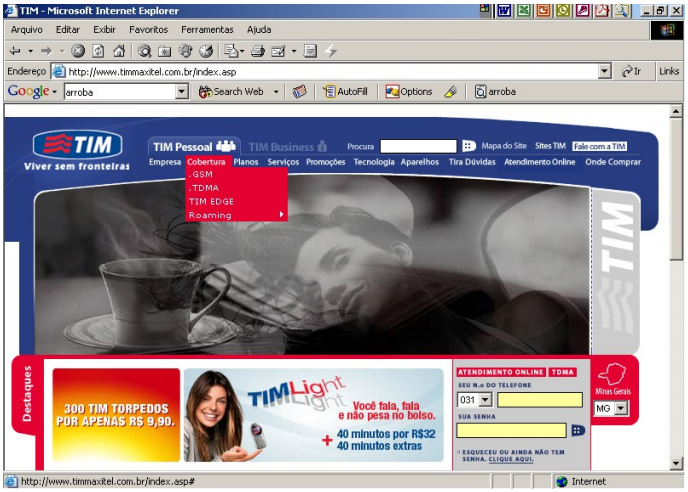


Figura 11 – Interface do *site* da empresa de telefonia móvel TIM, que traz algumas abreviaturas de serviços prestados, como por exemplo o termo GSM, que significa Global System Mobile (sistema móvel global)

2.4.3.5. Códigos de cores

A utilização das cores em uma interface exerce papel importante, sendo suas finalidades estratégicas: a transmissão de informações, chamar a atenção, contrastar e associar

objetos de interação. O uso puramente decorativo é desaconselhado. Para uma eficaz utilização deve-se levar em consideração alguns aspectos:

- a legibilidade final da informação;
- os efeitos das cores sobre a performance cognitiva do usuário e;
- as possibilidades dos dispositivos físicos.

Essas observações têm por finalidade conter a confusão visual resultante do emprego arbitrário e exagerado de cores não pertinentes. A utilização das cores deve seguir um elenco reduzido e equilibrado de opções, sendo utilizadas não mais do que 10 ou 11 cores. Estas não devem estar associadas a mais do que um significado e deve respeitar estereótipos naturais, como:

- o vermelho deve ser utilizado para perigo, alarme, atenção, alerta, calor e comandos de interrupção;
- o amarelo para advertências, teste e lentidão;
- verde para passagem livre, normalidade, vegetação e segurança;
- o laranja para valor limite e radiação;
- o azul para frio, água, céu e calma;
- cinza para inatividade, neutralidade;
- o branco é uma cor neutra.

Como ainda existem dispositivos físicos monocromáticos, a cor não deve ser utilizada como uma única de expressão. Deve existir uma definição alternativa que atenda a esses dispositivos.

2.4.3.6. Códigos de textura

É utilizada como codificação optativa na apresentação de gráficos e mapas. Se utilizadas juntamente com palavras elas devem ser escolhidas de modo a não prejudicar a leitura e/ou interpretação do texto.

2.4.3.7. Intermitência visual

A intermitência pode ser utilizada para o destaque em situações excepcionais, quando deseja-se chamar a atenção do usuário para a ação a ser tomada, em caráter de urgência, por exemplo. Este alerta não deve ser aplicado a mais de um elemento de cada vez. É interessante que o sistema permita ao usuário desativar a intermitência.

2.4.4. As primitivas

Trata-se das formas para a expressão de um objeto de interação que resultam da articulação de substâncias perceptíveis ao sistema cognitivo humano.

2.4.4.1. Cores

Recomenda-se cuidado com o uso indiscriminado da cor. É aconselhável que primeiro se faça o projeto em Preto e Branco e depois o colora com cuidado, usando cores neutras. Use poucas cores e com mesma luminância (brilho) e utilize cores brilhantes com cautela. Existem teorias que defendem que as cores causam sensações às pessoas. O verde, por exemplo, descansa; o vermelho atrai a atenção e pode causar irritação; o azul dá sono e o amarelo desperta. Usar cores de forma consistente e evitar usar cores opostas no espectro em áreas muito próximas.

2.4.4.2. Fontes

A atenção dispensada quanto à utilização de fontes envolve o uso da serifa e o espaçamento entre os caracteres. A serifa é caracterizada por uma terminação saliente dos caracteres. Fontes sem serifa são de percepção leve, mas de difícil leitura. O emprego de fontes com serifa, principalmente em textos impressos, facilita o reconhecimento rápido dos caracteres. Entretanto, em vídeos de baixa resolução, a leitura de fontes com serifa é prejudicada, pois podem confundir a interpretação do usuário. Portanto recomenda-se, para conteúdos a serem lidos em monitores de computador, a não utilização de serifa.

Figura 12.1 – Fontes com serifa

Figura 12.2 – Fontes sem serifa

Quanto ao tamanho da fonte, não utilizar menor que 10 pontos, limitando o uso de fontes diferentes para textos (até dois tipos). Deve-se evitar também fontes muito grandes, que pareçam "gritar" com o usuário. Evitar textos só com maiúsculas, e não exagerar com o sublinhado, negrito e itálico.

2.4.4.3. Bordas

Grande parte dos objetos de interação são delimitados por bordas. As bordas desempenham papel importante na leveza desses objetos. Essa característica pode ser assegurada através da natureza simples de seus traços e da distância segura entre as bordas e textos em geral.

2.4.4.4. Arranjo ou *layout*

É a forma de disposição dos itens de informação em uma janela, caixa de diálogo ou de mensagem. Alguns pontos devem ser observados quanto ao arranjo adequado:

- Procurar definir um *grid* para o *layout* das telas. Definir alinhamentos para os elementos conforme as linhas e colunas do *grid*;
- Definir focos de atenção (zonas de trabalho) agrupando os elementos interrelacionados. Colocar em evidência o que for mais importante no grupo (mais à esquerda, colorido etc.);
- Distribuir as informações da esquerda para a direita, em função da importância, destaque ou cronologia;
- Dar equilíbrio às telas distribuindo os elementos de forma balanceada. Evitar áreas vazias ou altamente carregada de componentes;
- Manter a consistência entre os *layouts* das telas (padronização).

2.4.4.5. Fundos ou *background*

Deve-se definir os fundos de telas, janelas, caixas de diálogo ou de mensagens com cores neutras, que garantam um contraste adequado com os textos e rótulos em primeiro plano. É recomendável não carregar o fundo da tela com elementos gráficos (para áreas

de trabalho e leitura). As cores e padrões para os fundos podem ser usados para diferenciar tipos de telas/áreas.

2.4.4.6. Formas sonoras

Apresenta os atributos de expressão "timbre" e "freqüência", utilizados para destaque ou diferenciação do sinal sonoro. O timbre está ligado a natureza da entidade física que gera um som. A mesma nota musical em um piano ou em clarinete soam diferente devido a seus timbres particulares. A freqüência, também denominada de registro de um som, pode ser alta ou baixa relativamente as oitavas. Aconselha-se a utilização de tons da mesma oitava para evitar problemas de construção de sinais sonoros.

3. A comunicação humano-computador e o design de interfaces

O processo de estruturação e *design* de interfaces, como visto nos capítulos anteriores, tem sido centrado no usuário, incorporando questões relacionadas diretamente a modelos cognitivos do processamento humano. Já os modelos dos processos de *design* em IHC envolvem uma discussão crítica dos ciclos de vida clássicos para o desenvolvimento de *softwares*, originais da área de Engenharia de *Softwares*, até modelos mais específicos do ciclo de *design*.

3.1. Engenharia cognitiva

Como visto anteriormente, a mente humana procura maneiras de dar sentido a tudo aquilo que vemos, sendo "preenchendo" imagens incompletas ou isolando elementos de uma representação complexa, tornando assim mais simples a interpretação. Portanto, como cita ROCHA & BARANAUSKAS (2000),

> "...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 "Engenharia Cognitiva" é uma espécie de "Ciência Cognitiva Aplicada", que tenta aplicar o que é conhecido da ciência ao *design* e construção de máquinas. Em resumo, como objetivos da Engenharia Cognitiva, temos:

- Entender os princípios fundamentais da ação humana que são relevantes à engenharia do *design*, indo além dos aspectos ergonômicos;
- Criar sistemas agradáveis de usar, indo além dos aspectos de facilidade de uso.

A Engenharia Cognitiva considera dois lados na interface: o próprio sistema e o lado do usuário. Assim, a realização de tarefas complexas é considerada uma atividade de resolução de problemas cujo processo é facilidade quando a pessoa possui um bom modelo conceitual do sistema físico. Ou seja, a pessoa interpreta a realidade que conhece e transfere o conhecimento para o sistema informatizado.

Portanto, na Engenharia Cognitiva, o *design* de interface relaciona três tipos de conhecimento:

- De *design*, programação e tecnologia;
- De pessoas, princípios do funcionamento mental;
- Comunicação e interação e conhecimento da tarefa.

3.2. Manipulação direta

Neste modelo a operação do sistema ocorre diretamente, com o uso de ações manuais ao invés de instruções fornecidas via teclado. Tais sistemas mudaram o paradigma de interação humano-computador, do "diálogo" baseado em linguagem de comando para a "manipulação" baseada na linguagem visual. 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.

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. Portanto a manipulação direta resulta em menor comprometimento de recursos cognitivos. 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.

Para representar a idéia 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.

3.3. Modelo de *design* de *software*

Design de *software*, ou "projeto de software", tenta relacionar a forma e função de um sistema de *software* à estrutura do processo que produz esse sistema.

O processo de *design* na Engenharia de *Software* parte de três princípios:

- O resultado do *design* é um produto;
- O produto é derivado de especificações fornecidas pelo cliente;
- Uma vez que o cliente e o *designer* concordam com as especificações, há pouca necessidade de contato entre eles até a entrega do produto.

3.3.1. Modelo Cascata

Caracteriza a visão tradicional da Engenharia de *Software* para o desenvolvimento de *software*, como um conjunto de processos e representações produzidas de maneira linear.

Figura 13 – Modelo Cascata de *design* de *software*

O principal problema com o modelo cascata é que se torna 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.

3.3.2. Modelo Espiral

Em resposta aos problemas do modelo cascata, propõe-se o modelo espiral. Esse modelo mostra que várias interações são necessárias e introduz a idéia de prototipagem para maior entendimento dos requisitos.

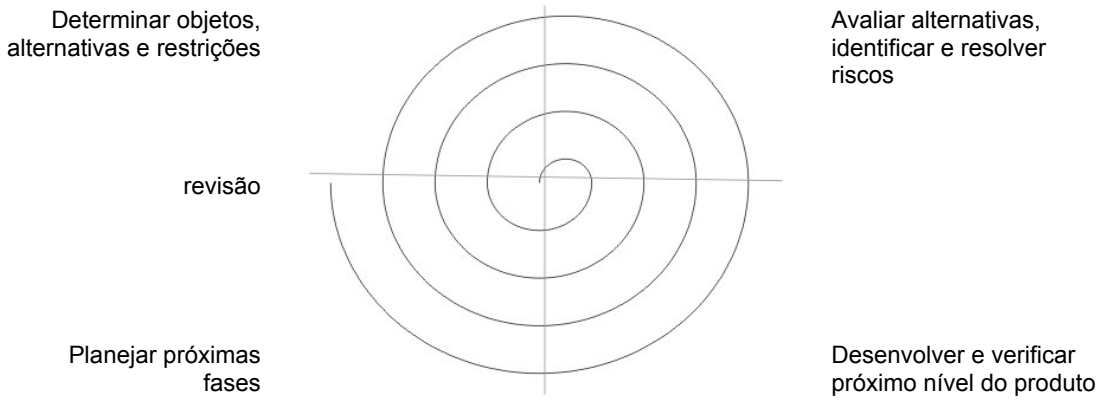


Figura 14 – Modelo Espiral de *design* de *software*

3.3.3. Modelo de Eason

Neste modelo, o desenvolvimento do *design* de *software* é representado como um processo de natureza cíclica centrado em pessoas, trabalho e tecnologia.

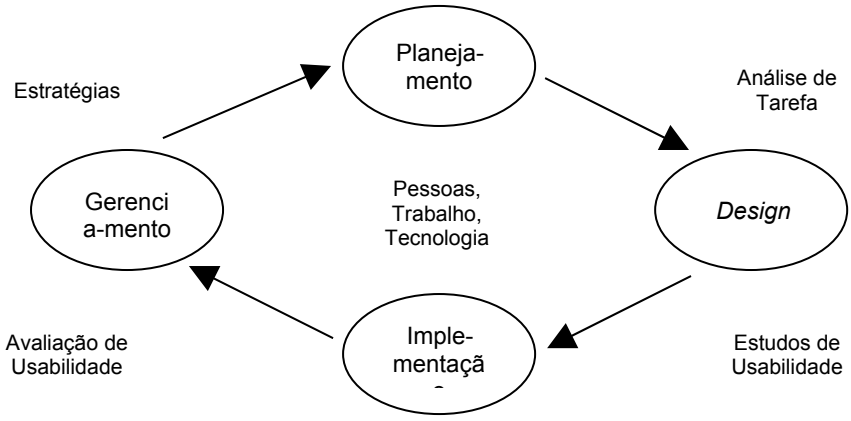


Figura 15 – Modelo de Eason de *design* de *software*

3.3.4. Modelo Estrela

O Modelo Estrela 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.

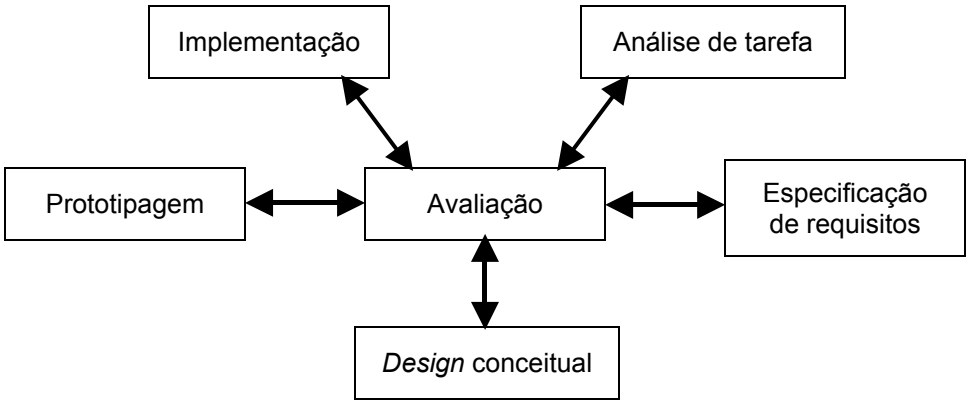


Figura 16 – Modelo Estrela de *design* de *software*

4. Estilos de interação

Estilos de interação são coleções de objetos de interface e técnicas associadas que são utilizadas para desenhar os componentes de interação de uma interface entre homem e computador. É a forma do usuário comunicar com o sistema.

A maioria dos estilos de interação é utilizada em interfaces de manipulação direta.

Pode-se classificar principalmente em:

- Linhas de comandos;
- Janelas (*windows*);
- Cardápios (*menus*);
- Formulários (*forms*);
- Interfaces pictóricas;
- Outros estilos.

4.1. Linhas de comandos

É um estilo de interação que não envolve o conceito de manipulação direta. Neste tipo de interação existe a necessidade do usuário conhecer os comandos do sistema para poder executa-lo, ao invés de apenas manipulá-los (com o arrastar e clique do *mouse* por exemplo), sem o comprometimento da carga cognitiva.

Um exemplo de interação baseada em linha de comando é o sistema operacional MS-DOS®, muito utilizado antes da popularização do Windows®. No MS-DOS era exigido do usuário que conhecesse os comandos e suas sintaxes para que as tarefas pudessem ser executadas. Por exemplo, para criar um pasta o usuário deveria conhecer o comando MD e todas as suas possibilidades, assim como o comando CD para acessar as pastas criadas.

Tais comandos são digitados no *prompt* do sistema, como mostra a figura abaixo.

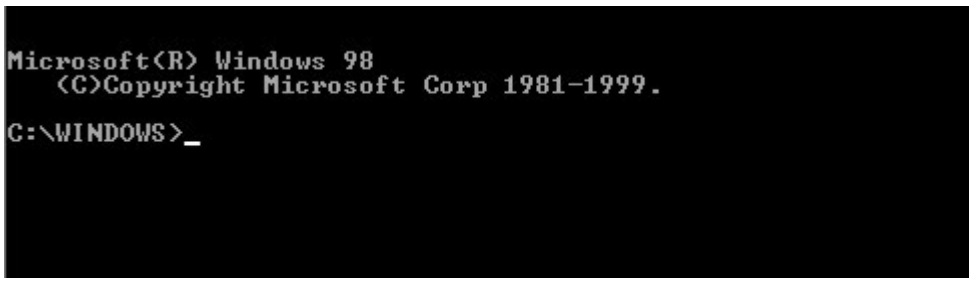


Figura 17 – *Prompt* do MS-DOS

4.2. Janelas (*windows*)

Uma janela é um objeto de tela que fornece uma arena para apresentação e interação com outros objetos de interação. Toda a interação entre o usuário e o sistema ocorre através da janela. Pelas janelas o usuário pode organizar trabalhos em tarefas, trabalhando com várias tarefas ao mesmo tempo.

As janelas podem ser apresentar em Janelas primárias, Janelas secundárias e algumas vezes estas podem ser representadas em Janelas partidas.

4.2.1. Janela primária

A janela primária é a área principal de trabalho de um determinado *software*. É a partir dela que todas as operações do sistema serão executadas.

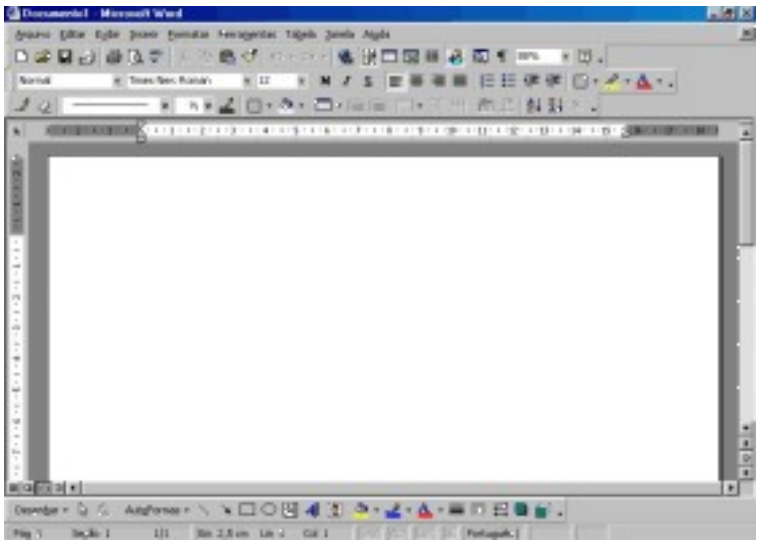


Figura 18 – Exemplo de janela primária – Tela de trabalho do MS-Office WORD®

4.2.2. Janela secundária

E executada a partir da janela primária. É reservada a este tipo de janela a execução de funções do sistema que não necessitem de uma área visual muito grande ou simplesmente não sejam importantes o suficiente para que serem executadas em uma janela primária.

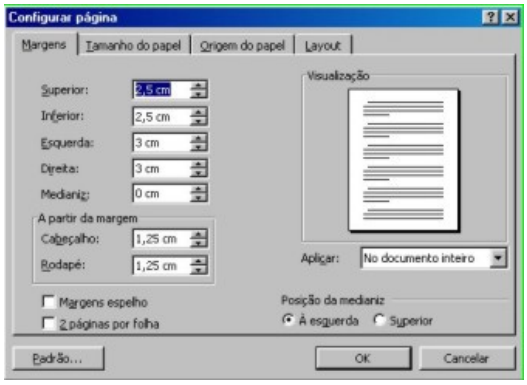


Figura 19 – Exemplo de janela secundária – Opções de configuração de página do MS-Office WORD®

4.2.3. Janela partida

Utilizada principalmente quando se deseja otimizar o espaço da interface de um sistema. A janela, geralmente a primária, é subdividida em duas ou mais áreas, sendo cada uma dessas partes responsáveis pela execução de determinada função do sistema.

Também é utilizada para dar a idéia de ordenação de informações.

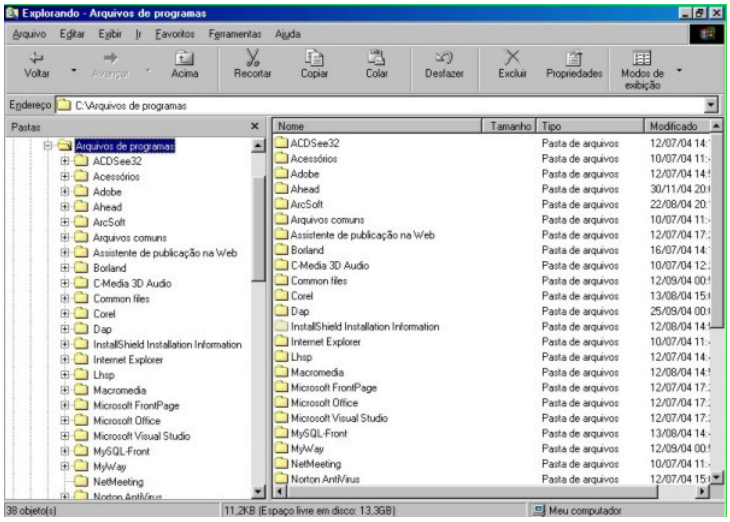


Figura 20 – Exemplo de janela partida – Tela de trabalho do gerenciador de arquivos Windows Explorer®

4.2.4. Projetando janelas

Dentre as normas de otimização de desenho de janelas, existem algumas observações que devem consideradas no momento na definição e estruturação dessas.

- Evite excesso de janelas em cada aplicativo. Administre as janelas, fazendo-se uso das janelas primária, secundárias e partidas, de acordo com o nível de importância e/ou complexidade das funções a serem utilizadas;
- Permita que as janelas sejam reposicionadas e redimensionadas. Assim, o usuário sente-se a vontade para trabalhar com uma quantidade específica de aplicativos, possuindo uma percepção visual geral do que está sendo executado;
- Mantenha a consistência da aparência das janelas. A padronização das janelas de determinado sistema é interessante para a personalização desse. Assim, torna-se possível a identificação de determinado *software* pela visualização de uma ou poucas janelas;
- Use diferentes janelas para diferentes tarefas independentes. Se por um lado sugere-se a utilização não excessiva de janelas, por outro é aconselhável a separação de tarefas diferentes do sistema em janelas diferentes. A coerência é a principal ferramenta na definição das janelas de trabalho. Tarefas semelhantes, ou que afetem diretamente o mesmo módulo de determinado sistema, podem estar contidas em uma mesma janela, desde que essa se torne de fácil entendimento e visualização pelo usuário.

4.3. Cardápios (*menus*)

Os cardápios, ou *menus*, são estilos de interação representados por listas de determinados itens, de onde uma ou mais seleções são feitas pelo usuário.

Os cardápios possuem algumas vantagens quanto à sua utilização, dentre as quais pode-se citar:

- a redução na necessidade de memorização de comandos e/ou informações;
- eliminação da digitação de valores, levando à redução de erros dos usuários;
- redução na necessidade de treinamento, uma vez que as informações disponíveis do sistema são apresentadas diretamente aos usuários.

Vários são os tipos de *menus* existentes, dentre eles pode-se citar:

- Menus *Push-button*

;

- Menus *Radio-button;*
- Menus *Check-button;*
- *Pop-up* menus*;*
- *Pull-down*;
- *Pallete menus*.

4.3.1. Menus *Push-button* – Botões de apertar

Apresentam-se como botões separados, sempre visíveis, a serem clicados pelos usuários. É utilizado, na sua maior parte, quando a interface do sistema possui poucas opções de escolha, justamente pelo fato desse tipo de menu ocupar muito espaço

.

Para uma maior efetivação da utilização desse cardápio, os rótulos (textos) dos botões devem estar bem claros e precisos, enquanto um botão *default*, com aparência diferente, também deve existir.

Uma de suas características, presentes na maioria das interfaces que adotam este estilo de interação, é o realce no botão, assim que escolhido (clicado) pelo usuário.

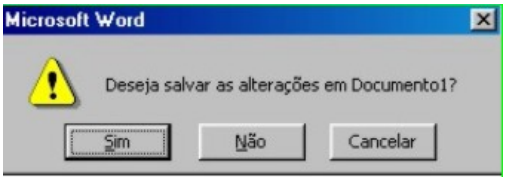


Figura 21 – Exemplo de menu *push-button*

4.3.2. Menus *Radio-button* – Botões de opções

Oferecem aos usuários escolhas que são mutuamente exclusivas, ou seja, apenas uma, dentre as opções disponibilizadas, poderá ser selecionada. Este tipo de cardápio é aconselhável ser utilizado quando a quantidade de opções é pequena.

É importante que sistema disponibilize marcações para indicar a escolha corrente do usuário, além de possibilitar que este mude a sua opção de escolha.

Geralmente os locais de marcação são apresentados no formato circular.

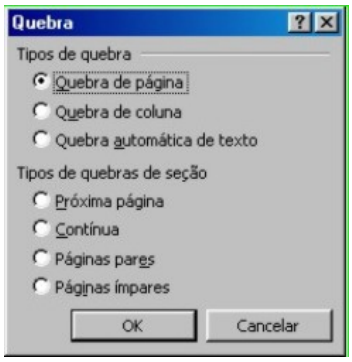


Figura 22 – Exemplo de menu *radio-button*

4.3.3. Menus *Check-button* – Botões de checar

Oferecem aos usuários escolhas que não são mutuamente exclusivas, ou seja, dentre as opções disponibilizadas, o usuário poderá selecionar mais de uma destas. Para este tipo de cardápio também é aconselhável a utilização quando a quantidade de opções é pequena.

É importante que sistema disponibilize marcações para indicar a escolha corrente do usuário, além de possibilitar que este habilite ou desabilite a sua opção de escolha.

Geralmente os locais de marcação são apresentados no formato retangular.

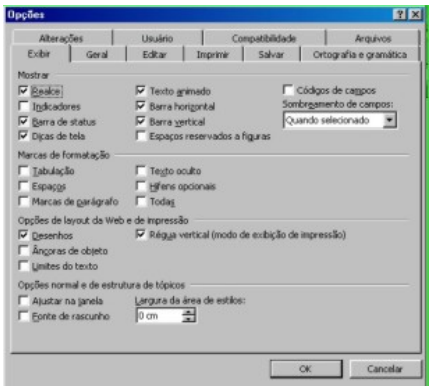


Figura 23 – Exemplo de menu *check-button*

4.3.4. Menus *Pop-up* – Cardápios instantâneos

Aparecem em diferentes lugares na tela, determinado pela posição atual do cursor no momento da solicitação da visualização das opções. Geralmente não há indicação da existência do menu *pop-up* no sistema, sendo muitas vezes utilizados apenas por usuários com um pouco mais de conhecimento sobre o *software*.

Este cardápio não utiliza espaço permanente da tela, economizando assim espaço. A economia no movimento do mouse também é uma de suas características, uma vez que, um simples clique com o botão direito do *mouse* já ativa o menu (como comumente acontece).

Figura 24 – Exemplo de menu *pop-up*

4.3.5. Menus *Pull-down* – Cardápios permanentes

Caracterizados por apresentarem-se sempre visíveis no sistema. Esses cardápios dão acesso às grandes e principais funções do *software*.

Figura 25 – Exemplo de menu *pull-down*

Por geralmente apresentar uma quantidade grande de informações, subdivididas em grupos de menus, muitas vezes as opções do cardápio que são pouco utilizados pelo usuário, tornam-se ocultas, estando visíveis somente aquelas que, de acordo com o histórico de acessos, são mais freqüentemente acessadas. Muitos aplicativos utilizam-se desse artifício, sendo reversível a visualização das outras opções com o clique do *mouse* em algum sinalizador, no próprio menu, que informa a existência de informações ocultas.

4.3.6. Menus *Pallete* – Cardápios de paleta

As opções do cardápio neste tipo de interação são representadas por ícones gráficos, sendo suas escolhas geralmente mutuamente exclusivas.

A utilização maior dos cardápios de paleta é percebida em editores gráficos, onde a apresentação visual é fator facilitador do trabalho do usuário. Para efeito de aplicação de normas de usabilidade, é importante que as opções de menu contenham rótulos textuais, informando o nome ou função da ferramenta disponibilizada.

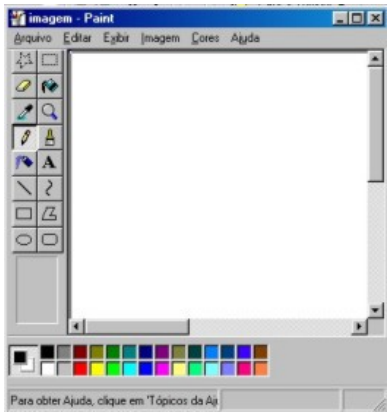


Figura 26 – Exemplo de menu *pallete*

4.4. Formulários

Um formulário é uma tela contendo campos rotulados que podem ser preenchidos pelo usuário, geralmente através de digitação ou por escolha em menus. É um estilo de interação muito utilizado hoje em dia, principalmente em aplicações via internet.

Funciona em conjunto com outro estilo de interação, a janela, como conteúdo dessa.

Seus conteúdos a serem preenchidos geralmente são apresentados como:

- texto livre. Aceita qualquer tipo de informação preenchida pelo usuário, não fazendo restrição a valores e/ou formatos;

- texto validado. Apesar de possibilitar ao usuário a digitação de valores, limita a ação deste, permitindo somente a entrada de valores com determinado formato;
- lista de escolha. É apresentada ao usuário uma relação das possíveis opções aceitadas pelo sistema, cabendo a este selecionar uma destas.

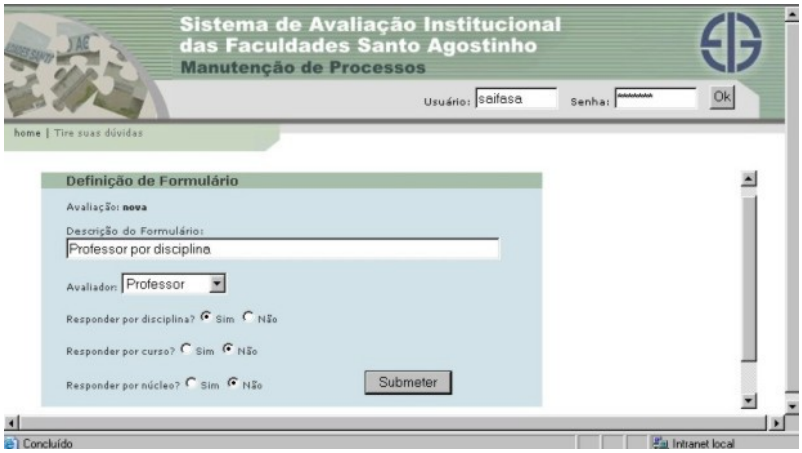


Figura 27 – Exemplo de formulário

4.4.1. Diretrizes para o projeto de formulários

- Utilizar um *lay-out* visualmente atraente e conteúdo consistente
- ;
- Reavaliar os formulários prontos no papel, nem sempre eles são ideais;

- Usar indicadores apropriados para campos no formulário;

- Usar rótulos e abreviações consistentes e familiares: CPF, CEP;

- Dar ao usuário apoio à edição e correção de erros dos campos;

- Utilizar mensagens de erros informativas e consistentes;

- Fornecer uma ajuda, ou mesmo mensagens explicativas, para preenchimento de campos;

- Fornecer valores *defaults* nos campos, quando possível;
-
- Fornecer um indicador de conclusão para formulário preenchido.

4.5. Interfaces pictóricas

Qualquer estilo de interação que proveja uma janela, botões, ícones, e outros, é geralmente chamada de interface gráfica, ou interface pictórica, do sistema.

Exemplos de interfaces pictóricas:

- Visualização científica e de dados;
- Banco de dados visual;
- Animação;
- Vídeo;
- Multimídia e Hipermídia;
- Realidade virtual.

4.6. Outros estilos de interação

Como as Tecnologias da Informação e Comunicação, principalmente a informática, vem desenvolvendo a uma velocidade surpreendente, novos estilos e modos de interação do usuários com os sistemas computacionais vem sendo desenvolvidos, sendo estes dos mais diversificados níveis de complexidade.

São utilizados, assim como os outros estilos apresentados anteriormente, de modo a facilitar a comunicação do homem com a máquina, dando cada vez mais traços de familiarização das pessoas com os equipamentos.

Além disso, esses estilos vêm suprir uma demanda muito valorizada atualmente: a questão da segurança das informações. Instrumentos são desenvolvidos de forma a prover uma maior confiabilidade e conforto ao usuário quanto à utilização de determinado sistema, das mais diversificadas funções.

Dentre estes estilos de interação, pode-se citar:

- Tela de toque;
- Síntese de fala;
- Reconhecimento de fala.

5. O projeto de interface

As atividades que envolvem o projeto da interface com o usuário visam definir formas de apoiar a realização da futura estrutura de trabalho no contexto de uso definido para o sistema. Está relacionado com os tipos de usuário, suas tarefas e o seu ambiente técnico, organizacional e social.

Nesta etapa, o projetista realiza um detalhamento da especificação do contexto de uso, processo no qual são definidas as diferentes características das Interfaces Humano-Computador.

Dentre as abordagens de projeto de Interface Homem-Computador – IHC, duas são mais conhecidas e serão aqui abordadas. Tratam-se dos modelos:

- "*The Bridge*", proposta por Tom Dayton (1996);
- "*Usage-Centered Design*", proposta por Constantine (1999).

5.1. Abordagem "*The Bridge*"

A abordagem "The Bridge" está baseada em uma seqüência de sessões de projeto participativo, envolvendo usuários, engenheiros de usabilidade, engenheiros de software, programadores, que constroem uma "ponte" entre os requisitos dos usuários e da organização e o projeto de uma interface que apoie estes requisitos.

5.1.1. Fluxo de tarefa

Nessa primeira etapa, projetistas e usuários definem um fluxo de trabalho para o sistema, a ser executado pelo usuário.
Esse fluxo é descrito por um fluxograma apresentando blocos para o início, os processos e decisões, assim como para o resultado esperado.

Esses blocos definidos devem conter:

- nomes, associados a objetos e atributos manipulados pelos usuários, e;
- verbos, associados as ações realizadas pelos usuários sobre estes objetos.

A figura abaixo mostra um exemplo de fluxo de tarefa de uma reserva em hotel.

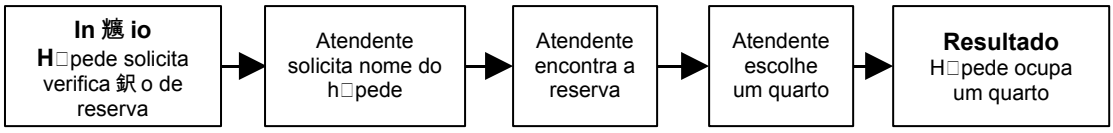


Figura 28 – Exemplo de fluxo de tarefa

Uma vez definidos, os Fluxos de Tarefas são analisados e transformados em classes de objetos de tarefas.

Os objetos de tarefas correspondem a janelas, caixas de diálogo e caixas de mensagens.

O processo de definição de classe de objetos de tarefa está associado ao conteúdos das caixas que compõem sua representação:

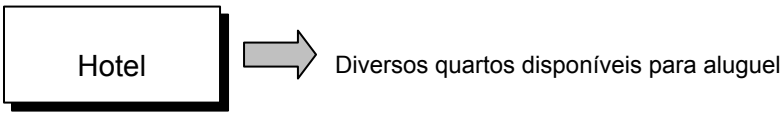
- Identificação;
- Propriedades;
- Ações;

- Relações de agregação entre objetos

.

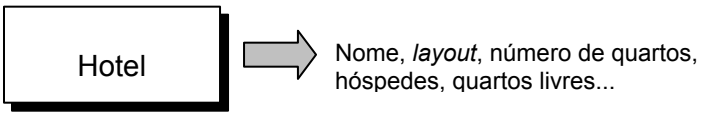
5.1.1.1. Caixa de identificação da classe

Esta definição se dá, geralmente, a partir dos substantivos nas descrições dos fluxos de tarefas.



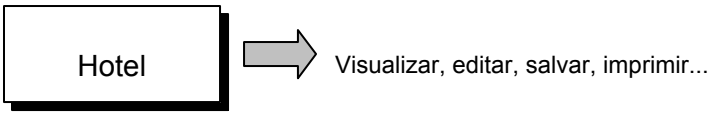
5.1.1.2. Caixa de propriedades

Seu conteúdo é definido a partir dos substantivos qualificadores que aparecem nas descrição de processos do fluxo de tarefas.



5.1.1.3. Caixa das ações

Representam as ações que os usuários podem realizar sobre estes objetos, usando-os para realizar suas tarefas. Correspondem comumente aos verbos que aparecem nas descrições dos fluxos de tarefas.



5.1.1.4. Caixa das relações de agregação entre objetos

Esta definição visa distinguir *composições* das quais uma classe faz parte e as classes de *componentes* que dela fazem parte. A composição de mais alto nível é o *desktop* no ambientes *Windows*.

Os objetos de tarefas são então mapeados e transformados em objetos de interface.

Os protótipos dos objetos de interface definidos nesta etapa, devem ter sua usabilidade testada pelos usuários participantes das sessões de projeto.

5.2. Abordagem "*Usage Centered Design*"

A abordagem para este projeto, proposto por Constantine, está baseada em três tipos de modelagem:

- Usuários e categorias relacionadas;
- Estrutura de trabalho;
- Arquitetura de interface.

5.2.1. Usuários e categorias relacionadas

A fontes de informação consideradas para a modelagem do usuário podem ser classificadas nas categorias:

- Usuários finais;
- Consumidores e gerentes;
- Especialistas no domínio, pessoal de treinamento, supervisores;
- Pessoal de marketing, vendas, apoio técnico, documentaristas.

Outras fontes para a modelagem do usuário incluem manuais, questionários, e qualquer outra forma de informação disponível nas empresas.

5.2.1.1. Papel do usuário

A principal componente do modelo de usuários é o papel de usuário. Definido como um conjunto abstrato de necessidades, interesses, expectativas, comportamentos e responsabilidades, caracterizando um relacionamento entre classes ou tipos de usuários e o sistema.

São características desta modelagem:

- Usuários divididos em categorias;
- Cada categoria tem acesso a diferentes funções do sistema;
- Categorias de usuários e suas funções devem ser devidamente documentadas no projeto do sistema;
- A definição de papeis focais, ou seja, a que tipo de usuário o sistema será prioritariamente destinado.

5.2.2. Estruturas de trabalho

Para entender os componentes da abordagem de Constantine para modelagem de estruturas de trabalho, é necessário esclarecer alguns conceitos-chave, em particular Cenários, Casos de Uso e Casos de Uso Essenciais.

5.2.2.1. Cenários

Descrição concreta e detalhada de uma sequência de eventos em uma situação específica.

5.2.2.2. Casos de Uso
Descrição narrativa da interação entre um usuário e alguma parte do sistema.

5.2.2.3. Casos de Uso Essenciais
Descrição simplificada, generalizada e livre de detalhes de tecnologia, de uma tarefa do sistema a ser realizada pelo usuário.

5.2.3. Arquitetura de interface
Uma arquitetura de interface capaz de apoiar usuários e os casos de uso definidos até agora no projeto, é feita pela elaboração de dois modelos:

- Modelo de conteúdo de interface;
- Mapa de navegação entre contextos.

5.2.3.1. Modelo de conteúdo de interface
É a apresentação interna, de conteúdos de vários contextos de interação e externa, das interconexões entre os contextos. Este modelo d
eriva do modelo de casos de uso.

O modelo de conteúdo de interface é composto por:

- Ferramentas abstratas – fornecem as funções;
- Materiais abstratos – fornecem os dados, as apresentações e as áreas de trabalho sobre as quais as ferramentas devem ser operadas.

O processo inicia-se pela analise das narrativas de casos de uso, linha por linha identificando quais ferramentas e materiais serão fornecidos de forma que o usuário seja capaz de realizar a interação.

Cada espaço abstrato de interação irá se tornar uma tela, janela ou caixa de diálogo que precisa ser entendida pelo usuário.

5.2.3.2. Mapa de navegação entre contextos
Um mapa de navegação é, na verdade, um diagrama de transição de estados, onde os espaços de interação são representados por retângulos e as transições por flechas, conectando os espaços.

Envolve a decisão entre limitar o conteúdo de cada espaço de interação, mantendo-os pequenos e simples, ou limitar o número deles, aumentando a complexidade de cada um.

A figura abaixo mostra um exemplo de mapa de navegação entre contextos de uma compra em supermercado.

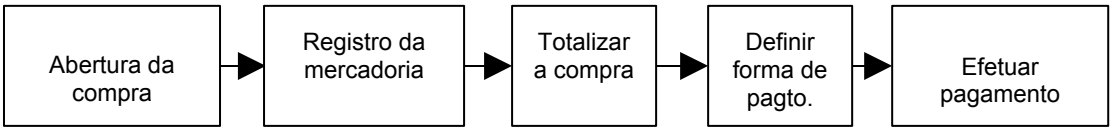


Figura 29 – Exemplo de mapa de navegação entre contextos

6. Usabilidade de sistemas computacionais

A Usabilidade é uma metodologia de definição do processo de *design* de sistemas computacionais. Tem por objetivo facilitar o aprendizado e uso do usuário, além de criar ambientes agradáveis para as pessoas.

O processo de *design* para usabilidade, é uma recomendação de alguns pesquisadores independentes, dentre eles Jakob NIELSEN, que já na década de 80 constataram que confiar na experiência de *designer* e em padrões, ou em 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.

Foram definidas algumas fases do processo de definição de usabilidade, como consenso desses grupos iniciais de pesquisadores, que entenderam o ciclo tradicional de desenvolvimento que começava com a definição do produto e terminava com sua entrega.

6.1. Fases do Processo de *Design* para Usabilidade

De maneira geral, o processo de *design* ou definição de usabilidade de um sistema é composto por algumas fases, a saber:

- Pré-*design*
- *Design* inicial
- Desenvolvimento iterativo
- Pós-*design*

6.1.1. Pré-*design*

Nessa etapa serão colhidas informações importantes antes de se dar início à estruturação do *design* do sistema. Envolve o conhecimento do usuário, como suas características individuais (escolaridade, idade, experiência com computadores...) para posteriormente, definir que funções do sistema cada um desses tipos utilizarão.

6.1.2. *Design* inicial

É o primeiro momento do contato do usuário com o que poderá ser a vir o *design* do sistema. É a partir desse primeiro contato que parte-se para as implementações no *design.* Recomenda-se o uso de métodos participativos, uma vez que, embora os usuários não sejam *designers*, são muito bons em reagir a *design* que não os agrade ou não funciona na prática.

6.1.3. Desenvolvimento Iterativo

Essa fase, posterior à fase de apresentação inicial, é baseada na prototipagem e testes empíricos. Tem como premissa básica que não se consegue que o sistema dê certo logo na primeira vez, não importando quão experiente o *designer* seja. Não se saberá se o sistema está funcionando até que se comece a testa-lo.

6.1.4. Pós-*design*

Conduz estudos de campo do produto em uso, para obter dados para nova versão e produtos futuros. Busca avaliar o impacto do produto na qualidade do trabalho do usuário.

6.2. Slogans de Usabilidade

Os slogans de usabilidade, definidos por NIELSEN, funcionam como uma orientação aos desenvolvedores de sistemas quanto a que aspectos observar no momento da estruturação da interface desses sistemas.

6.2.1. Sua melhor tentativa não é boa o suficiente:

- É impossível fazer uma boa interface simplesmente baseada em nossas melhores idéias;
- Usuário possui potencial infinito para mal interpretar;
- Melhor Design: baseado no entendimento do usuário e de suas tarefas.

6.2.2. Usuário está sempre certo

- Atitude errada do designer: julgar que o usuário é ignorante e/ou não tentou o suficiente;
- Designer de interfaces deve adquirir humildade em aceitar a necessidade de modificar uma “grande idéia” de forma a resolver problemas dos usuários.

6.2.3. Usuário não está sempre certo

- Não ir ao extremo e construir interfaces que o usuário gostaria. Eles não sabem o que é bom para eles.
- Tendência humana: rejeitar qualquer grande inovação em objetos com os quais estamos familiarizados e que atendem satisfatoriamente nossas necessidades.

6.2.4. Usuários não são designers

- Prover interfaces flexíveis que pudessem ser amplamente customizadas, cada usuário teria exatamente a interface que melhor lhe satisfizesse.
- Porém, estudos demonstram que usuários não customizam interfaces mesmo quando estas estão disponíveis.
- Motivos para não se dar tanta importância à Customização:
    1. É fácil apenas quando produzi um design coerente a partir do conjunto de opções disponíveis.

2. Ela vai exigir uma interface – adiciona complexidade
3. Leva o usuário a ter interface muito diferente do outro usuário.
4. Usuários nem sempre adotam as decisões de design mais apropriadas

6.2.5. Designers não são usuários

- Designers já utilizam computadores, possuem experiência computacional e conhecimento dos fundamentos conceituais do design do sistema.
- Não dá para voltar atrás e fazer o papel do novato.

6.2.6. Menos é mais

- Frase errada: “Se tudo está disponível então todos ficarão satisfeitos.”
- Cada elemento da interface acarreta sobrecarga ao usuário que tem que considerar se o usa ou não.
- Ter poucas opções às necessárias tarefas, geralmente significa ter melhor usabilidade, pois o usuário pode se concentrar em entender as poucas opções.

6.2.7. Help não ajuda

- Usuários perdidos tentando encontrar informação na enorme quantidade de material de help e quando encontra não consegue entendê-lo.
- Help não pode ser usado como desculpa para um design ruim
- Requisito básico: Melhor operar um sistema sem ter que usar help :

6.3. Atributos de Usabilidade

Tratam a tarefa e as características individuais dos usuários.

6.3.1. Facilidade de Aprendizagem

- Sistema tem que ser fácil de aprender de forma que o usuário possa rapidamente começar a interagir.
- Usuário não aprende uma interface antes de começar a usa-la.
- Avaliação: em função do tempo que o usuário demora para atingir um suficiente grau de proficiência na execução de suas tarefas.

6.3.2. Eficiência

- Uma vez aprendido a utilizar a interface, o usuário tem que possuir um elevado nível de produtividade.
- Avaliação: definir o que significa usuário experiente e avaliar um grupo desses executando tarefas típicas de um sistema

6.3.3. Facilidade de relembrar

- Uma vez usado a interface e voltar após a algum tempo a reusá-la não precisar aprendê-la novamente.
- Facilidade de aprendizado torna mais fácil de ser relembrada.

- As modernas interfaces ajudam o usuário a relembrar o que está disponível quando necessário.

6.3.4. Erros

- Ação que não leva ao resultado esperado.
- Sistema deve possuir pequena taxa de erros e estas devem ser de fácil recuperação, sem haver perda de trabalho.

6.3.5. Satisfação subjetiva

- Se os usuários gostam do sistema
- Sistema deve ser agradável de forma que o usuário fique satisfeito ao usa-lo.
- Relevância em sistemas fora do ambiente de trabalho: jogos, sistemas domésticos em geral.
- *Satisfação subjetiva* como atributo de usabilidade é diferente dos estudos gerais das pessoas com relação aos computadores apesar dos sentimentos que os usuários tem em relação aos computadores afetarem sua interação com um determinado sistema.
- Avaliação: perguntando ao usuário suas opiniões subjetivas.

6.4. O envolvimento do usuário no projeto

A forma de tratamento do usuário, outro aspecto importante no estudo da usabilidade, envolver alguns elementos:

- Conhecer o usuário: fator fundamental
- Classificar usuários ajuda a fazer um bom design atendendo a maior diversidade desses.
- Experiência: fator relevante, analisada em 3 dimensões:
  1. relação ao uso do sistema
     - Algumas interfaces projetadas apenas para novatos: Sistemas de museus, quiosques, etc (requisito básico : aprendizagem)
     - Uso do sistema altera o tipo de usuário e produz implicações no design, por isso deve-se acomodar mais de um estilo (usuário tem dificuldade de iniciar em um estilo+facil e depois migrar para outro +eficiente)
     - Acelerador de Interfaces:: elementos de interface que permitem que usuários realizem tarefas freqüentes de forma mais rápida. Ex.: teclas de função, abreviação de nomes de comandos, uso de duplo clique para ativar objetos. (Pode acarretar complexidade da interface e problemas por isso)
  2. relação ao uso de computadores
     - Usuários experientes tem idéias de que características procurar e de como o computador normalmente trata várias situações.
  3. relação ao domínio da aplicação
     - Interfaces projetadas para especialistas podem fazer uso de terminologia e jargão específico de uma área. Usuários com pouca experiência terão que ter mais explicação sobre o que o sistema faz e sobre o que as diferentes opções significam.

Além da experiência deve-se observar:

- Idade;

- Sexo;
- Habilidades;
- formação cultural
- complexidade em obter ótimos graus de usabilidade em todos os atributos simultaneamente
- estabelecer os objetos de usabilidade a serem atingidos e priorizados de acordo o projeto.

7. Avaliação de Interfaces

O processo de avaliação de uma interface não deve ser avaliado como uma fase única dentro do processo de *design* de um sistema, muito menos encarada como uma última etapa deste, sendo realizada somente quando finalizado. O ideal é que a avaliação ocorra durante o ciclo de vida de *design* do produto, de modo que seus resultados sejam utilizados para melhorias gradativas da interface.

Na maioria dos modelos de desenvolvimento de interfaces que utilizam os conceitos de usabilidade, a avaliação tem um papel central, ocorrendo constantemente, como no modelo Estrela, apresentado no capítulo 3.

Diferentes tipos de avaliação são utilizados em diferentes estágios do *design*, desde testes informais no primeiro estágio (onde as idéias estão sendo descobertas e exploradas), até avaliações mais formais, utilizadas em estágios mais avançados do processo.

Os fatores determinantes de um plano de avaliação incluem:

- estágio do *design*;
- quão pioneiro é o projeto;
- número esperado de usuários;
- quão crítica é a interface;
- custo do produto e orçamento alocado para o teste;
- tempo disponível;
- experiência dos *designers* e avaliadores.

Serão apresentados aqui alguns testes que poderão ser aplicados para se verificar a capacidade da interface em alcançar seu objetivo.

7.1. Objetivos da avaliação

O objetivo principal da avaliação de uma interface é conhecer o que os usuários querem e os problemas que eles percebem. Sabe-se que, quando melhor informados os *designers* estiverem sobre seus usuários, melhor será o *design* do produto.

Questões específicas podem ser o alvo de determinada avaliação, como a avaliação da aceitação de alguma alteração na estrutura de um produto, como também podem ser mais abrangentes, como com o objetivo de verificar se as idéias pensadas pelo *designer* são realmente o que os usuários necessitam ou desejam.

Portanto, as avaliações podem ocorrer tanto durante o desenvolvimento do produto quanto depois desse finalizado.

Pode-se dizer que a avaliação tem três grandes objetivos:

- avaliar a funcionalidade do sistema;
- avaliar o efeito da interface junto ao usuário;
- identificar problemas específicos do sistema.

7.1.1. Avaliando a funcionalidade

A importância da avaliação da funcionalidade está no sentido de verificar se a interface está adequada aos requisitos da tarefa do usuário, ou seja, se o *design* do sistema permite ao usuário efetuar a tarefa pretendida de modo mais fácil e eficiente. A 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.

7.1.2. Avaliando o efeito da interface junto ao usuário

É a avaliação da usabilidade. Inclui considerar aspectos como: avaliar a facilidade em aprender a usar o sistema; a atitude do usuário com relação ao sistema; identificar áreas que sobrecarregam o usuário de alguma forma, exigindo que uma série de informações sejam relembradas.

7.1.3. Avaliando problemas específicos do sistema

Tem o objetivo de identificar aspectos do *design* que, quando usados no contexto alvo, causam resultados inesperados ou confusão entre os usuários. Está relacionado tanto com a funcionalidade quanto com a usabilidade.

Os métodos de avaliação podem ser classificados em duas dimensões: se os usuários reais estão ou não envolvidos e se a interface está ou não implementada. Assim tem-se:

- Inspeção de usabilidade: não envolve usuários e podem ser utilizadas em qualquer fase do desenvolvimento de um sistema (implementado ou não);
- Teste de usabilidade: centrado no usuário. Para se usar esse método 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 implementado, um cenário, ou até a implementação completa.

7.2. Inspeção de usabilidade

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 *designer* 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.

Dos métodos existentes, pode-se citar como mais efetivos na avaliação de interfaces, os **métodos empíricos** ou **testes de usabilidade**.

Dentre esses métodos pode-se destacar:

- Avaliação heurística
- Revisão de *guidelines*

- Inspeção de consistência
- Percurso cognitivo

7.2.1. Avaliação heurística

Existem alguns métodos de inspeção bastante caros, tornando-se fora da realidade a sua aplicação. Nielsen propõe, entretanto, alguns métodos que são baratos, rápidos e fáceis de serem usados. A avaliação heurística é o principal método dessa proposta.

Esse tipo de avaliação deve ser vista como parte do processo de *design* interativo de uma interface. Envolve um pequeno conjunto de avaliadores examinando a interface e julgando suas características em face de reconhecidos princípios de usabilidade, denominados heurísticas. As heurísticas são regras gerais que objetivam descrever propriedades comuns de interfaces usáveis.

A experiência tem mostrado que diferentes pessoas encontram diferentes problemas, por isso a importância de se ter múltiplos avaliadores. É recomendado 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 detectar problemas os relata associando-os claramente com as heurísticas de usabilidade que foram violadas.

7.2.1.1. Heurísticas de usabilidade

1. Visibilidade do status do sistema

O sistema precisa manter os usuários informados sobre o que está acontecendo, fornecendo um *feedback* dentro de um tempo razoável.

2. Compatibilidade do sistema com o mundo real

O sistema precisa falar a linguagem do usuário, com palavras, frases e conceitos familiares aos usuários, ai 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 freqüentemente escolhem por engano funções do sistema e precisam ter claras saídas 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. Segui 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 ocorrer.

6. Reconhecimento ao invés de relembrança

Tornar objetos, ações e opções visíveis. O usuário não deve ter que lembrar informações 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 forma a aumentar a velocidade de interação. Permitir a usuários experientes “cortar caminho” em funções 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.

Depois dessa etapa inicial, as listas de problemas dos avaliadores são consolidadas em uma só. Geralmente uma sessão de avaliação, em sua etapa individual, dura cerca de duas horas. Sessões mais extensas podem ser necessárias para o caso de interfaces muito grandes ou muito complexas, com um número considerável de componentes de diálogo. Nesse caso é aconselhável dividir a avaliação em pequenas sessões, cada qual avaliando um cenário específico de interação.

Como dito, o resultado da avaliação heurística é uma lista de problemas de usabilidade da interface com referência aos princípios 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 problemas, não é difícil gerar um *design* revisado baseado nas diretrizes que foram providas pelo princípio de usabilidade violado.

7.2.1.2. Exemplo de problemas detectados através da avaliação heurística

Apresenta-se abaixo problemas isolados que foram detectados em avaliações heurísticas, de forma a deixar mais claro o tipo de resultados que podem ser obtidos pelo método.

Exemplo 1
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 a opção ‘Não’ está a direita da opção ‘Sim’, conforme figura 30. Isso foge completamente do padrão de diálogo de toda aplicação Windows em caixas de diálogo do tipo ‘Sim/Não’, sendo fonte constante de erro (vai contra as heurísticas **consistência e padrões** e **prevenção de erros**). Vale ressaltar 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. Outra aspecto a ser observado é o uso indevido da palavra 'abortar' na caixa de diálogo (vai contra a heurística **compatibilidade do sistema com o mundo real**).

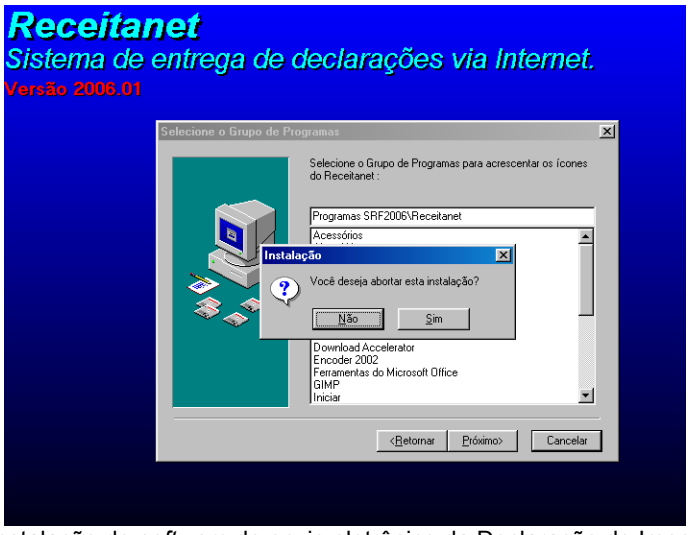


Figura 30 – Instalação do *software* de envio eletrônico da Declaração de Imposto de Renda

Exemplo 2

No Windows Explorer, ao tentar excluir um arquivo que está em uso, uma caixa de diálogo é aberta, conforme mostra figura 31. Nessa caixa aparece a mensagem de que não foi possível 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 o disco cheio com excluir um arquivo? (vai contra a heurísitca **ajudar o usuário a reconhecer, diagnosticar e corrigir erros**)

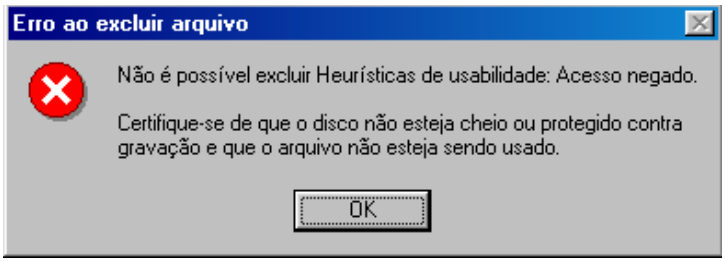


Figura 31 – Mensagem de erro do Windows Explorer, ao tentar excluir um arquivo em uso

7.2.1.3. Graus de severidade

Adicionalmente à lista de problemas de usabilidade detectados, 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 freqüência 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.

É preciso considerar o impacto do problema no mercado, por muitos problemas simples de serem superados tem um efeito importante na popularidade de um produto. Assim verifica-se a necessidade ou não de se adequar o sistema de maneira que seja corrigido o problema, identificado como um simples problema estético ou um problema catastrófico, que possa prejudicar a comercialização e/ou aceitação do sistema pelo público de usuários.

7.2.2. Percurso cognitivo

Método de inspeção 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. 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* a medida que delas necessitem.

Portanto, tem-se que o 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 (protótipo executável), 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, a fase preparatória e a fase de análise, conforme quadro abaixo.

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 informa 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 quatro 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?

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 intenções 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.

Um percurso 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*.

7.2.2.1. Fase preparatória

Deve-se analisar os pontos:

- Quem são os usuários do sistema?
  Pode ser uma descrição simples e geral, como por exemplo, 'pessoas que utilizam o Windows'. Mas o processo é mais revelador se a descrição inclui mais especificamente a experiência e conhecimento técnico que podem influencias os usuários na interação com uma nova interface. Por exemplo, usuários podem ser 'usuários do Windows que utilizam o Microsoft Excel'. No processo de percurso são considerados o conhecimento do usuário com relação à tarefa e com relação à interface.

- Que tarefa(s) 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. Para sistemas com complexidade maior, a análise deverá ser limitada a uma razoável, mas representativa, coleção de tarefas.

- Qual é 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, podem ser seqüências de diversas ações simples que o usuário típico pode executar como um bloco, tais como 'logar no sistema', ou 'salvar como do Menu Arquivo'. Depende do nível de conhecimento do usuário-alvo.

- 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.

7.2.2.2. Fase de análise

Essa fase 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.

As características 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 alcançando 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 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 ‘imprimir’, 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 HP Deskjet 1220’ 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.

7.2.2.3. Registro da informação durante a avaliação

Durante o percurso é importante registra toda informação gerada. É conveniente gravar na forma de videoteipe todo o processo de avaliação, incluindo comentário dos avaliadores.

Quanto as 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

7.2.2.4. Exemplo de estória de sucesso

Um usuário experiente em Windows inicia uma tarefa dando um clique no ícone da aplicação para abri-la. Assim, tem-se:

- usuário abre a aplicação porque ele sabe que deve abrir uma aplicação para usa-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 início da aplicação.

Deve-se notar que as três primeiras partes não seriam válidas para uma pessoa novata no uso de computadores, e a segunda e terceira não seriam válidas para pessoas inexperientes em Windows.