Projeto de Interfaces de Usuário

Perspectivas Cognitivas e Semióticas

Clarisse Sieckenius de Souza[*], Jair Cavalcanti Leite[†],
Raquel Oliveira Prates[*], Simone D.J. Barbosa[*]

clarisse@inf.puc-rio.br; jair@dimap.ufrn.br;
raquel@les.inf.puc-rio.br; sim@les.inf.puc-rio.br

Resumo

A área de Interação Humano–Computador (IHC) tem por objetivo principal fornecer aos pesquisadores e desenvolvedores de sistemas *explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário* [ACM SIGCHI, 1992]. Esta apostila apresenta duas bases teóricas que fundamentam os estudos de IHC, e descreve o processo de *design* de interfaces. Apresenta também os modelos e técnicas utilizados na análise e modelagem de usuários, tarefas e comunicação; e dois métodos para avaliação das interfaces resultantes.

Abstract

The main goal of the Human–Computer Interaction (HCI) area is to provide researchers and developers with explanations and predictions about user-system interaction phenomena, and practical results for user interface design [ACM SIGCHI, 1992]. This material presents two complementary theoretical bases for HCI research, and describes the user interface development process. It presents models and techniques used in users, tasks, and communication analysis and modelling; and two methods for evaluating the resulting user interfaces.

1. INTRODUÇÃO

1.1 Visão Geral

Nas últimas décadas, tem sido dada cada vez maior importância à interface de aplicações computacionais. A interface de uma aplicação computacional envolve todos os aspectos de um sistema com o qual mantemos contato [Moran, 1981]. É através da interface que os usuários têm acesso às funções da aplicação. Fatores de satisfação subjetiva, de eficiência, de segurança, de custo de treinamento, de retorno de investimento, todos, dependem de um bom *design* de interface.

Na indústria de software, o *design* de interface tem sido conduzido através de processos iterativos de construção e avaliação de protótipos baseados em princípios e diretrizes empíricas, tal como proposto em *The Windows Interface:*

[*] Pontifícia Universidade Católica do Rio de Janeiro — PUC-Rio
[†] Universidade Federal do Rio Grande do Norte — UFRN

Guidelines for Software Design [Microsoft, 1995] e *Macintosh Human Interface Guidelines* [Apple, 1992]. Entretanto, estes princípios podem ser conflitantes em determinadas situações. Para resolvê-los, é necessário basear a prática de *design* de interfaces em uma fundamentação teórica [Hartson, 1998]. Esta fundamentação orientará o *designer* ao longo da elaboração da sua solução particular para o conjunto de problemas que a aplicação pretende resolver.

A área de Interação Humano–Computador (IHC) tem por objetivo principal fornecer aos pesquisadores e desenvolvedores de sistemas *explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário* [ACM SIGCHI, 1992]. Com teorias a respeito dos fenômenos envolvidos seria possível prever antecipadamente se o sistema a ser desenvolvido satisfaz as necessidades de usabilidade, aplicabilidade e comunicabilidade dos usuários. Para isto, estudos de IHC visam desenvolver modelos teóricos de desempenho e cognição humanos, bem como técnicas efetivas para avaliar a usabilidade [Lindgaard, 1994]. Mais recentemente algumas propostas têm enfatizado que além de usabilidade, as aplicações devem buscar atingir aplicabilidade [Fischer, 1998] e comunicabilidade [de Souza, 1999], oferecendo ao usuário artefatos fáceis de usar, aplicar e comunicar.

IHC é uma área multidisciplinar, que envolve disciplinas como [Preece et al., 1994]: Ciência da Computação; Psicologia Cognitiva; Psicologia Social e Organizacional; Ergonomia ou Fatores Humanos; Lingüística; Inteligência Artificial; Filosofia, Sociologia e Antropologia; Engenharia e *Design*.

No contexto de IHC devemos considerar quatro elementos básicos: *o sistema, os usuários, os desenvolvedores e o ambiente de uso* (domínio de aplicação) [Dix et al., 1993]. Estes elementos estão envolvidos em dois processos importantes: a *interação usuário-sistema* e o *desenvolvimento do sistema*. O *curriculum* proposto para IHC identifica cinco enfoques para o estudo destes elementos e para a sua aplicação na melhoria dos processos de desenvolvimento e de interação usuário-sistema. Para cada um destes focos, diferentes disciplinas proporcionam os estudos teóricos que podem ser aplicados ao desenvolvimento. São eles:

- *design* e desenvolvimento do hardware e software: estudo de tecnologias de dispositivos de entrada e saída; e tecnologias de software, como ambientes gráficos e virtuais.
- estudo da capacidade e limitação física e cognitiva dos usuários: considera estudos de ergonomia para avaliar limites de esforço físico do usuário, e estudos de psicologia e ciência cognitiva sobre a capacidade humana de memorização, raciocínio e aprendizado.
- instrumentação teórica e prática para o *design* e desenvolvimento de sistemas interativos: envolve o conhecimento teórico a respeito dos fenômenos envolvidos; modelos para o processo de desenvolvimento que descrevam as etapas necessárias e como devem ser conduzidas; diretrizes, técnicas, linguagens, formalismos e ferramentas de apoio a estas etapas.
- modelos de interfaces e do processo de interação usuário–sistema: para desenvolver modelos abstratos do processo de interação compatíveis com as capacidades e limitações físicas e cognitivas dos usuários.

- análise do domínio e de aspectos sociais e organizacionais: para avaliar o impacto que o contexto onde está inserido o usuário exerce sobre seus conhecimentos, sua linguagem e suas necessidades.

1.2 Organização da apostila

Esta apostila está organizada em quatro capítulos. No Capítulo 1, apresentamos conceitos básicos da área de IHC que serão utilizados ao longo do texto. Apresentamos a evolução de IHC por diferentes perspectivas, e descrevemos os principais estilos de interação. O Capítulo 2 descreve as teorias da Engenharia Cognitiva e da Engenharia Semiótica e faz uma comparação entre elas. O Capítulo 3 descreve um modelo para o processo de *design* de uma aplicação de acordo com estudos de IHC, e as etapas de modelagem de usuários, tarefas e comunicação. Apresenta também as técnicas de cenários e storyboarding, e descreve ferramentas de apoio à construção de interfaces. No Capítulo 4, apresentamos métodos de testes de usabilidade e comunicabilidade para avaliação de interfaces, e descrevemos também os tipos de prototipação e seu papel na etapa de avaliação.

1.3 Conceitos Básicos

Interface e Interação

O termo **interface** é aplicado normalmente àquilo que interliga dois sistemas. Tradicionalmente, considera-se que uma interface homem-máquina é a parte de um artefato que permite a um usuário controlar e avaliar o funcionamento deste artefato através de dispositivos sensíveis às suas ações e capazes de estimular sua percepção. No processo de *interação* usuário-sistema a interface é o combinado de software e hardware necessário para viabilizar e facilitar os processos de comunicação entre o usuário e a aplicação. A interface entre usuários e sistemas computacionais diferencia-se das interfaces de máquinas convencionais por exigir dos usuários um maior esforço cognitivo em atividades de interpretação e expressão das informações que o sistema processa [Norman, 1986].

Moran propôs uma das definições mais estáveis de interface, dizendo que "*a interface de usuário deve ser entendida como sendo a parte de um sistema computacional com a qual uma pessoa entra em contato física, perceptiva e conceitualmente*" [Moran, 1981]. Esta definição de Moran caracteriza uma perspectiva para a interface de usuário como tendo um componente físico, que o usuário percebe e manipula, e outro conceitual, que o usuário interpreta, processa e raciocina. Moran e outros denominam este componente de *modelo conceitual do usuário.*

Vemos, pois, que a interface é tanto um *meio* para a interação usuário-sistema, quanto uma *ferramenta* que oferece os instrumentos para este processo comunicativo. Desta forma a interface é um *sistema de comunicação*.

A interface possui componentes de software e hardware. Os componentes de hardware compreendem os dispositivos com os quais os usuários realizam as atividades motoras e perceptivas. Entre eles estão a tela, o teclado, o mouse e vários outros. O *software da interface* é a parte do sistema que implementa os processos computacionais necessários (a) para controle dos dispositivos de

hardware, (b) para a construção dos dispositivos virtuais (os *widgets*) com os quais o usuário também pode interagir, (c) para a geração dos diversos símbolos e mensagens que representam as informações do sistema, e finalmente (d) para a interpretação dos comandos dos usuários.

Outra característica de uma interface é a revelação das *affordances* do sistema. ***Affordance*** é um termo que se refere às propriedades percebidas e reais de um artefato, em particular as propriedades fundamentais que determinam como este artefato pode ser utilizado [Norman, 1988]. Segundo Norman, as *affordances* fornecem fortes pistas ou indicações quanto à operação de artefatos; e quando se tira proveito delas, o usuário sabe exatamente o que fazer só olhando para o artefato. Por exemplo, a *affordance* de um botão é que o pressionemos, de um interruptor, que o comutemos, e assim por diante.

A **interação** é um processo que engloba as ações do usuário sobre a interface de um sistema, e suas interpretações sobre as respostas reveladas por esta interface (Figura 1.1). Ao longo deste texto, veremos como modelar e avaliar este processo.

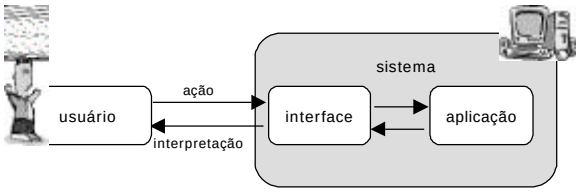


Figura 1.1 — Processo de interação humano–computador.

Usabilidade

A usabilidade de um sistema é um conceito que se refere à qualidade da interação de sistemas com os usuários e depende de vários aspectos. Alguns destes fatores são:

- **facilidade de aprendizado** do sistema: tempo e esforço necessários para que os usuários atinjam um determinado nível de desempenho;
- **facilidade de uso**: avalia o esforço físico e cognitivo do usuário durante o processo de interação, medindo a velocidade de e o número de erros cometidos durante a execução de uma determinada tarefa;
- **satisfação do usuário**: avalia se o usuário gosta e sente prazer em trabalhar com este sistema;
- **flexibilidade**: avalia a possibilidade de o usuário acrescentar e modificar as funções e o ambiente iniciais do sistema. Assim, este fator mede também a capacidade do usuário utilizar o sistema de maneira inteligente e criativa, realizando novas tarefas que não estavam previstas pelos desenvolvedores;
- **produtividade**: se o uso do sistema permite ao usuário ser mais produtivo do que seria se não o utilizasse.

Muitas vezes, o *designer* deve identificar quais destes fatores têm prioridade sobre quais outros, uma vez que dificilmente se consegue alcançar todos de forma equivalente. As decisões do projetista determinam a forma de interação entre usuários e sistemas. Freqüentemente *designers* definem a facilidade de uso como sendo o aspecto de usabilidade prioritário e, por vezes, acabam desenvolvendo sistemas em que os usuários não cometem erros, mas também não têm muita opção de ação ou decisão. Adler e Winograd chamam estes sistemas de sistemas anti-idiotas (*idiot-proof*) e advogam que novas tecnologias serão mais eficazes quando projetadas para aumentar, ao invés de substituir, as capacidades dos usuários [Adler & Winograd, 1992]. Assim, eles denominam *desafio de usabilidade* o projeto de novas tecnologias que buscam explorar ao máximo as capacidades dos usuários na criação de ambientes de trabalho mais eficazes e produtivos.

Outros pesquisadores também têm ressaltado a importância de os sistemas computacionais ampliarem as capacidades do usuário. Norman, um dos mais influentes pesquisadores e um dos pioneiros na aplicação de psicologia e ciência cognitiva ao *design* de interfaces de usuário, recentemente tem enfatizado que a tecnologia deve ser projetada com o objetivo de ajudar as pessoas a serem mais espertas, eficientes e inteligentes [Norman, 1991; Norman, 1993]. Fischer, por sua vez, argumenta que além de usabilidade o *designer* deve buscar atingir também aplicabilidade, ou seja, a sua utilidade na resolução de problemas variados [Fischer, 1998]. Ele insiste no fato de que todo usuário é especialista em um domínio e uma aplicação de software deve servir à sua especialidade. Neste sentido ela deve funcionar como um instrumento para o usuário e não presumir que o usuário é quem deve atender às exigências de peculiaridades tecnológicas.

Comunicabilidade

A comunicabilidade de um sistema é a sua propriedade de transmitir ao usuário de forma eficaz e eficiente as intenções e princípios de interação que guiaram o seu *design.* Faz parte da experiência diária das pessoas entender através de um ato de comunicação muitas mensagens subjacentes, como por exemplo perceber a filosofia de um autor através da sua peça de teatro. Da mesma forma, o objetivo da comunicabilidade é permitir que o usuário, através da sua interação com a aplicação, seja capaz de compreender as premissas, intenções e decisões tomadas pelo projetista durante o processo de *design.* Quanto maior o conhecimento do usuário da lógica do *designer* embutida na aplicação, maiores suas chances de conseguir fazer um uso criativo, eficiente e produtivo da aplicação. Junto com a usabilidade, a comunicabilidade pretende aumentar a aplicabilidade de software.

Para ilustrar, a Figura 1.2 mostra uma interface de alta comunicablidade, a de um software para tocar CD’s. Através da analogia com um o aparelho de tocar CD’s esta interface comunica de forma efetiva como os usuários devem interagir para tocarem seus CD’s no computador.

Figura 1.2 — Exemplo de alta comunicabilidade: interface de um software para tocar CD's.

Para contrastar, a Figura 1.3 mostra uma interface de baixa comunicabilidade. O usuário de um sistema Microsoft Windows 9x/NT executa a aplicação Find (Localizar) para procurar um computador e fornece o nome LINX. O computador não é encontrado, e uma mensagem com esta informação aparece para o usuário na barra de status na borda inferior da janela. O usuário analisando a interface percebe que a opção de menu Edit (Editar) está disponível, assim como o item Undo (Desfaz). No entanto, o usuário não sabe o que pode ser desfeito, uma vez que a única coisa feita foi uma busca sem sucesso. Ao ativar o item Undo, o usuário então recebe uma mensagem informando que o arquivo não pode ser removido. Neste momento, o usuário não tem como saber a que o sistema se refere, pois o seu discurso, claramente não está relacionado com a tarefa em questão.

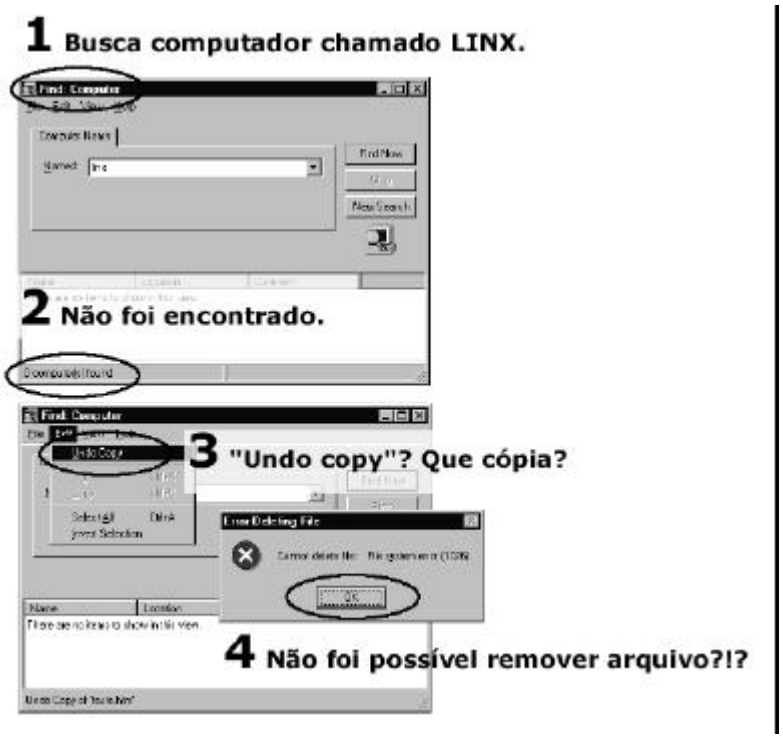


Figura 1.3 — Exemplo de baixa comunicabilidade: interface do software de localização de computadores e arquivos no Windows 95.

1.4 Perspectivas em IHC

Para compreender melhor as teorias de *design* de interface, precisamos entender as diferentes perspectivas que os sistemas de computador vêm atravessando ao longo do tempo (Figura 1.4) [Kaamersgard, 1988]. Inicialmente, o usuário era considerado uma máquina, que tinha que aprender a falar a linguagem do computador. Em seguida, com o surgimento da Inteligência Artificial, tentamos considerar o computador como uma pessoa. Nessas duas perspectivas, era fundamental dar poder ao sistema. Mais tarde, surgiu a perspectiva de computador como ferramenta, que o usuário utiliza para obter um resultado ou produto. Atualmente vemos outra mudança de perspectiva, na qual o computador é um mediador da comunicação entre pessoas. Nestas duas últimas perspectivas, o foco é no usuário, e não mais no sistema.

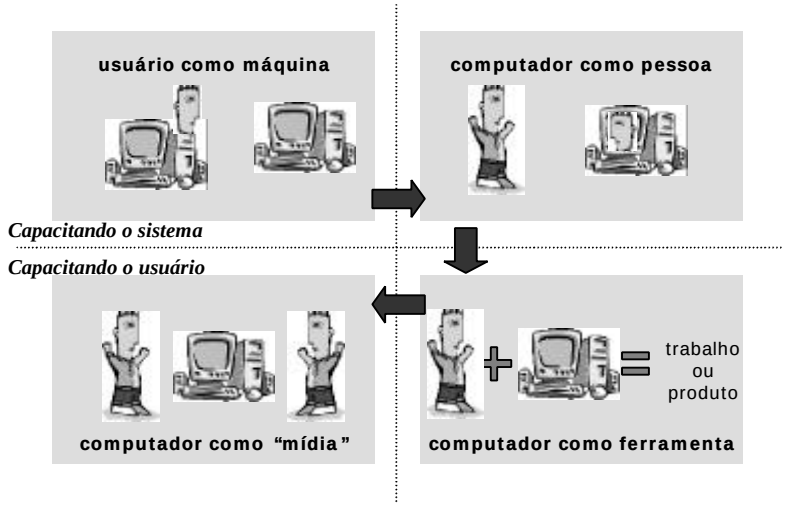


Figura 1.4 — Perspectivas em IHC.

O abordagem de desenvolvimento de sistemas centrado do usuário (User-Centered System *Design*, ou UCSD [Norman, 1986]) se aplica à perspectiva "computador como ferramenta". Já a abordagem da Engenharia Semiótica se aplica às duas últimas perspectivas, que ocorrem nas aplicações de software atuais. Estas abordagens serão descritas no Capítulo 2.

1.5 Estilos de Interação

Estilo de interação é um termo genérico que inclui todas as formas como os usuários se comunicam ou interagem com sistemas computacionais [Preece et al., 1994; Shneiderman, 1998]. Neste curso, destacamos os seguintes estilos de interação: linguagem natural, linguagens de comando, menus, WIMP, preenchimento de formulário e manipulação direta.

Além do estilo de interação, o **paradigma de interação** também determina como um usuário interage com o sistema. Um paradigma de interação indica a ordem em que os elementos envolvidos em uma operação são selecionados ou acionados pelo usuário. Este paradigma pode ser ação+objeto ou objeto+ação. No primeiro caso, o usuário tipicamente seleciona a operação a ser realizada, e em seguida o objeto sobre o qual deve atuar. No segundo caso, o usuário seleciona inicialmente o objeto, e em seguida a operação que deseja realizar sobre ele.

Linguagem Natural

Algumas aplicações permitem ao usuário se expressar em linguagem natural, ou seja, utilizando a língua com que ele se comunica com outros seres humanos, seja português, inglês, francês, ou outra qualquer.

A interação em linguagem natural é bastante atrativa para usuário com pouco ou nenhum conhecimento em computação. Entretanto, ela não se aplica a todos os tipos de sistemas. Sistemas de consulta a informações e sistemas baseados em conhecimento são exemplos onde a utilização de interfaces em linguagem natural é bastante interessante. No primeiro caso, por possibilitar que usuário não especialistas possam fazer consultas em sua própria língua. No segundo caso, para que o sistema gere explicações a partir da sua base de conhecimento, uma vez que a linguagem natural é expressiva o suficiente para a descrição do raciocínio artificial do programa, o que não seria possível com outros estilos de interação.

Uma aplicação que oferece interface em linguagem natural precisa lidar com construções vagas, ambíguas, e até gramaticalmente incorretas. Ainda não é possível desenvolver sistemas que compreendam qualquer expressão em linguagem natural, mas diversos tipos de sistemas especialistas utilizam com sucesso algum subconjunto de uma linguagem natural, nos quais o usuário deve se expressar de forma inequívoca e tendo em vista as frases que tais sistemas possam interpretar.

Para permitir que um usuário interaja com aplicações em linguagem natural, podemos fornecer uma interface textual onde ele deve digitar as frases que expressem seus comandos ou questionamentos. Outra alternativa são as interfaces orientadas por menus, através dos quais ele pode selecionar cada palavra ou expressão até compor a frase desejada.

Em uma aplicação em linguagem natural, tentamos aproximar a aplicação do usuário, de forma a privilegiar a forma de comunicação deste. Em outro extremo, tentamos aproximar o usuário do sistema computacional, utilizando linguagens artificiais bastante restritas, chamadas linguagens de comando.

Linguagem de Comando

A interfaces baseadas em linguagens de comandos proporcionam ao usuário a possibilidade de enviar instruções diretamente ao sistema através de comandos específicos [Preece et al., 1994]. Os comandos podem ser compostos por teclas de funções, por um único caractere, por abreviações curtas, palavras inteiras ou uma combinação de teclas e caracteres. Embora os comandos na forma de caracteres e teclas de função sejam disparados com um menor número de teclas digitadas,

estes comandos são mais difíceis de lembrar do que um nome ou abreviação bem escolhida.

As linguagens de comandos podem ser consideradas poderosas por oferecerem acesso direto à funcionalidade do sistema e por permitirem maior iniciativa do usuário e maior flexibilidade na construção dos comandos através da variação de parâmetros e combinação de palavras e sentenças. Contudo, este poder e flexibilidade implicam uma maior dificuldade dos iniciantes em aprender e utilizar o sistema. Os comandos e a sintaxe da linguagem precisam ser relembrados e erros de digitação são comuns mesmo nos mais experientes. A falta de padronização nos diversos sistemas é um fator importante na dificuldade de utilização deste estilo. Usuários especialistas, no entanto, conseguem maior controle do sistema e produtividade através de interfaces baseadas em linguagens de comandos.

Ao se projetar uma linguagem de comando, deve-se levar em conta a organização e estrutura dos comandos, assim como os nomes e abreviações utilizados. Os comandos podem ser simples ou compostos de parâmetros e/ou opções. Deve-se considerar cuidadosamente a ordenação de parâmetros, tentando refletir a estrutura da tarefa realizada. Quanto aos nomes e abreviações, deve-se dar preferência às palavras-chave em vez de símbolos aleatórios, e considerar cuidadosamente o nível de apresentação de cada comando, mantendo um equilíbrio entre especificidade e generalidade. Existem diversas estratégias de abreviação, como eliminação de variáveis, utilização das primeiras letras de uma palavra ou da primeira letra de cada palavra que descreve o comando, etc.

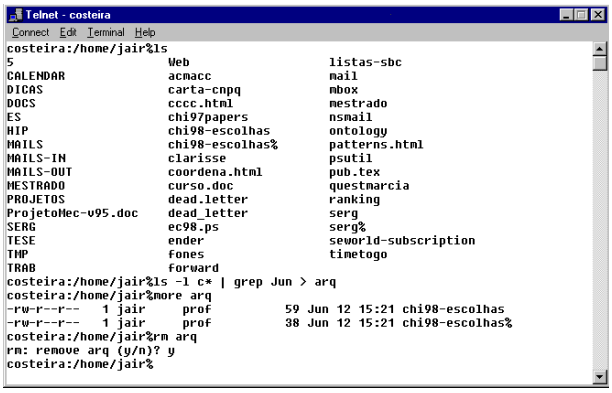


Figura 1.5 — Exemplo de interface no estilo de linguagem de comando.

Algumas interfaces baseadas em comandos podem ser auxiliadas por menus que indicam quais são os diversos comandos e como eles devem ser acionados. A ferramenta de gerenciamento do correio eletrônico PINE é um exemplo de interface baseada em comandos elementares auxiliados por menus. A Figura 1.6 ilustra uma tela do PINE.

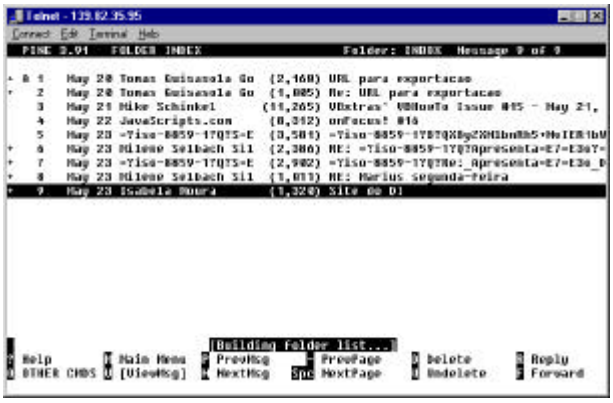


Figura 1.6 — Interface baseada em comandos auxiliada por menus.

Menus

Um menu é um conjunto de opções apresentadas na tela, no qual a seleção de uma ou mais opções resulta em uma mudança no estado da interface [Paap & Roske-Hofstrand, 1988]. Neste estilo de interação os usuários não precisam lembrar o item que desejam, apenas reconhecê-lo. Para que este estilo de interação seja eficiente, portanto, os itens de menu devem ser auto-explicativos.

Os menus podem ser de seleção simples ou múltipla, e podem ser utilizados para configurar um parâmetro ou disparar uma operação. Um menu de seleção simples pode tomar a forma de um grupo de botões de opção, ou *radio buttons*. Já um menu de seleção múltipla pode ser representado por um grupo de botões de seleção, ou *check boxes*. Quando o número de opções torna-se muito grande, temos a preferência por listas, de seleção simples ou múltipla.

A desvantagem da interação por menus é que estes ocupam muito espaço na tela. Existem diversas técnicas para se agrupar e apresentar as opções de menus. A mais comum é a categorização hierárquica das opções, na qual deve-se tomar cuidado para a seleção dos nomes dos grupos e de cada opção, para que reflitam as metas e tarefas do usuário. Um menu hierárquico pode ocorrer na forma de uma seqüência de telas, ou como um menu *pull-down*. Em um menu *pull-down*, o menu surge ao se clicar em seu título, e desaparece assim que se seleciona uma das opções.

Outra alternativa para poupar espaço de tela é oferecida por menus *pop-up*. Um menu pop-up aparece ao se clicar em um determinada área da tela ou elemento de interface, e pode permanecer visível até que o usuário selecione um de seus itens ou decida fechá-lo.

Preenchimento de Formulários

Interfaces no estilo de preenchimento de formulário são utilizadas principalmente para entrada de dados em sistemas de informação. Uma tela de preenchimento de formulário lembra um formulário em papel, apresentando campos que devem ser

preenchidos pelo usuário. O *layout* de um formulário com freqüência é semelhante a um formulário impresso que o usuário utilizava antes da implantação do sistema, facilitando seu aprendizado.

Este estilo de interação é útil principalmente quando diferentes categorias de informação devem ser fornecidas ao sistema, principalmente quando os mesmos tipos de dados devem ser digitados repetidamente [Preece et al., 1994], como em cadastros, controle de vendas e estoque, etc.

Estas interfaces são, em geral, fáceis de aprender. Para isto, devem deixar claro o tipo de dado que pode entrar em cada campo, facilitar a correção de erros de digitação e a verificação dos dados digitados através de técnicas como dígitos verificadores e totalização de valores. Os aspectos principais que vão influenciar na usabilidade do sistema são a produtividade do usuário, a sua satisfação e o esforço físico provocado pelo sistema, uma vez que estes sistemas são projetados para que os usuários forneçam um grande número de dados ao longo de um dia de trabalho.

Estas interfaces experimentaram uma popularização importante nas aplicações de Internet, através de formulários em HTML e scripts CGI.

WIMP (*Windows, Icons, Menus, and Pointers*)

O estilo de interação **WIMP**, um acrônimo em inglês para Janelas, Ícones, Menus e Apontadores, permite a interação através de componentes de interação virtuais denominados *widgets*. Este estilo é implementado com o auxílio das tecnologias de interfaces gráficas, que proporcionam o desenho de janelas e o controle de entrada através do teclado e do mouse em cada uma destas janelas. Os softwares de interfaces que implementam estes estilos permitem a construção de ícones que permite a interação através do mouse, comportando-se como dispositivos virtuais de interação.

WIMP não deve ser considerado um único estilo, mas a junção de uma tecnologia de hardware e software, associada aos conceitos de janelas e de *widgets* que permitem a implementação de vários estilos. Nas interfaces WIMP é possível encontrar os estilos de menus, manipulação direta, preenchimento de formulário e linguagem de comandos. WIMP pode ser considerado um estilo ou um *framework* de interface apoiado pela tecnologia de interfaces gráficas (GUI – *Graphical User Interfaces*). A Figura 1.7 apresenta uma tela de aplicação no estilo WIMP.

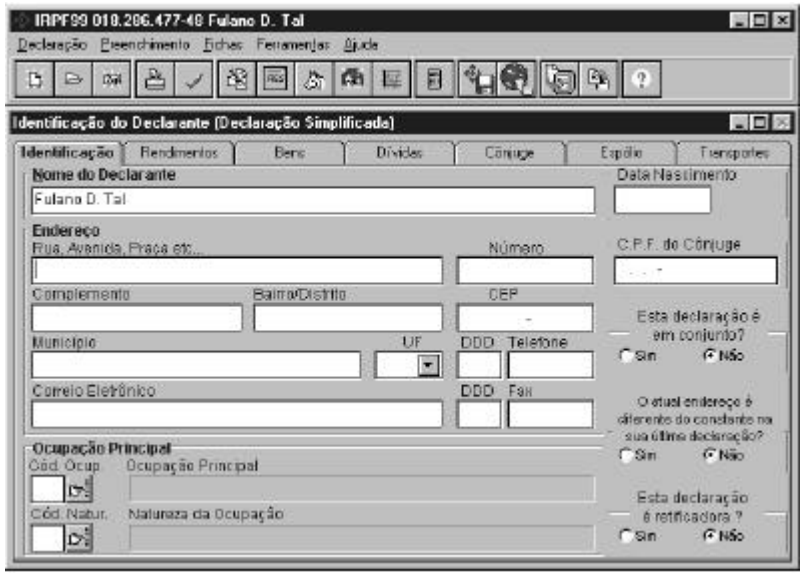


Figura 1.7 — Tela de uma aplicação no estilo WIMP.

Manipulação Direta

Interfaces de **manipulação direta** são aquelas que permitem ao usuário agir diretamente sobre os objetos da aplicação (dados ou representações de objetos do domínio) sem a necessidade de comandos de uma linguagem específica. Neste tipo de interface, os comandos são ações baseadas numa analogia entre o cursor e a mão, e as representações gráficas e os objetos do domínio. Na interação por linguagens de comandos o usuário interage indiretamente, através de comandos textuais e nomes que representem os objetos do sistema.

As interfaces gráficas que se utilizam da metáfora de *desktop*, como as do Apple Macintosh, baseada no Xerox Star, proporcionam um estilo no qual os usuários podem interagir com o gerenciador de arquivos do sistema operacional através de manipulação de ícones que representam arquivos, diretórios, discos e outros componentes computacionais. O usuário interage com ícones, utilizando o mouse ou outro dispositivo equivalente, através de ações do tipo clicar, arrastar (*drag-and-drop*), etc.

2. BASES TEÓRICAS

2.1 A Engenharia Cognitiva

As abordagens dominantes que têm caracterizado IHC são as de base cognitiva [Preece et al., 1994]. Elas têm raízes comuns com as áreas de psicologia cognitiva, ciência cognitiva e inteligência artificial que estudam a cognição, isto é, o processo pelo qual se pode adquirir conhecimento, e aplicam suas teorias na compreensão das capacidades e limitações da mente dos usuários. Os resultados delas são de longe mais numerosos do que os de qualquer outra abordagem.

A estratégia das abordagens cognitivas para o apoio ao *design* de sistemas interativos consiste na elaboração de modelos cognitivos genéricos que permitam aos *designers* entender os processos cognitivos humanos usados na interação e realizar experimentos ou previsões com estes modelos. A idéia básica é que modelos cognitivos que descrevem os processos e estruturas mentais (e.g. recordação, interpretação, planejamento e aprendizado) podem indicar para pesquisadores e projetistas quais as propriedades que os modelos de interação devem ter de maneira que a interação possa ser desempenhada mais facilmente pelos usuários. Como estas abordagens adotam uma perspectiva centrada nos aspectos cognitivos do usuário, o *design* feito com base nelas é chamado de *design* de sistemas centrado no usuário (*User Centered System Design – UCSD*).

Uma das teorias mais conhecidas de *design* centrado no usuário é a Engenharia Cognitiva [Norman, 1986]. Norman considera que o *designer* inicialmente cria o seu modelo mental do sistema, chamado modelo de *design*, com base nos modelos de usuário e tarefa (ver seções 3.2 e 3.3). O modelo implementado deste modelo de *design* é a imagem do sistema. O usuário então interage com esta imagem do sistema e cria seu modelo mental da aplicação, chamado de modelo do usuário. Este modelo mental é que permite ao usuário formular suas intenções e objetivos em termos de comandos e funções do sistema. A Figura 2.1 mostra o processo de *design* na abordagem da Engenharia Cognitiva.

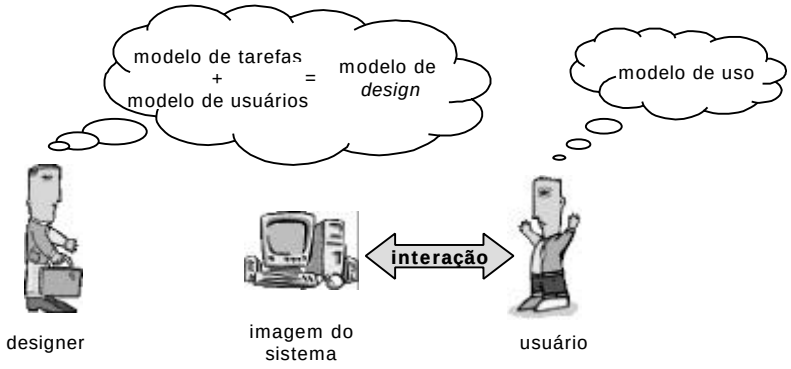


Figura 2.1 — Modelo de interação da Engenharia Cognitiva.

Assim, segundo a Engenharia Cognitiva a meta do *designer* é em desenvolver um sistema que permita ao usuário, durante o processo de interação, criar um modelo mental consistente com o modelo projetado pelo *designer*. Para que isto seja possível, Norman argumenta que o *designer* precisa entender o processo através do qual o usuário interage com a interface do sistema e propõe a teoria da ação.

A **teoria da ação** define que a interação usuário-sistema é desempenhada num ciclo-de-ação com sete etapas e dois "golfos" a serem atravessados. Um deles é **o golfo da execução** e envolve as etapas de formulação da meta, especificação da seqüência de ações e atividade física de execução. O outro é o **golfo da avaliação** e deve ser atravessado pelas etapas de percepção, interpretação e avaliação da meta (ver Figura 2.2).

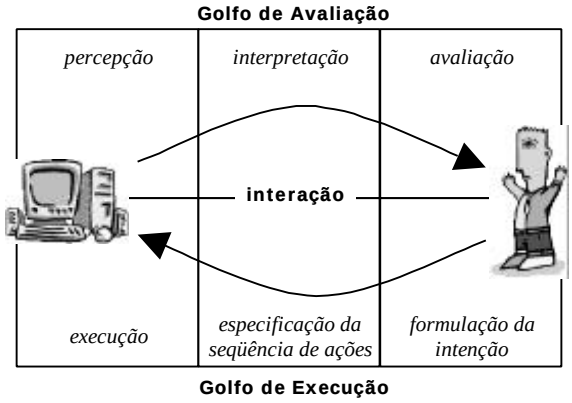


Figura 2.2 — Etapas de ação do usuário durante a interação com o sistema.

O usuário utiliza o sistema com o objetivo de realizar uma determinada tarefa. Para isto, ele deve formular metas a serem alcançadas através da interação com as funções disponíveis no sistema. Em seguida, o usuário deve definir quais são as ações a serem executadas para que ele consiga atingir a sua meta. Note-se que, até este ponto, o usuário realizou a preparação mental para a execução da meta. Resta-lhe concretizar o que foi mentalizado através de uma ação física. Estas três fases compreendem a travessia do golfo de execução, e não precisam ser necessariamente realizadas na seqüência descrita. Por exemplo, a especificação e o planejamento podem ser realizados intercaladamente ou pode-se começar a executar o comando sem que se tenha ainda especificado todas as ações por completo.

Assim que o sistema executa a ação definida pelo usuário inicia-se o golfo de avaliação. A primeira etapa da travessia deste golfo é a percepção do usuário do novo estado em que o sistema se encontra. O usuário então interpreta este novo estado e o avalia de acordo com a sua meta inicial. Com base nesta avaliação o usuário prossegue para definir sua próxima ação. É importante notar que, se o usuário não perceber que o sistema mudou de estado através de uma sinalização

clara, ele possivelmente interpretará que nada ocorreu e que a sua meta inicial não foi atingida.

Exemplo — Etapas da interação usuário–sistema

Em um sistema de biblioteca, um usuário que queira fazer uma consulta sobre um livro ou artigo poderia passar pelas seguintes etapas de interação, de acordo com a abordagem centrada no usuário:

formulação da intenção:	Quero procurar a referência completa do livro "Human-Computer Interaction", editado por Preece.
especificação da seqüência de ações:	Devo selecionar o comando de "busca" e entrar com os dados que eu tenho.
execução:	Ativo "busca" no menu; digito o nome do livro no campo "nome do livro"; digito o nome do autor no campo "nome do autor"; seleciono "OK"
percepção:	Apareceu uma nova tela com dados de livro.
interpretação:	Os dados apresentados correspondem à busca que eu fiz.
avaliação:	Encontrei as informações que eu queria. Completei a tarefa com sucesso.

O *designer* pode ajudar o usuário a atravessar estes golfos diminuindo-os. Para isto ele deve definir quais são as ações e estruturas mais adequadas para comandar as funções do sistema, escolher os elementos de interface que melhor comunicam a informação desejada, optar por *feedbacks* significativos, dentre outras escolhas de *design*. Assim, quanto mais próxima da tarefa e das necessidades do usuário for a linguagem de interface oferecida pelo *designer*, menos esforço cognitivo o usuário terá que fazer para atingir seus objetivos.

A Figura 2.1 mostra que, em Engenharia Cognitiva, o processo de *design* se inicia com o modelo mental que o *designer* cria do sistema. No entanto, a Engenharia Cognitiva focaliza centralmente a interação usuário-sistema, enfatizando o produto final do processo de *design*, o sistema, e o modo como o usuário o entende. A seguir, apresentamos a Engenharia Semiótica que complementa a Engenharia Cognitiva, à medida que focaliza centralmente o *designer* e o processo de *design*.

Exercícios

1. Imagine que você é o usuário do sistema de biblioteca do exemplo anterior e deseja imprimir a referência encontrada, e que na interface existe um botão "imprime". Descreva cada um dos passos que você tomaria para atravessar o golfo de execução.

Observação: A travessia do golfo de avaliação, neste caso, envolve dispositivos periféricos, papel, submetas, etc.

2. Imagine que você é o usuário do sistema de biblioteca e precisa fazer uma consulta. Para isto, você seguiu os três primeiros passos descritos no exemplo anterior para atravessar o golfo de execução. Para cada uma das respostas do sistema apresentadas abaixo, descreva seus passos para atravessar o golfo de avaliação (percepção, interpretação, avaliação):

- sistema não forneceu feedback
- sistema emitiu um som de bip
- sistema voltou para a tela inicial

2.2 A Engenharia Semiótica

As abordagens semióticas têm como base teórica a semiótica, disciplina que estuda os signos, os sistemas semióticos e de comunicação, bem como os processos envolvidos na produção e interpretação de signos. Um signo é algo que representa alguma coisa para alguém [Peirce, 1931]. Por exemplo, tanto a palavra <cão> em português, quanto uma fotografia de um cão representam o animal cachorro, e assim são signos de cachorro para falantes da língua portuguesa. Nestas abordagens toda aplicação computacional é concebida como um ato de comunicação que inclui o *designer* no papel de emissor de uma mensagem para os usuários dos sistemas por ele criados, [Nadin, 1988; Andersen et al., 1993; de Souza, 1993; Jorna & Van Heusden, 1996].

Para que a comunicação entre duas pessoas aconteça, é preciso que o emissor da mensagem a expresse em um código que tanto ele, quanto o receptor conheçam. Cada mensagem pode ser formada por um ou mais signos. Assim que o receptor recebe a mensagem, ele gera uma idéia daquilo que o emissor quis dizer e inicia o seu processo de compreensão [Jakobson, 1970]. Esta idéia que ele gera é chamada de interpretante, e pode, ele mesmo, gerar novos interprentantes na mente do receptor, numa cadeia indefinida de associações. A este processo se dá o nome de semiose ilimitada [Eco, 1976] e ele acontece até que ou o receptor acredite que ele tenha uma boa hipótese do que o emissor quis dizer, ou ele conclua que não é capaz de, ou não está disposto a, criar tal hipótese. Neste caso, ele pode ou não dar continuidade ao processo de comunicação, passando então para o papel de emissor. A Figura 2.3 ilustra este processo de comunicação.

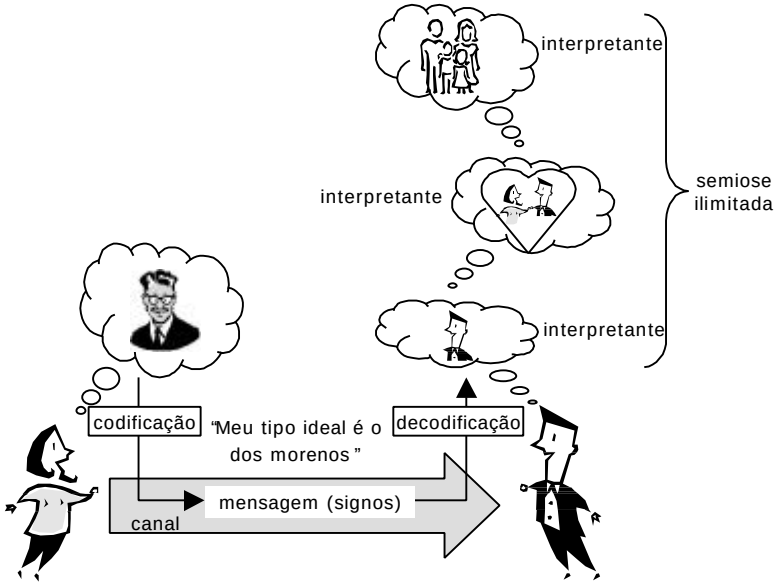


Figura 2.3 — Processo de comunicação entre duas pessoas.

Na Engenharia Semiótica [de Souza, 1993; de Souza, 1996] em particular a interface de um sistema é vista como sendo uma mensagem sendo enviada pelo *designer* ao usuário. Esta mensagem tem como objetivo comunicar ao usuário a resposta a duas perguntas fundamentais: (1) Qual a interpretação do *designer* sobre o(s) problema(s) do usuário?, e (2) Como o usuário pode interagir com a aplicação para resolver este(s) problema(s)? O usuário concebe a resposta a estas perguntas à medida que interage com a aplicação. Assim, esta mensagem é unilateral, uma vez que o usuário recebe a mensagem concluída e não pode dar continuidade ao processo de comunicação [de Souza, 1993] naquele mesmo contexto de interação. Além disso, como esta mensagem (a interface) é ela mesma capaz de trocar mensagens com o usuário, ela é um artefato de comunicação sobre comunicação, ou meta-comunicação.

A Figura 2.4 mostra este processo de comunicação entre *designer* e usuário. Existem dois pontos que devem ser ressaltados nesta figura. Primeiramente, note-se que a interação usuário-sistema é parte da meta-mensagem do *designer* para o usuário, uma vez que é a partir desta meta-mensagem que o usuário aprenderá a interagir com o sistema. Além disso, para que a comunicação entre o *designer* e o usuário tenha sucesso, o modelo conceitual da aplicação pretendido pelo *designer* e o modelo da aplicação percebido pelo usuário, embora diferentes, devem ser consistentes entre si.

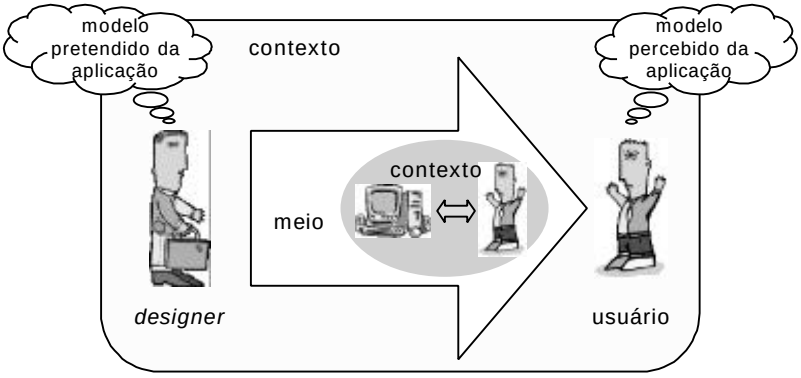
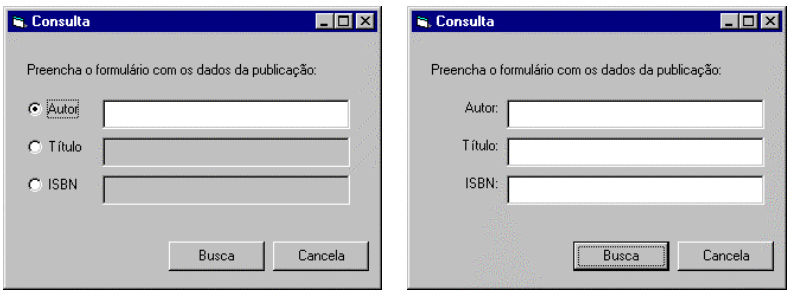


Figura 2.4 — Ato de comunicação entre designer e usuário, na Engenharia Semiótica.

Na abordagem da Engenharia Semiótica, o *designer* é autor de uma mensagem ao usuário, que é transmitida pela interação que caracteriza o processo metacomunicativo. Assim, o *design* de interfaces envolve não apenas a concepção do modelo da aplicação, mas a comunicação deste de maneira a revelar para o usuário o espectro de usabilidade da aplicação. A Engenharia Semiótica ressalta ainda que a presença do *designer* no cenário comunicativo deve ser explicitada e tornada sensível para os usuários para que eles tenham maior chance de entender as decisões de *design* tomadas e a aplicação com que estão interagindo, sendo assim capazes de fazer um uso mais criativo e eficiente desta aplicação.

A Figura 2.5 apresenta duas telas de consulta de uma aplicação. Observe que a primeira tela (*2.5a*) comunica claramente a restrição da busca a apenas um campo, enquanto na segunda (*2.5b*) se permite realizar a busca por um ou mais campos.



(a) *(b)*

Figura 2.5 — Exemplos de diferentes mensagens para uma tarefa de consulta.

Exercício

Imagine que você é o *designer* do sistema de biblioteca e deseja projetar a interface para que o usuário faça uma consulta a um livro ou artigo.

1. Que informações você considera importantes para esta tarefa?
2. Que mensagem você pretende passar ao usuário?
3. Como você organizaria a tela para passar esta mensagem?

2.3 Engenharia Semiótica *x* Engenharia Cognitiva

Tanto a Engenharia Semiótica quanto a Engenharia Cognitiva vêem o processo de *design* se iniciando com o *designer* que cria o seu modelo mental da aplicação, e com base neste, implementa a própria aplicação. O usuário interage com esta aplicação e através dela cria o seu próprio modelo mental da aplicação. A criação da aplicação pelo *designer* e a interação do usuário são assíncronas, ou seja, se dão em diferentes momentos no tempo.

A Engenharia Cognitiva se concentra na segunda etapa deste processo de *design*, ou seja, na interação usuário-sistema, deixando a etapa *designer*-sistema em segundo plano. Assim, ela enfatiza o produto deste processo, que é o sistema, e a interpretação do usuário deste produto. Em outras palavras, a Engenharia Cognitiva dá subsídios para se definir a **meta ideal** do processo de design, um **produto**, cognitivamente adequado para a população de usuários.

A Engenharia Semiótica por sua vez, junta estas duas etapas ao transferir seu ponto de vista para um nível mais abstrato, no qual o *designer* envia ao usuário uma meta-mensagem. Desta forma, a Engenharia Semiótica dá um *zoom-out* no processo de *design* e inclui a Engenharia Cognitiva. Assim, todos os resultados obtidos na Engenharia Cognitiva continuam sendo válidas na Engenharia Semiótica. No entanto, a interação usuário-sistema deixa de ser o foco da Engenharia Semiótica, dando lugar para a expressão do *designer* e o processo de *design* como um todo. A Figura 2.6 mostra a relação entre as Engenharias Semiótica e Cognitiva. Em outras palavras, a Engenharia Semiótica dá subsídios para se definir o **plano de design**, um **processo** semioticamente coeso e consistente, levando com segurança a mensagem do produtor (designer) ao consumidor (usuário).

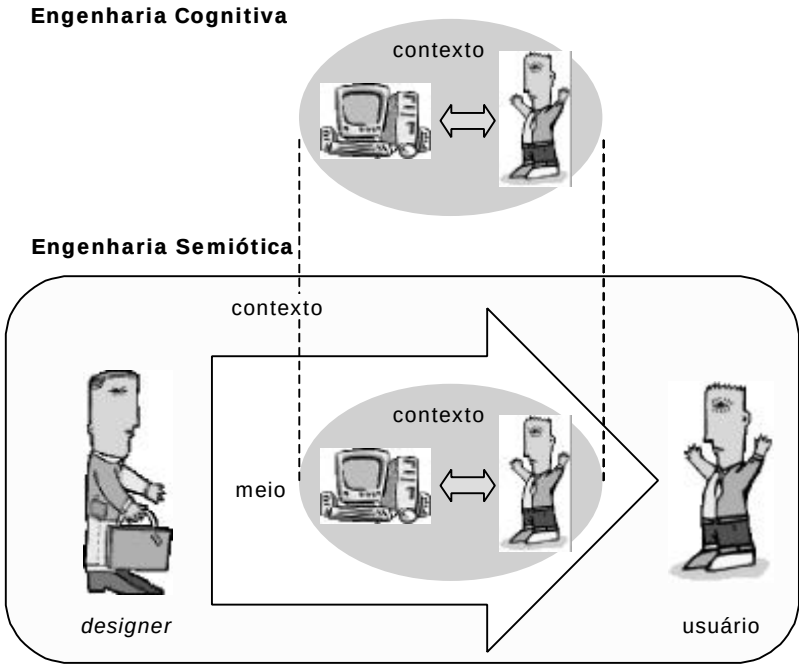


Figura 2.6 — Relação entre Engenharia Cognitiva e Engenharia Semiótica.

A conseqüência de a Engenharia Cognitiva focalizar na interação usuário-sistema é que ela dá margem para que se passe a idéia de que existe uma solução ideal para o problema do usuário. Quando isto acontece, o *designer* não deixa claro para o usuário que a solução sendo oferecida é uma dentre várias e que esta solução é determinada por suas [do designer] interpretações e decisões de *design*. O usuário por sua vez não percebe que a aplicação é a criação de uma outra pessoa e que pode conter interpretações não ideais, ou até mesmo errôneas, e muitas vezes trata a aplicação como se ela fosse infalível ou, ao contrário, como se ela é que estivesse enganada sobre o domínio ou o usuário, e não o designer que a concebeu.

Ao trazer o *designer* para dentro do foco, a Engenharia Semiótica evidencia a sua presença e permite ao usuário entender que todo sistema é uma solução potencial de um *designer* (ou de uma equipe de *design*). Assim, o usuário, ao ter problemas de interação com a aplicação, pode tentar entender o que o *designer* pretendia, e acertar o seu modelo mental da aplicação, aproximando-o cada vez mais daquele do *designer*. Fazendo isto, o usuário é capaz de alcançar um melhor entendimento das motivações e decisões tomadas pelo *designer*, e assim usar a aplicação de forma mais eficiente.

3. MODELOS E TÉCNICAS DE MODELAGEM EM IHC

3.1 Um modelo para o processo de *design* de interfaces

Design é a atividade intelectual de conceber e descrever um produto a partir dos requisitos de seus potenciais usuários. Esta atividade requer técnicas e ferramentas adequadas, aliadas à criatividade, ao talento e à experiência do *designer*.

O produto concebido em uma atividade de *design* precisa ser apresentado através de um protótipo e/ou de uma especificação. A prototipação consiste na descrição do que foi concebido, utilizando materiais mais baratos e dimensões reduzidas. O objetivo é poder fazer uma avaliação. A especificação consiste na descrição abstrata, rigorosa, idealmente correta e completa do produto, utilizando uma notação ou linguagem adequadas.

A especificação e a prototipação permitem visões e formas de avaliação complementares do produto concebido. A especificação permite uma descrição e avaliação a partir de técnicas associadas às notações utilizadas. Já a prototipação permite uma descrição e avaliação mais concreta do produto no contexto de utilização.

O *design* de interfaces de usuário é uma atividade que requer análise dos requisitos dos usuários, concepção, especificação e prototipação da interface, e avaliação da utilização do protótipo pelos usuários. Diversos modelos do desenvolvimento de software baseados em prototipação argumentam que este processo deve ser realizado de forma cíclica, isto é, a avaliação deve levar a um novo *design* e ser posteriomente avaliado.

Na abordagem da Engenharia Semiótica, a mensagem do *designer* contempla tanto a funcionalidade quanto o modelo de interação. A interface é, portanto, responsável por fazer o usuário ter condições de interagir com a funcionalidade do sistema. O *design* da interface de usuário depende da especificação dos modelos de interação e da funcionalidade do sistema.

A especificação da funcionalidade visa descrever quais funções o sistema deve oferecer para os usuários. Por funções entenda-se aquilo que permite ao usuário atingir as suas metas, independente de como é implementado. Elas são chamadas aqui de **funções do domínio** numa referência àquelas funções do sistema aplicadas ao domínio.

A especificação do modelo de interação visa descrever de forma abstrata e precisa como o usuário pode interagir com o sistema independente de quais dispositivos de interação ou widgets ele vai utilizar e de como este processo de interação será implementado pelo software da interface. O objetivo é descrever que o usuário pode comandar uma função do sistema, que ele deve fornecer uma informação específica ou que ele deve ler uma mensagem na tela. A partir desta descrição o *designer* da interface deve escolher qual o modelo de interação específico (estilo ou padrão) o usuário irá utilizar para interagir. A descrição abstrata oferece a vantagem de ser independente de estilo e ferramenta de interface.

No desenvolvimento de sistemas centrado-no-usuário a especificação da funcionalidade e a do modelo de interação são derivadas do modelo de tarefas (considerando também os diferentes perfis de usuários) e são a base para o restante do desenvolvimento. A partir destas especificações o *designer* da interface deve realizar a especificação detalhada da interface de acordo com o modelo de interação, bem como a construção de um protótipo de interface. Num processo de *design* baseado em prototipação apenas este último é realizado.

O protótipo da interface deverá permitir uma avaliação da interação com os usuários do sistema. Esta avaliação pode ser feita utilizando-se dois tipos de testes básicos: testes de usabilidade e testes de comunicabilidade. Os testes de usabilidade visam avaliar fatores relacionados com facilidade de uso, produtividade, flexibilidade e satisfação do usuário. Os testes de comunicabilidade visam avaliar as decisões do *designer* em termos de escolhas de signos de interface para comunicar melhor o modelo conceitual da aplicação.

Resumindo, o processo de *design* de interfaces inicia-se com a análise de usuários e tarefas (que constitui a análise de requisitos) e deve ser conduzido num processo cíclico ou iterativo no qual cada passo apresenta evoluções a partir da etapa anterior. Cada ciclo envolve a **especificação** da funcionalidade e do modelo de interação, a **prototipação** de interfaces (que possibilite a interação de acordo como o modelo especificado) e a sua **avaliação** junto aos usuários. A partir desta avaliação um novo ciclo de especificação, prototipação e avaliação deve ser realizado. Este processo pode ser visualizado na Figura 3.1.

É importante ressaltar que o *design* da interface pode (e deve) ser conduzido independentemente da implentação do software da interface.

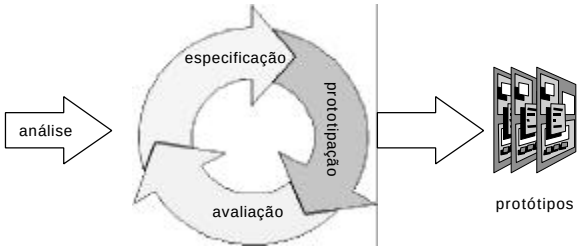


Figura 3.1 — Processo de design de interfaces.

3.2 Análise e Modelagem de Usuários

O usuário deve sempre ser o foco central de interesse do projetista ao longo do *design* da interface. O objetivo da análise e modelagem de usuários é identificar quem são os usuários e caracterizá-los, isto é, especificar quais funções exercem, quais capacidades possuem, etc. Podemos destacar os seguintes fatores de análise de usuários:

- **Papel** (ou função) específico de cada usuário: definido pelas tarefas que ele realiza. Numa organização, as pessoas trabalham juntas, de maneira estruturada. A estrutura da organização define relacionamentos entre as

pessoas. A implicação imediata dos diferentes papéis de cada usuário são as diferentes tarefas que eles irão realizar. Algumas tarefas podem ser comuns a diferentes papéis de usuários, enquanto outras podem ser exclusivas de papéis específicos.

- **familiaridade com computadores**: o grau de conhecimento e capacidade medido pela proficiência no uso de dispositivos de entrada (p.ex., teclado, mouse) e pela familiaridade com o modelo de interação do ambiente de interface. Esta capacidade pode ser adquirida pela experiência prévia com outras interfaces semelhantes, ou pela freqüência de uso. Os valores extremos desta medida são **iniciante** e **experiente**. Um usuário iniciante costuma cometer erros e precisa de auxílio e apoio extensivo ao aprendizado (e.g. help on-line e tutorial). Um usuário experiente não precisa de suporte extensivo, e prefere uma interação mais rápida, que ofereça *shortcuts* (atalhos), para aumentar seu desempenho. Usuários iniciantes podem se tornar experientes através do uso freqüente da aplicação.

- **nível de conhecimento do domínio da aplicação**: um usuário que não tem conhecimento prévio do domínio da aplicação é considerado **novato**, enquanto um usuário que conhece não apenas o domínio, mas também diferentes maneiras de realizar as tarefas é considerado **especialista**. A interface projetada para um usuário novato deve proporcionar informações sobre o estado da aplicação para guiar o usuário e oferecer dicas claras para recuperação de erros. Para um especialista, ela deve oferecer sofisticados recursos que lhe permitam estender a aplicação.

- **freqüência de uso da aplicação**: um usuário **ocasional** não melhora suas capacidades com o computador ou com a aplicação ao longo do tempo, e pode até diminuí-las. Em contrapartida, a capacidade de um usuário **freqüente** aumenta e suas necessidades de ajuda e apoio ao aprendizado diminuem ao longo do tempo, enquanto a necessidade de interação mais sofisticada aumenta.

- **contexto sócio-cultural**: há um conjunto de fatores que visam analisar as diferenças sócio-culturais entre os usuários da aplicação, tais como problemas de línguas ou dialetos diferentes e tradições culturais distintas. Estes fatores têm grande importância em aplicações que serão distribuídas em diversos países, ou em diversas regiões de um mesmo país (é o caso de software de uso geral e software de companhias multinacionais).

Procedimento para condução da análise de usuários

A análise de usuários pode ser dividida em cinco etapas [Lee 1993]:

1. **Identificar fatores de análise críticos centrais para a aplicação**. Nesta etapa, identificamos quais dos fatores citados acima têm importância para a aplicação em questão. Por exemplo, em uma aplicação mono-usuário, a função do usuário não é um fator crítico para análise. Além disto, uma aplicação situada em um ambiente social comum também não apresenta diferenças sócio-culturais.

2. **Explorar outros fatores críticos adicionais para a aplicação**. Além dos fatores básicos citados acima, pode-se considerar outros, como a familiaridade com algum estilo de interface gráfica (Windows, Macintosh, Motif, etc.), por exemplo. Estes fatores podem ajudar a determinar características do projeto e da aplicação que ainda não estejam especificadas, como plataforma, estilo de interação preferido, requisitos não-funcionais como eficiência e custo, etc. Ao identificar novos fatores críticos para a modelagem de usuário, é importante definir também quais suas implicações sobre o design da interface, ou seja, que aspectos da interface serão influenciados ou determinados para cada valor definido para estes fatores.

3. **Estimar a distribuição de usuários para cada fator**. Após identificar os fatores críticos, deve-se fazer um levantamento estimado do percentual de usuários em cada grupo de fator crítico. Por exemplo, pode-se estimar que 10% dos usuários sejam iniciantes, e 90% deles experientes com computadores. Mais que isto, é aconselhável ter em conta que nenhum usuário permanece novato para sempre, e que a aplicação deve tratar do ciclo de aprendizado inteiro.

4. **Identificar grupos majoritários de usuários**. Para cada fator é preciso determinar qual a categoria dominante. Desta forma é possível ter um perfil do usuário majoritário.

5. **Analisar a implicação coletiva da distribuição de usuários**. A distribuição não deve ser um fator absoluto e alguns aspectos subjetivos podem ser considerados. Como vimos, deve-se considerar que usuários iniciantes vão adquirir experiência, e que isto pode mudar o modelo final de usuários. Por exemplo, em uma aplicação onde a maioria seja experiente no uso de computadores e no estilo de interação escolhido, deve-se priorizar o acesso rápido às funções, e não ao suporte extensivo ao aprendizado, com tutoriais detalhados. Contudo, é preciso dar condições para os novatos se tornarem proficientes, com módulos ou configurações especiais a eles dedicadas.

Como resultado da análise e modelagem de usuário, temos a indicação dos requisitos mais importantes que devem estar presentes na interface a ser projetada. A partir desta indicação, os projetistas devem poder tomar decisões acertadas sobre as características da interface.

Exemplo — Questionário para Análise de Usuários

Para um sistema de bibliotecas, um questionário para análise dos usuários da biblioteca poderia conter os seguintes fatores:

- nível de familiaridade com computador: iniciante, intermediário, ou experiente
- nível de familiaridade/especialização no domínio da aplicação (conhecimento sobre como realizar as mesmas tarefas sem o auxílio do computador): novato, intermediário, ou especialista
- freqüência de uso: ocasional (menos de duas vezes por mês) ou freqüente
- ambiente gráfico preferido: Windows 95, Macintosh, ou indiferente

Exercício

Para o mesmo sistema de bibliotecas, identifique os fatores críticos para análise dos funcionários da biblioteca, desde funcionários que cadastram os livros até os funcionários que atuam no empréstimo dos mesmos.

3.3 Análise e Modelagem de Tarefas

O objetivo da análise de tarefas é fornecer ao designer a visão dos usuários das tarefas que eles precisam realizar. A modelagem de tarefas consiste em formalizá-las de forma a mapeá-las na interface gráfica.

Cenários

Cenários são narrativas textuais, pictóricas ou encenadas, de situações fictícias mas plausíveis (senão desejáveis) de **uso situado** da aplicação. Devem ser ricos em contextualização e possuir um **foco** claro que transmita a usuários e designers as idéias sendo testadas.

O levantamento de requisitos sobre as tarefas e os usuários pode ser melhor realizado quando os analistas procuram descrever situações do processo de trabalho. Os métodos baseados em cenários consistem de uma coleção de narrativas de situações no domínio do problema que permitem a identificação de componentes de design. Assim, eles são um meio de representação de fácil compreensão para os usuários envolvidos (muitas vezes de formação bastante heterogênea) que passam a poder avaliar, criticar e fazer sugestões. Eles permitem a reorganização de componentes e tarefas do domínio. Isto ocorre porque quando se introduz novas tecnologias, parte do contexto de tarefa de uma organização é alterado. Nesta reengenharia, novas tarefas e práticas são incorporadas enquanto outras são eliminadas.

Exemplo — Cenário

Para o nosso sistema de bibliotecas, um cenário de uso típico seria:

"Um aluno chega na biblioteca para procurar livros-texto dos cursos que está freqüentando. Ele entra no sistema e seleciona os cursos, e obtém uma lista de todos os livros-texto e sua localização na biblioteca. Seleciona a opção de 'bibliografia complementar', e uma nova lista de livros e artigos lhe é apresentada. Ele então manda imprimir todas as referências encontradas."

Exercício

Elabore um cenário de uso para situações de consulta em uma biblioteca. Em seguida, entreviste alguns usuários de bibliotecas para elaborar cenários para a mesma situação. Note as diferenças entre o ponto de vista do designer (o seu) e do usuário (seus entrevistados).

Repita este exercício para outras situações, como por exemplo, para tarefas dos funcionários de uma biblioteca.

Os cenários também podem ser utilizados para formar uma base de casos [Carey & Rusli, 1995], que pode ser utilizada tanto no aprendizado de IHC quanto no reuso de *designs*. Neste caso, ao invés de se utilizar cenários para tentar prever possíveis situações de uso, os cenários são registros de situações reais, que podem ser comparadas em diversos níveis, como em desempenho e eficiência de determinadas escolhas e decisões de *design.*

Análise de Tarefas Utilizando Cenários: Técnica de Questionamento Sistemático

A descrição de informações do domínio através de narrativas só é efetivamente realizada se o processo de compreensão por parte dos analistas e projetista for realizado de maneira sistemática [Carroll et al., 1994]. A técnica apresentada aqui pode ser utilizada, segundo os autores, tanto para métodos de análise e design orientado a objetos quanto para interação usuário-sistema.

O *questionamento sistemático* é uma técnica de psico-lingüística que permite a psicólogos e lingüistas examinar o conteúdo e a estrutura de informações contidas numa narrativa. Uma narrativa é um sumário de um conjunto de eventos e ações envolvendo agentes e objetos do mundo. Nem todas as informações integrantes do contexto são passadas através da narrativa. Muitas destas informações são inferidas do conhecimento cultural de cada indivíduo. Outras, entretanto, precisam ser obtidas diretamente na fonte, isto é, junto ao autor da narrativa.

Essa técnica foi desenvolvida para se entender melhor o processo de compreensão de estórias em narrativas. O objetivo é compreender tudo o que envolve o

contexto daquilo que está sendo passado na narrativa. Assim, analistas e projetista podem utilizar esta técnica para adquirir mais eficazmente conhecimento sobre o domínio e inferir objetos e interações que não estão descritos na narrativa.

São realizadas diversas questões sobre cada componente da narrativa. As respostas a estas questões servem de ponte entre uma idéia e outra, e revelam novas conexões cruciais para o entendimento preciso da narrativa.

A técnica de questionamento sistemático pode ser decomposta nos seguintes passos:

1. **Geração do cenário**. As narrativas que compõem o cenário devem ser escritas pelo especialista no domínio. O analista pode motivá-lo fazendo perguntas como num processo convencional de entrevista. Um exemplo de cenário seria: "Eu quero sacar R$100,00. Eu insiro o cartão do banco no caixa eletrônico, pressiono o botão de saque rápido, digito minha senha, retiro o dinheiro e o cartão".

2. **Elaboração da rede de proposições**. As narrativas devem ser simplificadas e expressas através de proposições, como por exemplo, "cliente insere cartão no caixa eletrônico"; "cliente pressiona botão de saque rápido"; "cliente digita senha"; "cliente pega dinheiro"; "cliente retira cartão".

3. **Análise**. A partir das proposições, pode-se determinar o perfil das tarefas (ações e objetos) e dos usuários (agentes das ações).

4. **Questionamento sistemático**. Novas proposições podem ser elaboradas através de questões que são feitas sobre elementos das proposições anteriores, num processo iterativo. Existem diversos tipos de questões, dentre as quais destacamos:

    - **Por que…?** Estas questões visam revelar conseqüências e causas a respeito de eventos da narrativa.

    - **Como…?** Estas questões oferecem maiores detalhes a respeito de determinados eventos, permitindo determinar sub-tarefas e maiores detalhes sobre a interação.

    - **O que é …?** Questões deste tipo podem ser feitas sobre objetos, e revelam atributos e hieraquias de objetos.

    - **Então "isto" é/ocorre "assim"?** Perguntas de verificação podem ser feitas para se saber se as proposições estão sendo entendidas corretamente. Estas perguntas têm resposta *sim* ou *não*. Uma taxonomia mais completa ainda está sendo pesquisada pelos autores do método.

Exercício

Para o cenário apresentado no exemplo anterior, elabore a rede de proposições, faça uma análise e levante as questões sobre os elementos destas proposições.

Modelos de Tarefas

Existem diversos modelos de tarefas, cada qual com notação e objetivos específicos, dentre os quais podemos destacar:

- **TAG (*Task-Action Grammar*)** [Payne & Green, 1989]: gramática gerativa, onde são especificadas ações a partir das tarefas. A técnica consiste em identificar tarefas simples, representá-las de forma categorizada e através de regras de produção. Esta notação permite tratar diversos tipos de consistência, a nível lexical, sintático e semântico. Permite ainda verificar a completeza do modelo
- **UAN (User Action Notation)** [Hix & Hartson, 1993]: utilizado principalmente para interfaces de manipulação direta, onde representa aspectos do comportamento do sistema do ponto de vista do usuário, ou seja, que tarefas e ações o usuário realiza na interface. Este modelo associa as ações do usuário ao *feedback* do sistema, ao estado do sistema e ao modelo computacional da aplicação.
- **GOMS (*Goals, Operators, Methods, and Selection Rules*)** [Card et al., 1983]: este modelo pretende representar o comportamento dinâmico da interação com o computador, com base num modelo do comportamento humano que possui três subsistemas de interação: o **perceptual** (auditivo e visual), o **motor** (movimentos braço-mão-dedo e cabeça-olho), e **cognitivo** (tomadas de decisão e acesso a memória). Este comportamento é definido por:
    - **metas** (*goals*), que podem ser decompostas numa hierarquia de sub-metas;
    - **operadores** (*operators*), que são os atos perceptivos, motores ou cognitivos básicos que os usuários devem executar para afetar o ambiente da tarefa;
    - **métodos** (*methods*), que são sequências de passos para se atingir uma meta; e
    - **regras de seleção** (*selection rules*), expressões do tipo "condição–ação", que devem ser utilizadas para selecionar um determinado método para atingir uma meta, sempre que houver mais de um método disponível para tanto.
- **Modelo Keystroke-Level** [Card et al., 1983]: este modelo faz parte da família GOMS de modelos, mas em um nível mais baixo, o nível de atividade motora. O objetivo aqui é prever o tempo que o usuário leva para realizar uma tarefa. São considerados os operadores primitivos, como uma tecla digitada, um botão clicado, os atos de apontar, de levar a mão até um dispositivo, desenhar, realizar uma operação mental e esperar a resposta do sistema. Os tempos de execução de uma tarefa são calculados sobre métodos, que são seqüências de operadores.

Abordaremos a análise de tarefas através de um modelo GOMS simplificado [Lee, 1993]. O GOMS permite que se construa modelos de tarefas bem mais complexos

e detalhados do que o necessário numa tarefa de análise para a construção de interfaces. Usaremos uma versão simplificada do GOMS, pois:

- o modelo da tarefa não deverá descrever informações de *design* da interface, uma vez que ela ainda não foi construída;
- o analista não é um especialista em psicologia cognitiva;
- o modelo simplificado pode ser expandido para o original, caso seja necessário.

Procedimento para condução da análise de tarefas

1. **Faça a análise *top-down.*** Comece pelas metas mais gerais, e vá acrescentando detalhes em direção às mais específicas.
2. **Use termos gerais para descrever metas**. Não use termos específicos de interfaces.
3. **Examine todas as metas antes de subdividi-las**. Isto facilita o reuso de metas.
4. **Considere todos os cenários de tarefas**. Utilize regras de seleção para representar alternativas.
5. **Use sentenças simples para especificar as metas**. Estruturas complexas indicam a necessidade de decompor uma meta em sub-metas.
6. **Retire os passos de um método que sejam operadores**. Os operadores são dependentes da interface, e não são tratados no modelo GOMS simplificado.
7. **Pare a decomposição no limite do design da interface**. A modelagem GOMS deve terminar quando as descrições estiverem tão detalhadas que os métodos sejam operadores ou envolvam pressuposições de *design*.

Para aplicações com múltiplas funções de usuários, temos algumas orientações específicas:

- Inicie especificando metas de alto nível para cada função de usuário.
- Se uma meta for compartilhada por mais de uma função de usuário, identifique estas funções de usuário ao definir a meta. Isto torna-se desnecessário se a meta for compartilhada por todas as funções de usuário.

Exemplo — Modelagem de Metas

Em nosso sistema de biblioteca, teríamos as seguintes funções de usuário: FU1 (usuário da biblioteca), FU2 (funcionário responsável pelo empréstimo), FU3 (funcionário responsável pelo cadastro dos exemplares).

As metas de alto nível poderiam ser:

; 1: consultar uma referência (um asterisco ‘’ representa todas as funções de usuário)

FU1, FU2; 2: reservar um exemplar
FU2; 3: registrar um empréstimo

FU2; 4: registrar uma renovação de empréstimo

FU2; 5: registrar uma revolução

FU3; 6: cadastrar um exemplar

FU3; 7: alterar dados de um exemplar

FU3; 8: remover dados de um exemplar

Exemplo — Modelagem de Tarefa

Vamos modelar a tarefa *;1: consultar uma referência.

*;1. consultar uma referência // números indicam a estrutura das metas (1, 1.1, 1.2, 2, etc.)

*;1.1a: se (conhecer dados precisos sobre a referência) // letras indicam regras de seleção
então (realizar busca)
{
1: iniciar busca // métodos são representados entre chaves
2: digitar dados conhecidos
3: disparar busca
4: verificar dados apresentados
5: encerrar consulta
}

*;1.1b: se (não conhecer dados precisos sobre a referência)
então (realizar varredura)
{
1: iniciar varredura
2: comparar referência apresentada com a referência desejada
3a: se (referência apresentada não for a desejada e houver próxima referência)
então ({
3a.1: ir para a próxima referência
2: comparar referência apresentada com a referência desejada //reuso
})
3b: se (referência apresentada for a desejada ou estiver na última referência)
então (encerrar consulta)
}

Exercício

Modele as outras tarefas. Sempre que possíve, reutilize metas e métodos.

3.4 Modelagem de Comunicação

Na modelagem de comunicação, o *designer* tem que definir o que vai dizer, e como. Trata-se de organizar e transmitir aos usuários respostas às duas perguntas fundamentais da Engenharia Semiótica:

1. Qual a interpretação do *designer* sobre o(s) problema(s) do usuário?
2. Como o usuário pode interagir com a aplicação para resolver este(s) problema(s)?

Partindo-se das modelagens de usuários e de tarefas, organizamos as mensagens utilizando signos. Cada signo possui dois aspectos: **expressão**, que é o que se percebe, e **conteúdo**, que é o que o signo significa ou representa. A expressão do signo deve revelar seu conteúdo, ou seja, transmitir informações sobre seu significado e comportamento. Uma tela pode ser considerada um signo complexo, composto de diversos outros signos.

O modelo para a interface como expressão permite ao designer construir uma expressão através de signos de interface. Estes signos são dispostos espacial e temporalmente no *medium* interface, que no caso das interfaces convencionais é a tela do computador.

A estruturação do conteúdo da mensagem é motivada pelo modelo do processo de interação apresentado por Norman que descreve os passos necessários para o usuário realizar o *mapeamento tarefa-ação* [Norman, 1986]. O modelo conceitual da aplicação deve conter **funções de aplicação** ou **do domínio** que o usuário possa utilizar para atingir as metas estabelecidas. Estas funções atuam sobre signos que representam conceitos do domínio e são chamado de **signos do domínio**. As funções da aplicação são controladas por uma estrutura de ações que compõem um **comando de função**. A ativação de uma função pelo seu comando causa uma mudança no estado dos signos do domínio. Os signos do domínio, as funções da aplicação e os comandos de funções são os tipos de elementos básicos que formam o conteúdo da mensagem e precisam ser expressos através dos signos de interface.

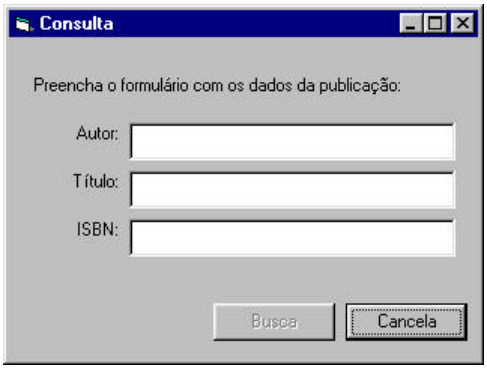


Figura 3.1 — Exemplo de interface para uma função de aplicação

No exemplo da Figura 3.1, temos uma única função da aplicação – a função *busca*. São signos do domínio *publicações* e *autores*. A interface apresenta para o designer o comando associado a esta função e indica quais são as informações sobre os signos do domínio e como o usuário pode acionar os controles da função.

Para apoiar a comunicação dirigida do designer para os usuários, criamos a LEMD (Linguagem de Especificação de Mensagens do Designer [Leite, 1998]). A LEMD diferencia os seguintes tipos de mensagens:

- **Mensagens de metacomunicação direta**, que permitem ao designer enviar uma mensagem diretamente ao usuário para se referir a qualquer componente de usabilidade. Esta mensagem é especificada através do elemento **View** da LEMD. Na Figura 3.1, o texto que vem na parte superior da janela "Preencha o formulário com os dados..." é uma mensagem do designer que apresenta a função e as ações que o designer deve desempenhar para comandá-la.
- **Mensagens sobre estados de signos do domínio**, que revelam o estado do sistema e permitem ao usuário avaliar se a sua meta foi atingida. O estado de um signo como "título da publicação" deve ser representado adequadamente para os usuários. O designer deve decidir, por exemplo, se é melhor comunicar este signo através de tabelas de números ou de gráficos. Como resultado da busca efetuada pela função ilustrada pela Figura 3.1, o designer pode utilizar texto para representar o signos do domínio: nome do autor, título e data da publicação, editora, ISBN, número de referências encontradas e escopo da busca.
- **Mensagens sobre funções da aplicação**, revelando o estado operacional de cada função e o que o usuário deve fazer para controlá-la. Uma das grandes deficiências das aplicações atuais é falta de representação e controle operacional de funções. Na maioria das vezes o único retorno que o usuário tem é o resultado final. O designer deve poder mostrar o desempenho do sistema e permitir que o usuário interrompa ou reinicie quando a função estiver sendo executada. Não mostramos mensagens sobre o estado operacional da função de busca. Entretanto, um signo que mostre se a função

está realizando a busca ou se houve alguma interrupção é um exemplo de mensagem deste tipo.

- **Mensagens sobre interações básicas** indicando ao usuário a interação a ser desempenhada. As interações básicas previstas na linguagem são *acionar* (**Activate**), fornecer informação (**Enter**) e selecionar informação (**Select**). O acionamento pode ser comunicado através de *botões de acionamento*. O fornecimento de informações, expresso na LEMD através de **Enter** pode ser expresso por diversos widgets. Para o usuário fornecer as informações textuais sobre as referências no exemplo da Figura 3.1, o designer optou por utilizar campos de texto (*text boxes*), e para o acionamento da busca um botão de comando (*command button*).
- **Mensagens sobre a estrutura sintática dos comandos**, ou seja, a estrutura e a articulação das interações que o usuário precisa desempenhar. A estrutura sintática determina como as interações básicas podem ser articuladas na formação de comandos compostos. As interações podem ser agrupadas em seqüência (**Sequence**), repetição (**Repeat**), agrupamento (**Join**), combinação (**Combine**) e seleção (**Select**). Na interface para o comando impressão utilizamos algumas destas estruturas. Por exemplo, para que o usuário possa fazer o controle operacional da função ele deve primeiro fornecer as informações associadas aos signos do domínio. Neste caso, existe uma estrutura seqüencial expressa por um layout vertical e pela cor cinza do botão de acionamento, dando a idéia de que eles não está disponível, até que o usuário forneça as informações nos campos superiores.
- **Mensagens de metacomunicação para apresentação e controle da leitura da mensagem** que comunicam como o usuário deve "ler" a própria mensagem do designer. A navegação entre telas, mover, aumentar e diminuir janelas são exemplos de ações que o usuário faz para "ler" a interface, como quem folheia um livro. Estas ações do usuário não modificam o estado funcional do sistema e, portanto, não são considerados comandos em nosso modelo. A LEMD diferencia os controles de leitura dos comandos convencionais chamando a atenção do designer para este aspecto.

Uma especificação completa da LEMD [Leite, 1998] não é necessária para esta introdução a IHC. Porém, podemos exemplificar os efeitos de usá-la através do exemplo que utilizamos. A especificação do modelo de interação pode ser descrita como uma mensagem utilizando a LEMD. A especificação da mensagem que comunica este modelo é a seguinte.

```
Task-Message Busca
Join {
   View "Preencha o formulário com os dados da publicação"
   Sequence {
      Join {
         Enter Information-of Autor
         Enter Information-of Título
         Enter Information-of ISBN
      }
      Join {
         Activate Start Application-Function  Realiza busca
         Activate Waive Application-Functi on Realiza busca
      }
}
```

A mensagem acima deve ser expressa através dos widgets de uma interface gráfica. As construções da interface estão associadas a regras de correlação de acordo com a Tabela 3.1.

Sequence	Configuração espacial com os elementos da seqüência em cinza claro indicando "não disponível"
Join	Configuração espacial (agrupamento visual)
Enter	Campos para digitação de valores.
Activate	Botão de acionamento

Tabela 3.1 — Regras de mapeamento semântico utiliza

A partir da especificação da mensagem do designer e aplicando as regras de correlação associadas a signos de interface da ferramenta Microsoft Visual Basic, pode-se elaborar a mensagem final mostrada na Figura 3.1.

3.5 *Storyboarding*

Esta técnica envolve a especificação através de imagens que descrevam certas situações que estão sendo planejadas. Como permite descrever situações de uso, esta técnica está bastante relacionada com cenários. Entretanto, enquanto cenários utilizam principalmente narrativas para a descrição dos casos de uso, o *storyboarding* utiliza desenhos e ilustrações, utilizando papel ou computadores. Assim, os cenários são mais adequados para a análise das tarefas, enquanto *storyboards* permitem a validação dos cenários e a elaboração de protótipos não operacionais para *designs* iniciais. Além disto, o *storyboarding* é usado de maneira mais livre. A Figura 3.2 apresenta um exemplo de storyboard gerado a partir do cenário exemplificado na seção 3.3.

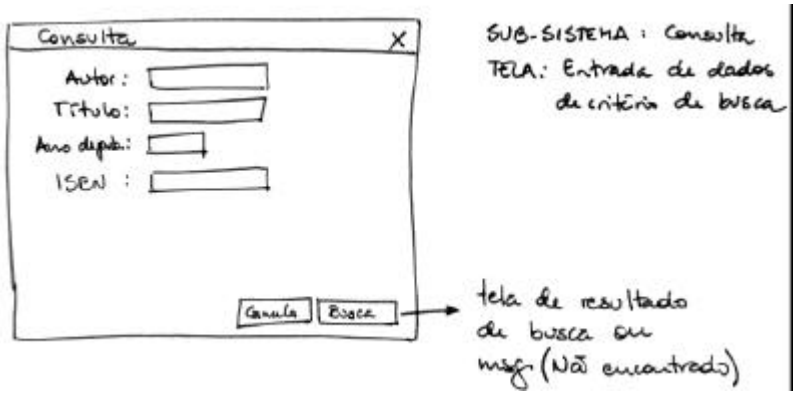


Figura 3.2 — Storyboard para consulta em uma biblioteca.

3.6 Ferramentas de Apoio à Construção de Interfaces

Uma interface gráfica é composta de diversos componentes visuais interativos, chamados *widgets*. Para cada ambiente gráfico, geralmente existe uma ou mais

bibliotecas de widgets. Uma biblioteca define padrões de aparência e comportamento da interface, chamados comumente de *look and feel* de uma interface gráfica.

Para implementar interfaces gráficas utilizando *widgets*, utilizamos ***toolkits***. Um *toolkit* pode ser utilizado para mais de uma plataforma, ou para uma plataforma específica. Um *toolkit* oferece uma ou mais bibliotecas de *widgets*.

Ferramentas de alto nível para apoio à construção de interfaces facilitam a utilização de *toolkits* e sistemas de janelas para a construção de interfaces gráficas. Geralmente são ferramentas de implementação, permitindo construir desde protótipos a sistemas inteiros. Para o ambiente Windows, por exemplo, o Microsoft Visual Basic e o Borland Delphi podem ser considerados ferramentas de apoio à construção de interfaces gráficas.

É importante notar que a utilização de ferramentas de apoio à construção de interfaces não é o suficiente para garantir a qualidade da interface resultante. Elas devem ser utilizadas após terem sido feitas análises e modelagens de usuários, de tarefas e de comunicação da aplicação, ou seja, após o *designer* ter toda a informação necessária para tomar decisões de *design* acertadas.

Existem diversos esforços de pesquisa no sentido de criar ferramentas de apoio ao *design* de interfaces. Seguindo a abordagem da Engenharia Semiótica, desenvolvemos a Linguagem de Especificação de Mensagens do *Designer* [LEMD — Leite, 1998] e modelos de *design* da interação de aplicações de manipulação direta [Martins, 1998], multi-usuário [Prates, 1999], e de aplicações extensíveis [Barbosa, 1999; da Silva et al., 1997]. Um dos objetivos destes trabalhos é justamente possibilitar a criação de ferramentas que auxiliem o *designer* a comunicar suas intenções e pressuposições aos usuários, através da interface da aplicação.

4. AVALIAÇÃO DE INTERFACES

A avaliação da interface é um importante passo do processo de *design*, afinal é através dela que se consegue estimar o sucesso ou insucesso das hipóteses do *designer* sobre a solução que ele está propondo, tanto em termos de funcionalidade, quanto de interação. Ainda que o *designer* se baseie em uma abordagem teórica e conte com a ajuda de diretrizes e princípios de *design*, é necessário que ele avalie o resultado obtido [Hartson, 1998].

As avaliações de interface podem ser classificadas como formativas ou somativas [Preece et al., 1994; Hartson, 1998]. As formativas são aquelas que são feitas durante o processo de *design*, permitindo que identifique e conserte um problema de interação antes que a aplicação seja terminada, ou até mesmo antes de ser implementado. As somativas, por sua vez, avaliam o produto já terminado.

Existem vários métodos possíveis para se coletar e analisar dados. A escolha de qual método deve ser aplicado em um caso específico depende de vários fatores, como o que se deseja avaliar, disponibilidade de pessoas especialistas, ambiente e equipamento para aplicar o teste, acesso ao usuário e dentre outros. Muitas vezes

fatores como o orçamento ou tempo disponível é que decidem o método de avaliação a ser aplicado.

A maior parte dos métodos de avaliação existentes podem ser descritos como observação e monitoração de usuários, coleta da opinião dos usuários, experimentos ou testes de *benchmark*, interpretação de interações que ocorrem naturalmente, ou ainda predição do uso a ser feito da aplicação.

- Observação e monitoração de usuários normalmente é feita informalmente, ou no ambiente de trabalho do usuário ou em um laboratório, e os dados são coletados através de notas do observador ou algum tipo de gravação, como por exemplo de vídeo.
- Coletar a opinião dos usuários é tão importante quanto avaliar o seu desempenho, uma vez que se os usuários não gostarem da aplicação por qualquer razão, eles não a usarão.
- A aplicação de experimentos e testes de *benchmark* em IHC normalmente adotam uma perspectiva semi-científica em compraração com o que normalmente associado a eles. Isto porque as variáveis a ser medidas envolvem interações complexas com seres humanos e o valor obtido muitas vezes pode ser questionado. Assim, este tipo de avaliação normalmente se refere a técnicas mais rigorosamente controladas do que em observação e monitoração de usuários (mesmo que as técnicas sejam as mesmas).
- O objetivo de métodos interpretativos é permitir que o *designer* entenda melhor como usuários usam o sistema em seu ambiente natural e como estes sistemas integram com outras atividades. Assim, dados são coletados informalmente e depois interpretados pelos *designers*. Muitas vezes os usuários participam da coleta, análise ou interpretação dos dados.
- Métodos de predição buscam prever os tipos de problemas que os usuários terão ao usar o sistema, sem no entanto testá-lo com usuários de fato. Normalmente estes métodos envolvem técnicas de modelagem psicológicas dos usuários ou métodos de inspeção, nos quais especialistas em interfaces avaliam o sistema em relação aos problemas que os usuários possivelmente terão.

Nas próximas duas seções apresentaremos duas técnicas de avaliação de interfaces em mais detalhes, Testes de Usabilidade e Testes de Comunicabilidade.

Exercício

Compare os métodos de avaliação formativa e somativa de interfaces. Que vantagens e desvantagens você esperaria de cada um deles?

4.1 Testes de Usabilidade

O objetivo de testes de usabilidade é medir quantitativamente o valor alcançado pelo sistema em cada um dos fatores de usabilidade de interesse (ver seção 1.3).

Para isto, são executados experimentos com os usuários e os valores de cada um destes fatores é medido. Com base na interpretação dos resultados obtidos com os testes, o *designer* conclui se os valores atingidos são ou não satisfatórios.

Uma das técnicas formativas mais conhecidas de testes de usabilidade é a Engenharia de Usabilidade [Nielsen, 1993]. Ela é um método que permite que se aplique um procedimento sistemático para se testar a usabilidade de um produto durante o seu desenvolvimento. Para aplicá-la o *designer* define quais são os fatores de usabilidade prioritários ao seu *design* e define o valor quantitativo desejado para cada um destes aspectos. Para que esta decisão seja tomada, o *designer* deve conhecer ou medir os valores destes fatores no momento de início do projeto[1]. Durante o desenvolvimento do sistema estes valores são medidos e usados para determinar se as metas do sistema já foram alcançadas. É importante ressaltar que os valores definidos como objetivos devem ser valores realistas, senão alcançá-los pode se tornar uma tarefa impossível ou fazer com que o custo de atingi-las seja maior que o benefício.

Exemplo — Fatores Críticos para Usabilidade

Alguns fatores críticos em um sistema de consulta para um usuário de biblioteca são apresentados na tabela a seguir:

fator	**método de medição**	**pior caso**	**nível almejado**	**melhor caso**
taxa de aprendizado	comparação entre primeira e segunda medições	mesma coisa	segunda medição melhor	"muito" melhor
uso inicial	tempo para realizar uma consulta	20 min	1,5 min	0,5 min
uso esporádico, após 2 semanas sem uso	número de vezes em que o sistema de ajuda é acessado em cada tela	mais de uma vez	nenhuma vez	nenhuma vez
preferência sobre fichas impressas	questionário	igual	maior preferência ao sistema	nenhuma preferência às fichas
avaliação inicial	questionário sobre impressões	neutro	positivo	muito positivo
avaliação de usuário experiente	questionário sobre impressões	neutro	positivo	muito positivo
restrições ao uso do sistema	questionário	muitas	poucas	nenhuma

O processo de *design* pode ser então caracterizado como sendo uma iteração *design*-avaliação-*design*. A cada iteração o *designer* avalia os valores dos fatores

[1] Mesmo que ainda não exista um sistema que auxilie o usuário na execução da tarefa, os valores atuais (por exemplo, tempo de execução da tarefa ou número de erros cometidos usando lápis e papel) devem ser medidos [Newman & Lamming, 1995]

para ver se estes já alcançaram o valor desejado. Caso ainda não tenham, o *designer* precisa definir como proceder para que estes valores sejam atingidos.

Exercício

O exemplo anterior mostra alguns fatores relevantes para o sistema de biblioteca. Que outros fatores você citaria como importantes?

Descreva um cenário onde as prioridades dos fatores de usabilidade seriam diferentes das definidas no exemplo.

Observação: Considerando o mesmo cenário, o designer pode definir as prioridades de forma diferente. Por exemplo, pode considerar o fator 'satisfação do usuário' mais importante que 'rapidez na recuperação da informação'.

4.2 Testes de Comunicabilidade

O método de avaliação da comunicabilidade [de Souza et al., 1999] de um software é baseado na Engenharia Semiótica (ver seção 2.2) e tem como objetivo avaliar a sua interface com relação à sua propriedade de comunicabilidade (ver seção 1.3). Para isto, este método propõe um conjunto de interjeições que o usuário potencialmente pode usar para se exprimir em uma situação onde acontece uma ruptura na sua comunicação com o sistema. Estas interjeições de fato não são direcionadas à aplicação, mas sim ao seu *designer*.

A aplicação do método pode ser dividida em duas etapas: a coleta de dados e a análise destes dados. Os passos para se fazer a coleta são:

1. Solicitar ao usuário a execução de uma tarefa pré-determinada na aplicação
2. Gravar a interação do usuário com a aplicação, usando para isto um software de captura as ações do usuário, como por exemplo o Lotus ScreenCam ®. Anotações do aplicador do teste e gravação em vídeo podem ser feitos para enriquecer os dados.
3. Entrevista com o usuário (opcional) sobre a sua interação com a aplicação.
4. Uma vez coletados os dados passa-se para sua análise:
5. Ver gravações da interação e atribuir a interjeição apropriada nos momentos de ruptura da interação.
6. Tabular a informação obtida, ou seja, as interjeições obtidas
7. Interpretar a tabela de acordo com as interjeições e os problemas de usabilidade associados a elas, obtendo então um mapa dos pontos críticos da interação e um perfil da interação da aplicação.

A escolha das interjeições foi feita de forma a se obter um conjunto capaz de expressar as rupturas de interação que acontecem durante o uso de uma aplicação, e ao mesmo ser natural, ou seja, interjeições que fazem parte do cotidiano das pessoas e que potencialmente seriam expressas pelos usuários nestas situações. A

seguir apresentamos o conjunto de interjeições, seus significados e "sintomas", ou seja, ações que permitem atribuir uma determinada interjeição (e não outra) à situação.

- **Cadê? / E agora?** Usuário procura em menus e toolbars por uma função especifica que ele deseja executar. No caso do "E agora?" o usuário não sabe o que fazer e tenta descobrir qual o seu próximo passo.
  Sintomas: Usuário inspeciona menus, sub-menus e *tooltips* (dicas) sem, no entanto, executar nenhuma ação.

- **Que é isso?** O usuário tenta descobrir o que significa um objeto ou ação da interface.
  Sintomas: Usuário coloca cursor sobre algum símbolo da interface esperando um *tooltip*, ou procura o help daquele símbolo, ou ainda hesita entre duas opções que lhe pareçam equivalentes.

- **Epa! / Onde estou?** O usuário executa uma ação que não era a desejada e imediatamente o percebe, desfazendo então a ação. No caso do "Onde estou?" o usuário, sem perceber, executa ações que apropriadas para outros contextos, mas não para o que ele se encontra.
  Sintomas: Usuário executa uma ação e em seguida a desfaz.

- **Por que não funciona? / Ué, o que houve?** A ação executada não obtém o resultado esperado, no entanto, o usuário não entende porque este resultado não foi alcançado. Assim, ele insiste, acreditando que ele tenha cometido algum erro na execução da ação. No caso do "Ué, o que houve?" o usuário não tem *feedback* do sistema e não consegue entender o resultado da sua ação.
  Sintomas: Usuário executa uma ação e não percebe, entende ou aceita o resultado. Ele então repete os mesmos passos para conferir o resultado.

- **Para mim está bom…** O usuário obtém um resultado que ele acredita ser o desejado, mas que não o é.
  Sintomas: Usuário dá a tarefa por terminda sem, no entanto, perceber que não alcançou o resultado desejado.

- **Não dá.** O usuário não é capaz de alcançar o objetivo proposto, ou porque os recursos (tempo, paciência, informação desejada, etc.) não estavam disponíveis, ou porque ele não sabia como.
  Sintomas: Usuário abandona a tarefa sem ter conseguido atingir seu objetivo.

- **Deixa pra lá… / Não, obrigado.** O usuário não entende as soluções de interação primárias oferecidas pelo *designer*, e resolve seu problema de alguma outra forma. No caso do "Não, obrigado." ele entende a solução, mas prefere outras formas de interação.
  Sintomas: Usuário não encontra a forma de interação principal oferecida para se executar uma ação, ou então decide não usá-la. Note-se que este caso tipicamente segue uma interjeição de "Onde está?"

A maior parte dos problemas de interação e usabilidade podem ser classificados como sendo de falha na execução da tarefa, navegação, atribuição de significado

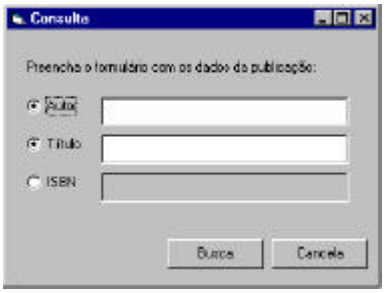
ou de não percepção. Problemas de falha na execução da tarefa são os mais graves, uma vez que o usuário não neste não consegue atingir o seu objetivo que o levou a usar a aplicação. Os de navegação se referem àqueles nos quais os usuários se "perdem" durante a interação com o sistema. Os de atribuição de significado, conforme o nome diz, o usuário não é capaz de atribuir um significado (relevante) a signos encontrados na interface. Finalmente, no caso dos de não percepção aspectos da interface e da aplicação passam despercebidos ao usuário. Na Tabela 4.1 mostramos como as interjeições acima podem ser associadas a estas classes de problemas, e acrescentamos ainda mais uma: a de recusa de formas de interação. Neste caso, por algum motivo o usuário decide não usar algumas destas formas disponibilizadas pelo *designer*.

Interjeição	**Problema**
Cadê? E agora?	Navegação
Que isso?	Atribuição de significado
Epa! Onde estou?	Navegação / Atribuição de significado
Por que não funciona? Ué o que houve?	Atribuição de significado
Para mim está bom…	Atribuição de significado
Não dá.	Falha de execução da tarefa
Deixa para lá… Não, obrigado.	Incompreensão de como usar *affordance* Recusa de usar *affordance*

Tabela 4.1 — Associação entre interjeições e problemas de interação e usabilidade.

Exemplo — Ruptura de Comunicabilidade

Considere a tela de consulta abaixo. O usuário que esteja familiarizado com o ambiente Windows espera que os botões de opção sejam mutuamente exclusivos. Entretanto, dois deles estão selecionados simultaneamente. Ao se deparar com uma situação destas, o usuário tipicamente se pergunta "O que houve?" ou "Por que não funciona?". Observe que, neste caso, "funcionar" diz respeito às expectativas do usuário com relação ao comportamento de botões de opção.



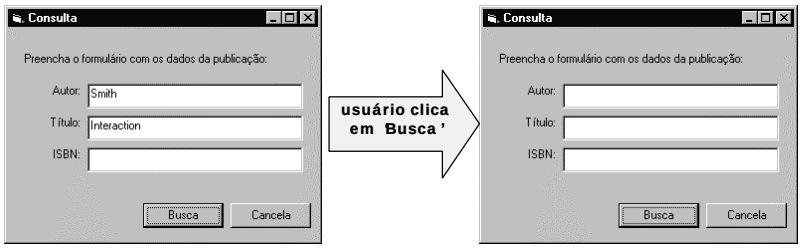
Este exemplo revela uma ruptura de atribuição de significado, entre o comportamento esperado, transmitido pela escolha de um determinado componente de interface, e o comportamento real deste componente.
Sempre que se utiliza componentes de uma biblioteca de *widgets*, é fundamental conhecer seu comportamento e em que contextos de uso devem ser utilizados.

Que critérios deveriam ser levados em conta na seleção de uma tarefa para um teste de comunicabilidade?

- Os pontos onde o designer está ciente ou acredita que possa haver ruptura de comunicação. Alguns destes pontos podem surgir devido ao dilema de comunicabilidade.
- Pontos críticos da aplicação, ou seja, onde o usuário não pode errar.

Exercício

Classifique as rupturas de comunicabilidade que podem ocorrer na seqüência de telas a seguir.



4.3 Testes de Usabilidade x Testes de Comunicabilidade

Apesar de os testes de usabilidade e comunicabilidade serem capazes de identificar alguns problemas comuns, eles focalizam diferentes aspectos da interface. Testes de usabilidade fornecem medidas quantitativas sobre o produto, mesmo que durante o processo de *design*. Assim, mesmo que através deles se determine aspectos do produto que precisam ser modificados, eles não fornecem nenhum indicador de que ação tomar para se alcançar o resultado desejado.

Os testes de comunicabilidade também geram medidas quantitativas e estas podem ser usadas para avaliar os problemas mais graves, ou pelo menos mais freqüentes, de interação e usabilidade do produto. No entanto, estes testes fornecem também uma avaliação qualitativa da interface, à medida que identificam pontos de ruptura da comunicação entre *designer* e usuários[2]. Como a identificação é feita através das interjeições potenciais do usuário, ela não apenas

[2] Estes pontos de ruptura identificam uma mudança do nível de abstração desta comunicação, uma vez que quando isto acontece a "conversa" entre *designers* e usuários deve deixar de ser sobre o problema sendo resolvido e passar a ser sobre a interação.

aponta que existe uma ruptura, mas ela também informa ao *designer* o que foi que o usuário não entendeu, e dá indicadores de que tipo de informação precisa ser alterada ou acrescentada para que ele entenda.

Tanto testes de usabilidade quanto de comunicabilidade podem funcionar tanto como formas de avaliação formativa ou somativa. No entanto, vale a pena ressaltar que para se aplicar e analisar testes de usabilidade, é necessário ser especialista, tanto em interfaces, quanto em usabilidade. Já em testes de comunicabilidade, como as interjeições são parte do vocabulário cotidiano das pessoas, outros *designers* (não especialistas em interface ou comunicabilidade), ou até mesmo usuários poderiam participar da etapa de identificação de rupturas e coleta de dados[3]. As etapas de análise e de propostas de solução continuam tendo que ser feitas por um especialista. Entretanto, *designers* e usuários que tivessem participado da etapa anterior poderiam ser colaboradores e fornecer *insights* interessantes, uma vez que eles saberiam quais são os problemas a serem resolvidos.

Por avaliarem diferentes aspectos da interface, em certas situações pode-se desejar aplicar ambos, de forma complementar, a uma mesma interface. Assim, se teria medidas quantitativas e qualitativas da interface que dariam indicações tanto sobre aspectos do desempenho desta interface, quanto da sua qualidade de comunicação.

Exercícios

Crie um cenário que ilustre uma situação onde um teste de usabilidade seja mais apropriado. Faça o mesmo para teste de comunicabilidade.

Você consegue imaginar um cenário onde uma avaliação de comunicabilidade não tenha nada para acrescentar, quando comparado a uma avaliação de usabilidade? E o contrário?

4.4 Prototipação

Um protótipo é uma aplicação, normalmente experimental e incompleta, que permite aos *designers* avaliarem suas idéias de *design* durante o processo de criação da aplicação pretendida. Ele deve ser construído rapidamente e com baixo custo e seu tempo de vida não é definido.

Dentre as informações extraídas de um protótipo, podemos destacar a funcionalidade necessária ao sistema, seqüências de operação, necessidades de suporte ao usuário, representações necessárias, *look and feel* da interface [Preece et al., 1994] e comunicabilidade da aplicação.

Um protótipo pode ser classificado em relação à sua função no processo de desenvolvimento, o seu objetivo de avaliação e a técnica de sua construção. Em relação à sua função no processo de desenvolvimento ele pode ser classificado como:

[3] Esta facilidade faz com que esta etapa do teste de comunicabilidade possa ser feita com um orçamento menor, onde se tem a participação dos próprios clientes.

Função	Descrição
Apresentação	Permite ao designer apresentar ao cliente a sua percepção do sistema, mostrando que ele é viável e que a sua interface se adequa aos requisitos do usuário.
Autêntico	Ilustra aspectos específicos da interface de usuários ou da funcionalidade, ajudando na compreensão dos problemas envolvidos. Normalmente é provisório e funcional.
Funcional	Ajuda a equipe de desenvolvimento compreender questões relacionadas com a construção do sistema. Esse protótipo é derivado do modelo de domínio ou da especificação do software e não interessa aos usuários.
Sistema Piloto	Contém um núcleo básico da aplicação a ser experimentado com os usuários. Assim, é usado para fins mais do ilustrativos.

Em relação ao seu objetivo de avaliação, a prototipação pode ser classificada como exploratória, experimental e evolutiva. A prototipação **exploratória** é usada para ajudar a esclarecer requisitos do usuários, ou para examinar uma variedade de opções de solução de *design* para que se determine a mais adequada. A prototipação **experimental** enfatiza aspectos técnicos do desenvolvimento, oferecendo aos desenvolvedores resultados experimentais para a tomada de decisões de *design* e implementação. Finalmente, a prototipação **evolutiva** avalia o impacto que a introdução de novas tecnologias podem trazer para uma pessoa, seu modo de trabalhar e para a organização como um todo. Neste caso, os *designers* devem trabalhar em cooperação com os usuários em um processo contínuo de reengenharia.

Por fim podemos classificar as técnicas de construção do protótipo como completa, horizontal ou vertical. Um protótipo **completo** contém toda a funcionalidade da aplicação pretendida, mas porém com baixo desempenho. O protótipo **horizontal** exibe apenas uma camada específica da aplicação, como por exemplo a camada de interface. O **vertical**, por sua vez, apresenta a implementação completa (interação e funcionalidade) de uma parte restrita da aplicação.

Como mencionamos anteriormente, o protótipo não tem tempo de vida definido. Em certos casos, ele é temporário e uma vez usado para avaliar o aspecto desejado, ele é jogado fora. Em outros, ele vai sendo incrementado e se torna parte integrante do sistema.

Uma das grandes vantagens de protótipos é eles serem parte de um *design* iterativo centrado no usuário, permitindo que os *designers* experimentem idéias junto a usuários e recebam seu *feedback*. Assim, eles são uma prática comum na avaliação da interpretação dos requisitos de *design*, alternativas de solução e das soluções propostas.

Exemplo

O resultado de uma consulta a um sistema de biblioteca pode conter diversas referências. Para avaliarmos o tipo de tela de resultado desta busca, poderíamos construir um protótipo horizontal, que contenha diferentes organizações da informação. Por exemplo, uma opção seria apresentar uma das referências encontradas na consulta de cada vez, em detalhes, com a possibilidade de navegar seqüencialmente pelas referências. Outra opção seria apresentar uma lista que resumisse as informações mais importantes da referência, e permitir que o usuário selecione um dos itens da lista, caso deseje ver mais detalhes. Para este tipo de avaliação, utilizamos um protótipo horizontal parcial, ou seja, apenas as telas e signos de interface envolvidos nesta tarefa, sem que seja necessário implementar o acesso a um banco de dados real.

Em contrapartida, se a rapidez e eficiência dos algoritmos de busca for especificada como um fator crítico nesta aplicação, seria necessário fazer uma implementação vertical desta funcionalidade, de forma a permitir fazer a avaliação de desempenho, possivelmente utilizando diversos algoritmos diferentes para compará-los.

Exercícios

Em um sistema de biblioteca, que tipo de protótipo você construiria para:

- convencer um bibliotecário a comprar a aplicação?
- avaliar a melhor forma de um funcionário cadastrar os dados de uma publicação?
- testar a comunicação com a base de dados?

REFERÊNCIAS BIBLIOGRÁFICAS

ACM SIGCHI (1992) "Curricula for human-computer interaction". Technical report, ACM, NY, 1992. Disponível on-line em http://www.acm.org/sigchi/.

Adler, P. & Winograd, T. (eds., 1992) *Usability: Turning Technologies into Tools*. Oxford University Press. New York, NY.

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

Apple Computer, Inc. (1992) *Macintosh Human Interface Guidelines*. Reading, Ma. Addison Wesley.

Barbosa, S.D.J. (1999) *Programação Via Interface.* Tese de Doutorado. Departamento de Informática, PUC-Rio.

Card, S.; Moran,T. & Newell A. (1983) *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Carey, T. e Rusli, M. (1995) "Usage Representations for Reuse of Design Insights: A Case Study of Access to On-line Books. In Carroll (ed.) *Scenario-Based Design: Envisioning Work and Technology in System Development*. New York, NY: John Wiley & Sons.

Carroll et al. (1994) " Binding Objects to Scenarios of Use", International Journal of Human-Computer Studies.

da Silva, S.R.P.; de Souza, C.S.; Ierusalimschy, R. (1997) "A Communicative Approach to End-User Programming Languages". Em Lucena, C.J.P. (ed.) *Monografias em Ciência da Computação*. Departamento de Informática. PUC-RioInf MCC 47/97. Rio de Janeiro.

de Souza, C.S.; Prates, R.O.; Barbosa, S.D.J. (1999) "A Method for Evaluating Software Communicability". Em Lucena, C.J.P. (ed.) *Monografias em Ciência da Computação*. Departamento de Informática. PUC-RioInf MCC 11/99. Rio de Janeiro. 11p.

de Souza, C.S. (1993) "The Semiotic Engineering of User Interface Languages". International Journal of Man-Machine Studies 39. Academic Press. pp. 753-773.

de Souza, C.S. (1996) "The Semiotic Engineering of Concreteness and Abstractness: from User Interface Languages to End-User Programming Languages". Em Andersen, P.; Nadin, M.; Nake, F. (1996) *Informatics and Semiotics*. Dagstuhl Seminar Report No. 135, p. 11. Schloß Dagstuhl., Germany.

de Souza, C.S. (1999) "Semiotic engineering principles for evaluating end-user programming environments". Em Lucena, C.J.P. (ed.) *Monografias em Ciência da Computação*. Departamento de Informática. PUC-RioInf MCC 10/99. Rio de Janeiro. 23p.

Dix, A.; Finlay, J.; Abowd, G.; e Beale, R. (1993) *Human-Computer Interaction*. Prentice-Hall International.

Eco, U. (1976) A Theory of Semiotics. Bloomington, IN: Indiana University Press.

Fischer, G. (1998) "Beyond 'Couch Potatoes': From Consumers to Designers" In *Proceedings of the 5th Asia Pacific Computer-Human Interaction Conference*. IEEE Computer Society. pp.2-9.

Hartson, H.R. (1998) "Human-Computer Interaction: Interdisciplinary roots and trends". In *The Journal of System and Software*, **43**, 103-118.

Hix, D. & Hartson, H. (1993) *Developing User Interfaces: Ensuring Usability Through Product & Process*. John Wisley & Sons.

Jakobson, R. (1970) Lingüística e Comunicação. Cultrix, São Paulo.

Jorna, R. & Van Heusden, B. (1996). Semiotics of the user interface. *Semiotica*, 109(3/4):237-250.

Kammersgaard (1988) "Four differents perspectives on Human-Computer Interaction." *in International Journal of Man-Machine Studies*, 28, 343-362

Lee, G. (1993) *Object-Oriented GUI Application Development*. NJ: Prentice Hall.

Leite, J. C. (1998) *Modelos e Formalismos para a Engenharia Semiótica de Interfaces de Usuário*. Tese de Doutorado. Departamento de Informática. PUC-Rio.

Lindgaard. G. (1994) *Usability Testing and System Evaluation* London, UK: Chapman & Hall.

Martins, I.H. (1998) *Um Instrumento de Análise Semiótica para Linguagens Visuais de Interfaces*. Tese de Doutorado. Departamento de Informática. PUC-Rio, Brasil.

Microsoft Corporation. (1995) The Windows Interface Guidelines for Software Design. Redmond. Microsoft Press.

Moran, T. (1981) "The Command Language Grammars: a represetantion for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, 15, 3-50.

Nadin, M. (1988) "Interface Design and Evaluation – Semiotic Implications". Em Hartson, R. e Hix, D. (eds.), *Advances in Human-Computer Interaction*, Volume 2, 45-100.

Newman, W. & Lamming, M. (1995) *Interactive System Design*. Addison-Wesley.

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

Norman, D. (1986) Cognitive Engineering. In D. Norman & S. Draper (eds.) User Centered System Design. Hillsdale, NJ. Lawrence Erlbaum. pp.31-61.

Norman, D. (1988) *Psychology of Everyday Things.* BasicBooks. HarperCollins Publishers.

Norman, D. (1991) "Cognitive Artifacts". In Carroll (ed.) *Designing Interaction: Psychology ate the Human-Computer Interface*. pp.17-38.

Norman, D. (1993) *Things that make us smart: defending human attributes in the age of the machine.* Reading, MA: Addison-Wesley.

Paap, K.R. e Roske-Hofstrand, R.J (1988). "Design of Menus" In Helander (ed.) *Handbook of Human-Computer Interaction.* Amsterdam: North-Holland.

Payne, S. e Green, T.R.G. (1989). Task-action grammar: the model and its developments. In Diaper (ed.) *Task Analysis for Human-Computer Interaction*. Chichester: Ellis Horwood.

Peirce, C.S. (1931-1958). *Collected Papers.* Edição brasileira: Semiótica. São Paulo, Ed. Perspectiva (coleção estudo, n.46) 1977.

Prates, R.O. (1999) *A Engenharia Semiótica de Linguagens de Interfaces Multi-Usuário.* Tese de Doutorado. Departamento de Informática. PUC-Rio, Brasil.

Preece, J.; Rogers, Y.; Sharp, E.; Benyon, D.; Holland, S.; Carey, T. (1994) *Human-Computer Interaction.* Addison-Wesley.

Shneiderman, B. (1998) *Designing the User Interface, 3rd Edition.* Reading, MA: Addison Wesley.