

6. *Elaboração de esboços de interface a partir de um modelo de interação.* Selecione uma das soluções modeladas e elabore esboços de interface alternativos para concretizar a solução de IHC: um conjunto de esboços a partir de um modelo de tarefas e outro a partir de um modelo de interação. Discuta as diferenças nas decisões tomadas em cada caso e nas soluções elaboradas.
7. *Sistema de ajuda.* Escolha um dos objetivos apoiados pelos esboços elaborados na atividade anterior e elabore o conteúdo de ajuda correspondente. Discuta as formas de acesso a esse conteúdo que podem ser fornecidas a partir da interface, indicando nos esboços esses pontos de acesso.
8. *Oportunidades de adaptação.* Considerando diferentes perfis de usuários, atividades e contextos de uso do sistema, identifique oportunidades de adaptação manual ou automática e reveja as soluções modeladas e esboçadas para possibilitar essas formas de adaptação. Discuta meios de comunicar aos usuários de que maneira eles podem adaptar o sistema ou ajustar as adaptações realizadas automaticamente pelo sistema.

8

Princípios e Diretrizes para o Design de IHC

Objetivos do Capítulo

- Apresentar princípios e diretrizes para o design de IHC, exemplificando seu uso.
- Discutir os benefícios de se utilizar padrões de design de IHC e apresentar alguns modelos de documentação de padrões.
- Descrever brevemente o uso de guias de estilo e apresentar uma estrutura para esse documento.

Este capítulo apresenta alguns princípios, diretrizes (Norman, 1988; Tognazzini, online; Nielsen, 1993; Shneiderman, 1998; Cooper, 1999) e padrões de design (Tidwell, 2005) comumente utilizados na construção das interfaces de usuário.

8.1 Introdução

A literatura de IHC está repleta de conjuntos de princípios, diretrizes (*guidelines*) e heurísticas. Princípios costumam representar objetivos gerais e de alto nível; diretrizes, regras gerais comumente observadas na prática; e padrões, soluções específicas a certos contextos bem delimitados, envolvendo certos usuários desempenhando determinadas tarefas (Mayhew, 1999). Entretanto, essa classificação nem sempre é utilizada precisamente, e alguns proponentes de diretrizes as intitulam de princípios. Os conjuntos mais conhecidos de princípios e diretrizes são os de Norman (1988), de Tognazzini (2003),¹ de Nielsen (1993)² e as regras de ouro de Shneiderman (1998).

Todos os pesquisadores e profissionais de IHC ressaltam que o uso de princípios e diretrizes jamais substitui as demais atividades de análise, design (conceitual e concreto) e avaliação. Embora princípios e diretrizes possam ser utilizados como auxílio ao design, elas não substituem um processo cuidadoso que inclui a busca pelo entendimento do problema, a elaboração de soluções candidatas e a avaliação dessas soluções alternativas.

Alguns conjuntos de diretrizes são desenvolvidos especificamente para certos ambientes de trabalho, como o Windows®, o MacOs® e o Gnome®, para certos dispositivos, como dispositivos móveis e televisão digital interativa, e certos domínios, como educação, governo eletrônico etc. Alguns conjuntos de diretrizes, como os voltados para certos ambientes de trabalho, visam motivar designers a seguir uma certa padronização para assegurar aparência e comportamento (*look and feel*) semelhantes com o que os desenvolvedores ou alguma outra organização propuseram. Entretanto, diretrizes frequentemente são recomendações genéricas e descontextualizadas da teoria ou base empírica que lhes deram origem. Além disso, muitas vezes duas ou mais diretrizes são conflitantes. A aplicação adequada de boa parte dos princípios e diretrizes depende, em alguma medida, do conhecimento do designer acerca do domínio do problema, dos usuários e das suas atividades nesse domínio. Sendo assim, cabe ao designer considerar cuidadosamente se e quais diretrizes são adequadas à sua situação de design, e como elas devem se manifestar na solução de IHC.

¹ <http://www.asktog.com/basics/firstPrinciples.html>.

² Nielsen (1993) denomina seu conjunto de diretrizes alternadamente de princípios, diretrizes e heurísticas.



Diretrizes são comumente utilizadas em listas de verificação (*checklists*), em que inspetores examinam uma interface para avaliar se ela está em conformidade com o conjunto selecionado de diretrizes (veja a Seção 9.2). No Brasil, temos como exemplo a ErgoList,³ lista de verificação desenvolvida com base em princípios de ergonomia.

8.2 Princípios e Diretrizes Gerais

Norman (1988) destaca a necessidade de projetarmos o sistema utilizando um modelo conceitual que o usuário possa apreender rapidamente e sem dificuldade. O modelo conceitual deve auxiliar a interpretar o relacionamento entre as ações e informações apresentadas pelo sistema e o conhecimento no mundo.

Segundo Norman (1988), o design deve facilitar: determinar quais ações são possíveis a cada momento, fazendo uso de restrições (*constraints*); tornar as coisas visíveis, incluindo o modelo conceitual do sistema, as ações alternativas e os resultados das ações; avaliar o estado corrente do sistema e seguir mapeamentos naturais entre as intenções e as ações requeridas, entre as ações e o efeito resultante, e entre a informação que está visível e a interpretação do estado do sistema.

Os princípios e as diretrizes comumente utilizados em IHC giram em torno dos seguintes tópicos: correspondência com as expectativas dos usuários; simplicidade nas estruturas das tarefas; equilíbrio entre controle e liberdade do usuário; consistência e padronização; promoção da eficiência do usuário; antecipação das necessidades do usuário; visibilidade e reconhecimento; conteúdo relevante e expressão adequada; e projeto para erros. As próximas subseções apresentam algumas diretrizes e ilustram como podem ser utilizadas no design da interação e da interface.

8.2.1 Correspondência com as Expectativas dos Usuários

Segundo Norman (1988), devemos *explorar os mapeamentos naturais*, seja entre as variáveis mentais e as físicas, seja entre as tarefas e os controles utilizados para manipular essas variáveis no mundo real e no sistema projetado (veja Seção 3.4). Deveremos nos certificar de que o usuário consegue determinar os relacionamentos entre: intenções e ações possíveis; entre ações e seus efeitos no sistema; entre o estado real do sistema e o que é percebido pela visão, audição ou tato; entre o estado percebido do sistema e as necessidades, intenções e expectativas do usuário.

Por exemplo, ao projetar um sistema de comércio eletrônico, devemos examinar como as pessoas fazem suas compras em lojas físicas: uma pessoa entra em uma loja, escolhe um ou mais produtos (com ou sem ajuda de um vendedor), e somente precisa se identificar no momento de finalizar a compra e pagar pela mercadoria escolhida.

³ <http://www.labutil.inf.ufsc.br/ergolist/>.

Da mesma maneira, um sistema de comércio eletrônico deve permitir ao usuário buscar e selecionar os produtos anonimamente, e exigir sua identificação apenas no momento de finalizá-la (Figura 8.1a). Entretanto, ainda hoje, alguns sistemas na Web exigem que o usuário se identifique antes mesmo de encontrar um produto do seu interesse (Figura 8.1b).

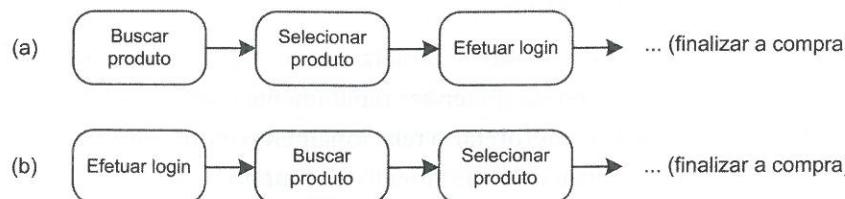


Figura 8.1 Sequências alternativas de cenas em sites de comércio eletrônico, solicitando a identificação do usuário em diferentes momentos da interação.

Shneiderman (1998) recomenda *estruturar o diálogo de forma a seguir uma linha de raciocínio e fornecer um fechamento*. Segundo ele, as sequências de ações devem ser organizadas em grupos com início, meio e fim. Segundo Nielsen (1993), o projetista deve seguir as convenções do mundo real, fazendo com que a informação apareça em uma ordem natural e lógica. Para isso, é importante entendermos as sequências de ações que são familiares aos usuários e, caso a solução se desvie do que lhes é familiar, deve ao menos refletir uma organização lógica que lhes seja plausível. Além disso, ele ressalta a importância de fornecer um *feedback* informativo na conclusão de um grupo de ações (veja Seção 8.2.7), para proporcionar aos usuários a satisfação de terem concluído uma tarefa, um sentimento de alívio, um sinal de que podem deixar de lado planos de contingência que tiverem formulado e uma indicação de que já podem se preparar para o novo grupo de ações.

Além dos aspectos estruturais, o designer deve projetar a interface utilizando o idioma do usuário, com palavras, expressões e conceitos que lhe são familiares, em vez de utilizar termos orientados ao sistema ou a desenvolvedores.

Tognazzini (2003) recomenda o *uso de metáforas* de forma cuidadosa, para permitir que os usuários identifiquem rapidamente sutilezas do modelo conceitual subjacente ao sistema. Boas metáforas são como histórias, criando imagens na mente. Elas devem evocar algo familiar, mas geralmente trazem também algo novo. Por exemplo, uma pasta num gerenciador de arquivos não tem a mesma limitação de número de itens de conteúdo que uma pasta física.



8.2.2 Simplicidade nas Estruturas das Tarefas

Norman (1988) recomenda *simplificar a estrutura das tarefas*, reduzindo a quantidade de planejamento e resolução de problemas que elas requerem. Tarefas desnecessariamente complexas podem ser reestruturadas, em geral utilizando inovações tecnológicas. Para simplificar a estrutura das tarefas, os designers podem seguir quatro abordagens tecnológicas: a) manter a tarefa a mesma, mas fornecendo diversas formas de apoio para que os usuários consigam aprender e realizar a tarefa; b) usar tecnologia para tornar visível o que seria invisível, melhorando o *feedback* e a capacidade de o usuário se manter no controle da tarefa; c) automatizar a tarefa ou parte dela, mantendo-a igual; e d) modificar a natureza da tarefa. Norman alerta para o perigo de a automação tirar controle demais do usuário, escravizando-o ou tornando-o tão confiante e dependente da tecnologia a ponto de reduzir ou até mesmo eliminar sua capacidade de trabalhar sem a automação.

8.2.3 Equilíbrio entre Controle e Liberdade do Usuário

Norman (1988), Nielsen (1993), Tognazzini (2003), Shneiderman (1998) e Cooper (1999) destacam a importância de manter o usuário no controle. Tognazzini (2003) lembra que o computador, a interface e o ambiente de trabalho “pertencem” ao usuário. Ele afirma que, quando deixamos o usuário “no comando”, ele aprende rapidamente e ganha um sentimento de maestria. Entretanto, ele ressalta a necessidade de buscar um equilíbrio, pois quando não há limites ou restrições os usuários podem se sentir perdidos ou angustiados com o excesso de opções. Devemos tentar reduzir o número de opções ou decisões que o usuário precisa tomar a cada instante. Norman (1988) recomenda *explorar o poder das restrições, tanto naturais como artificiais*, e projetar restrições para que o usuário sinta como se houvesse apenas uma coisa possível a fazer: a coisa “certa”, é claro.

Segundo Tognazzini (2003), os usuários não devem ficar presos num caminho de interação único para realizar uma atividade. O caminho mais rápido ou preferencial pode ser o de “menor resistência”, mas usuários que queiram explorar diferentes alternativas e cenários devem conseguir fazê-lo.

Sempre deve ser fornecida aos usuários uma “saída” clara e rápida, mas deve ser mais fácil se manter “no caminho” do que sair dele inadvertidamente (Figura 8.2). A decisão entre oferecer mais ou menos liberdade ao usuário varia com o seu perfil. Usuários mais inexperientes podem precisar de mais assistência e menos alternativas, ao passo que usuários experientes de interfaces de uso frequente devem poder comandá-la como melhor lhes convier.

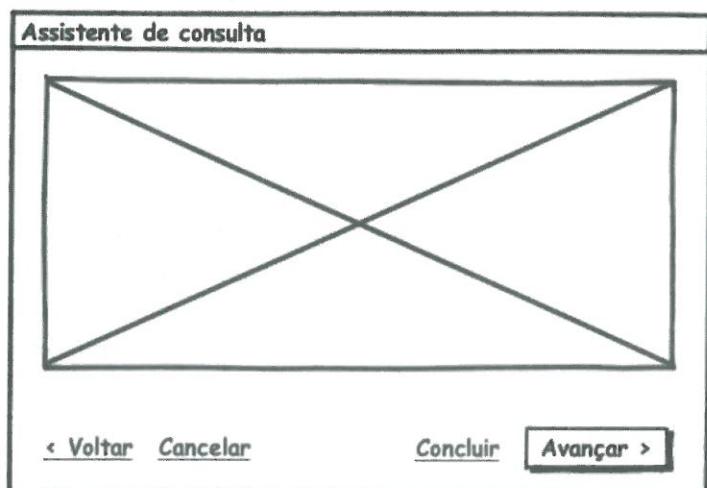


Figura 8.2 Tela de assistente com indicação clara do botão primário.

Cooper (1999) afirma que o software deve ser *maleável* e ter “jogo de cintura”. Para ele, isso significa aceitar permanecer em estados intermediários e realizar algumas ações fora de ordem ou antes que seus pré-requisitos tenham sido satisfeitos. Em outras palavras, não deve forçar os usuários a trabalharem de um único modo. Entretanto, deve sempre guardar um registro de tudo o que tiver sido feito para que o responsável por cada ação possa ser identificado.

Shneiderman (1998) recomenda permitir que o usuário tenha controle local da interação, ou seja, que o usuário inicie as ações, em vez de apenas reagir a ações do sistema. Ele se refere principalmente a usuários experientes, que costumam querer sentir que controlam o sistema e o sistema responde às suas ações, e não o contrário. Além dele, Nielsen (1993) e Tognazzini (2003) também destacam a importância de permitir que o usuário cancele, desfaça e refaça suas ações. Os usuários frequentemente escolhem funções do sistema por engano e precisam de uma “saída de emergência” claramente marcada para sair do estado indesejado sem ter de percorrer um diálogo extenso (Figura 8.3).

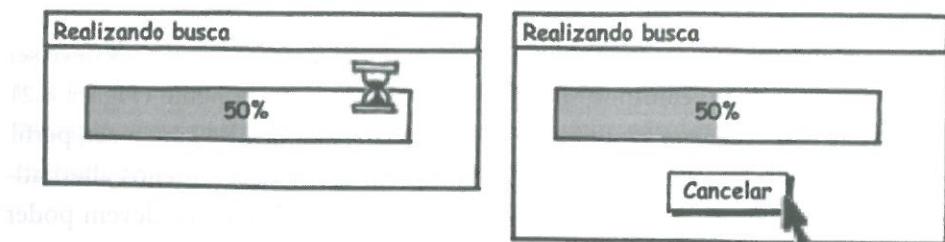


Figura 8.3 Telas de progresso sem e com opção de cancelar.

Permitir que o usuário desfaça suas ações reduz a sua ansiedade e o medo de errar, pois ele sabe que os erros podem ser desfeitos. Essa possibilidade também é importante para o *aprendizado por exploração*. Os usuários frequentemente exploram partes da interface para descobrir o que aconteceria se eles fizessem isso ou aquilo, e assim aprendem mais sobre o sistema e sobre seu próprio trabalho. Para isso, é importante que ações potencialmente perigosas possam ser desfeitas, pois podem ser acionadas equivocadamente pelos usuários. Se um usuário pedir para apagar um arquivo, por exemplo, o sistema deve obedecer e se preparar para desfazer a ação, caso solicitado.

Segundo Cooper (1999), a possibilidade de desfazer ações evita a necessidade de apresentar diversos diálogos pedindo confirmação das ações dos usuários. Usar diálogos de confirmação em excesso e de forma indiscriminada não apenas aumenta o tempo de realização das tarefas, mas também pode tornar a comunicação ineficiente, pois muitos usuários acabam prosseguindo a interação sem mesmo ler o conteúdo desses diálogos. Quando uma operação considerada perigosa não puder ser desfeita, devemos projetar medidas de segurança para que ela não seja acionada incidentalmente (Figura 8.4).

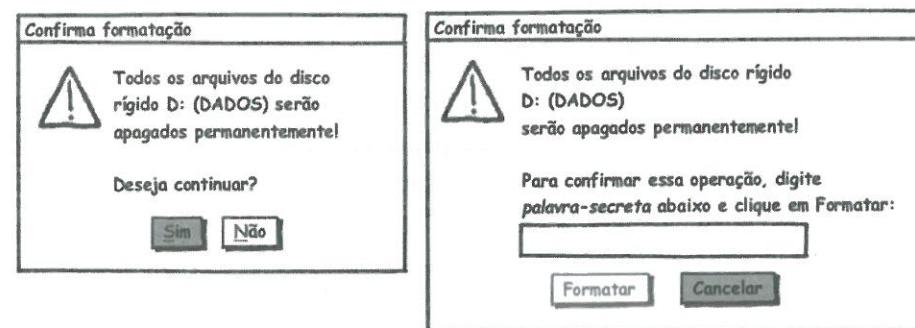


Figura 8.4 Diálogos de confirmação para uma operação que não pode ser desfeita. Qual é a mais segura?

Para aumentar o controle do usuário sobre a interação, o sistema também deve permitir que os usuários trabalhem de modo *flexível*, através de parâmetros configuráveis, por exemplo. Entretanto, devemos buscar um equilíbrio entre a gama de opções oferecidas ao usuário e sua capacidade de entender as consequências da combinação de parâmetros escolhida. O sistema não deve forçar o usuário a escolher o tempo todo um sem-número de opções para prosseguir rumo ao seu objetivo (Cooper, 1999). Daí a importância de escolher bons valores padrão (*defaults*) para quando não for necessário incomodar o usuário.



8.2.4 Consistência e Padronização

Para facilitar o aprendizado e uso de um sistema, Norman (1988) recomenda *assegurar a consistência da interface com o modelo conceitual embutido no sistema*. Isso requer que tudo sobre o produto (modelo de design e imagem do sistema — veja Seção 3.4 —, incluindo documentação e manuais de instrução) esteja consistente com e exemplifique a operação do modelo conceitual adequado.

Tanto Norman (1988) como Tognazzini (2003) acreditam que a consistência mais importante é com as expectativas dos usuários, como visto na seção anterior. Segundo eles, mesmo quando essa correspondência não é possível, ou seja, quando precisamos definir mapeamentos arbitrários, devemos *padronizar*. Norman, Tognazzini, Nielsen e Schneiderman recomendam padronizar as ações, os resultados das ações, o *layout* dos diálogos e as visualizações de informação. Ações relacionadas em situações semelhantes devem funcionar da mesma forma. Por exemplo, um botão Fechar não deve ser utilizado para cancelar um diálogo em algumas situações e para confirmá-lo em outras.

Os usuários não devem ter de se perguntar se palavras, situações ou ações diferentes significam a mesma coisa. Por exemplo, utilizar rótulos Salvar e Gravar indiscriminadamente em um mesmo sistema pode confundir o usuário. A mesma terminologia deve ser utilizada em perguntas, menus e sistemas de ajuda. Nielsen (1993) recomenda ainda seguir as convenções da plataforma ou ambiente computacional. Segundo Cooper (1999), um sistema que se comporte de modo irregular ou instável não é confiável.

Tognazzini (2003) ressalta ainda que alguns elementos de interface exigem maior consistência do que outros. Em particular, elementos que não possuem correspondência visual na interface ou cuja operação é internalizada pelos usuários (e acionada “sem pensar”) não devem variar em diferentes partes do sistema. Por exemplo, sequências de teclas de atalho e o comportamento da roda do mouse para rolamento ou zoom devem ser padronizados em todo o sistema.

Em contrapartida, se dois elementos de interface possuem comportamento diferente, eles devem ter aparências distintas. Por exemplo, um elemento de interface utilizado para selecionar uma opção deve ser distinto de um elemento utilizado para disparar uma ação do sistema.

Tognazzini (2003) alerta, no entanto, que eventualmente queremos propositalmente tornar algo inconsistente, para que o usuário não possa atuar de forma automática e precise refletir sobre o que está fazendo. Esse é o caso, por exemplo, de diálogos de confirmação em que os botões Sim e Não são substituídos por rótulos

mais representativos dos efeitos de cada botão, e que não podem ser acionados pelo simples pressionamento das teclas S ou N.

8.2.5 Promovendo a Eficiência do Usuário

Tognazzini (2003) recomenda considerar sempre a *eficiência do usuário* em primeiro lugar, e não a do computador. As pessoas são mais custosas do que máquinas, e uma economia de tempo e esforço do usuário costumam trazer mais benefícios do que economias semelhantes de processamento ou armazenamento. Para isso, Tognazzini sugere *manter o usuário ocupado*. Toda vez que o usuário precisa esperar o sistema responder antes que possa continuar seu trabalho, há perda de produtividade e desperdício de dinheiro. Sendo assim, processamentos demorados não devem prender a interação, mas sim permitir que os usuários continuem seu trabalho com outras partes do sistema, deixando esses processos executando em *background*. Para isso, um bom projeto de arquitetura do software é essencial. Aliás, esse ponto reforça a necessidade de cooperação, comunicação e parceria constante entre os engenheiros e os designers de IHC durante todo o processo de desenvolvimento (veja a Seção 1.2).

Segundo Cooper (1999), o sistema deve ser *sensível ao que o usuário está fazendo* e não deve interrompê-lo desnecessariamente enquanto o usuário estiver trabalhando em algo. Por exemplo, mudar o estado de sistemas de comunicação (e.g., Windows® Live Messenger, Google® Talk) para ocupado enquanto o usuário estiver projetando uma apresentação.

O designer deve *proteger o trabalho dos usuários* (Tognazzini, 2003). Os usuários nunca devem perder o seu trabalho, seja por um erro seu, por uma falha na transmissão de rede, uma falha no fornecimento de energia para o computador ou qualquer outra razão.

O sistema deve se lembrar de tudo o que o usuário disse, para não perguntar de novo (Cooper, 1999), e se *manter informado sobre o usuário*. O sistema deve ser capaz de saber: se essa é a primeira vez em que o usuário acessou o sistema; onde o usuário está no sistema; para onde ele está indo (caso haja um caminho claro em direção a um objetivo); o que o usuário tem feito durante a sessão de uso atual; onde ele estava quando deixou o sistema na última sessão; e outras informações semelhantes que permitam poupar trabalho do usuário e melhorar sua experiência de uso do sistema. O usuário deve ser capaz de sair de um sistema numa máquina e acessá-lo a partir de outra, voltando exatamente ao ponto em que parou.

Para promover a eficiência de usuários frequentes, Nielsen (1993) e Schneiderman (1998) recomendam *fornecer atalhos e aceleradores*. À medida que a frequência de uso aumenta, aumenta também a vontade dos usuários de reduzir o número de

interações e acelerar o passo da interação. Teclas de atalho e comandos ocultos são bastante úteis a usuários experientes, e não prejudicam a interação dos usuários novatos. Exemplos de aceleradores são teclas de atalho para itens de menu (e.g., Ctrl+S para salvar) e acionamento de botões de comando em barras de ferramentas (e.g., botão para imprimir o documento atual utilizando a configuração default). Caso haja sequências de operações frequentes, o designer também pode fornecer meios mais sofisticados, como gravação de macro ou programação por demonstração (Cypher, 1993; Lieberman, 2001).

Para operações frequentes, o designer pode oferecer também a configuração de valores default, individualmente ou em grupo, formando perfis de execução dessas operações. A Figura 8.5 apresenta um diálogo de geração de arquivo PDF de um editor de apresentações que permite armazenar valores de parâmetros na forma de perfis de exportação.

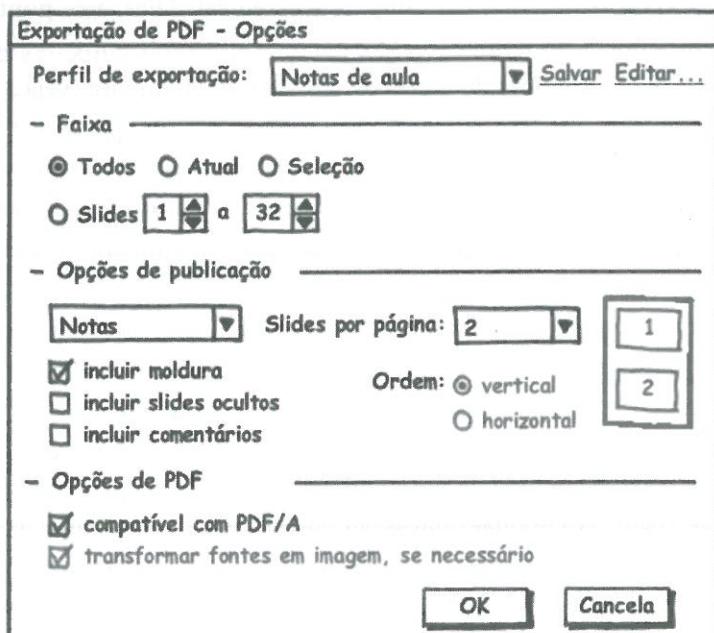


Figura 8.5 Diálogo com parâmetros configuráveis que podem ser armazenados em um perfil de exportação.

8.2.6 Antecipação

As aplicações devem tentar prever o que o usuário quer e precisa, em vez de esperar que os usuários busquem ou coletem informações ou invoquem ferramentas. O designer deve fornecer ao usuário todas as informações e ferramentas necessárias para cada passo do processo (Tognazzini, 2003).



Segundo Cooper (1999), o software deve *tomar iniciativa* e fornecer informações adicionais úteis, em vez de apenas responder precisamente a pergunta que o usuário tiver feito. Por exemplo, quando um usuário pergunta sobre o telefone de um restaurante, o software pode informar também seus dias e horários de funcionamento. Cooper acredita que, enquanto o software estiver à toa, deve *anticipar* e se preparar para situações que provavelmente acontecerão, de modo a responder mais rapidamente quando chegar a hora. Além disso, o software deve ser *observador* e se lembrar quais ações o usuário realiza em sequência, para tentar antever o próximo passo a cada momento e facilitar a sua execução.

Tognazzini destaca a importância de definir cuidadosamente os valores e a configuração padrão (defaults). Ele afirma que, de qualquer forma, os defaults devem ser facilmente substituídos por valores específicos mais adequados à situação atual. Se possível, campos contendo defaults devem vir já selecionados, de forma que os usuários possam editar seu conteúdo com novos valores rápida e facilmente. As pessoas tendem a aceitar os valores defaults para aquilo que elas não entendem ou assumem que o sistema já aprendeu sobre elas ou que seja a resposta “certa”. Por isso, a escolha deve ser cuidadosa. Considere cada alternativa apresentada na Figura 8.6. Ela é eficiente? É neutra? Ou induz a uma determinada opção?

<input type="checkbox"/> Não quero receber a newsletter semanal da Empresa <input checked="" type="checkbox"/> Quero receber a newsletter semanal da Empresa	Quer receber a newsletter semanal da Empresa? <input checked="" type="radio"/> sim <input type="radio"/> não
<input checked="" type="checkbox"/> Não quero receber a newsletter semanal da Empresa <input type="checkbox"/> Quero receber a newsletter semanal da Empresa	Quer receber a newsletter semanal da Empresa? <input checked="" type="radio"/> sim <input type="radio"/> não
<input checked="" type="checkbox"/> Não quero receber a newsletter semanal da Empresa <input type="checkbox"/> Quero receber a newsletter semanal da Empresa	Quer receber a newsletter semanal da Empresa? <input checked="" type="radio"/> sim <input type="radio"/> não
<small>* indica campo obrigatório</small>	

Figura 8.6 Ilustrações do uso (ou não uso) de valores default.

8.2.7 Visibilidade e Reconhecimento

Norman (1988) afirma que o designer deve *tornar as coisas visíveis: abreviar os golpos de execução e avaliação* (veja Seção 3.4). Antes de executar uma ação, é necessário tornar visível para os usuários o que é possível realizar e como as ações devem ser feitas. Para isso, a interface deve oferecer ações que correspondam a intenções do usuário.

Além disso, a interface não deve oferecer opções que não estejam disponíveis ou não façam sentido em um determinado momento da interação. Depois que o usuário realiza uma ação, a interface deve lhe fornecer indicações do estado do sistema que sejam prontamente percebidas e consistentes com o seu modelo mental, para que ele possa interpretá-las adequadamente e entender os efeitos da ação realizada.

Em outras palavras, o estado do sistema, os objetos, as ações e as opções devem estar atualizados e facilmente perceptíveis (Nielsen, 1993; Shneiderman, 1998; Tognazzini, 2003). O usuário não deve ter de se lembrar para que serve um elemento de interface cujo símbolo não é reconhecido diretamente. Também não deve ter de se lembrar de informações de uma parte da aplicação quando tiver passado para uma outra parte da aplicação. O sistema não deve exigir que o usuário memorize muitas informações ou comandos durante a interação, devido à limitação humana do processamento de informação na memória de curto prazo. As instruções de uso do sistema devem estar visíveis ou facilmente acessíveis sempre que necessário.

Os usuários também não devem ter de procurar informações sobre o estado do sistema. Em vez disso, eles devem ser capazes de olhar rapidamente para o seu ambiente e obter pelo menos uma primeira aproximação desse estado. Segundo Tognazzini, os *mecanismos de status* são essenciais para fornecer a informação necessária para os usuários responderem adequadamente às mudanças ocorridas. O usuário poderá ter dificuldades em controlar o sistema adequadamente se não possuir informações suficientes sobre o seu estado atual. Entretanto, Cooper (1999) recomenda que o software não exagere nas mensagens de status. Em geral, as informações de status podem ser bem sutis. Por exemplo, o ícone de caixa de entrada do cliente de correio eletrônico pode aparecer vazio, meio cheio ou lotado, para refletir o número de mensagens não lidas.

Quando o usuário realiza uma ação, o sistema deve mantê-lo informado sobre o que ocorreu ou está ocorrendo, através de feedback (resposta do sistema) adequado e no tempo certo (Nielsen, 1993; Shneiderman, 1998; Tognazzini, 2003). Para ações frequentes e com resultado esperado, a resposta pode ser sutil, mas para ações infreqüentes e com grandes consequências, a resposta deve ser mais substancial. A Figura 8.7 apresenta um feedback sutil como resultado de um cadastro bem-sucedido, e outro feedback, destacado, indicando uma falha.

Tognazzini (2003) afirma que, para reduzir a sensação de o programa não estar respondendo, o sistema deve: fornecer feedback visual/sonoro até 50 ms após um clique de botão; sinalizar que está ocupado (e.g., através de uma ampulheta, animada para indicar que o sistema não travou) quando o sistema realizar uma ação que leve entre 0,5 s e 2 s; apresentar, quando a ação levar mais do que 2 segundos, uma men-



sagem indicando a demora estimada e cada passo sendo realizado, juntamente com uma barra de progresso e opções de cancelamento, suspensão ou de execução em *background*. Quando uma operação demorada (mais do que 10 segundos) terminar, o sistema deve emitir um som e fornecer uma indicação visual destacada, para que usuários que tenham desviado sua atenção possam retomar o seu uso do sistema.

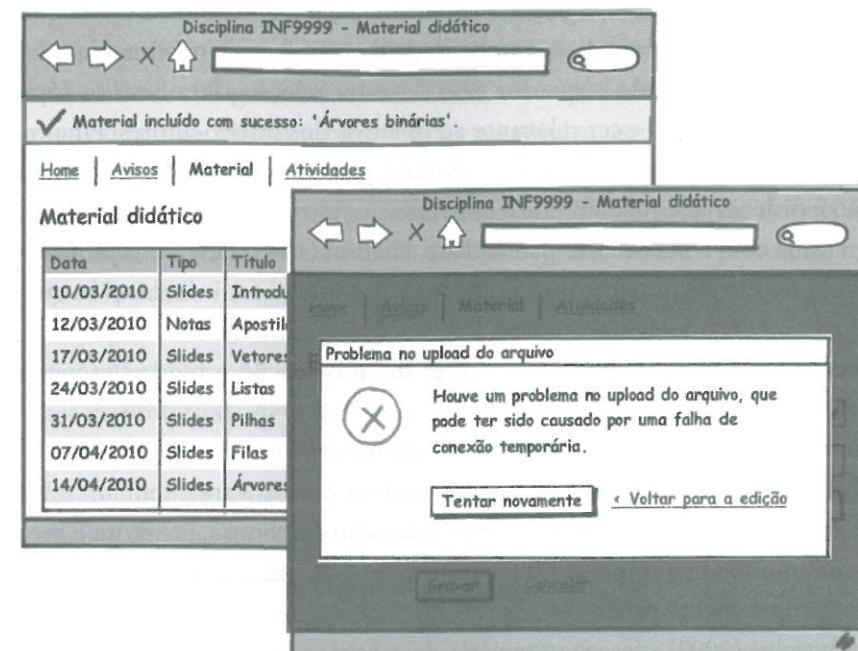


Figura 8.7 Feedback sutil e destacado.

O designer deve manter o usuário informado sobre o caminho que percorreu no sistema ou Web site até o ponto em que se encontra. O usuário não deve ser responsável por elaborar um mapa mental do que fez até o momento ou por onde passou no sistema. Além disso, sinalizações claras orientam a interação do usuário e lhe ajudam a navegar pela aplicação rapidamente, sempre cientes de onde estão (Tognazzini, 2003).

8.2.8 Conteúdo Relevante e Expressão Adequada

Segundo Reeves e Nass (1996), as pessoas dão tratamento humano para qualquer mídia ou tecnologia que apresente comportamento semelhante ao de uma pessoa, mesmo sabendo que isso é tolice e negando que tenham feito isso *a posteriori*. Em linha com o princípio de conversa cooperativa de Grice (1972), eles destacam que uma interação polida segue quatro máximas: qualidade, quantidade, relação (ou relevância) e modo (ou clareza).

A máxima da *qualidade* afirma que não devemos dizer nada que saibamos não ser verdade ou para o que não tenhamos evidências, ou seja, não devemos mentir ou especular. A máxima da *quantidade* diz respeito à quantidade de informação comunicada: a contribuição de uma fala deve ser tão informativa quanto necessário para os objetivos da conversa, e não mais. Uma constante dentre os profissionais de IHC é a busca pela simplicidade. Seguem o lema “menos é mais”. A máxima da *quantidade* está fortemente relacionada à simplicidade da interface. A máxima da *relação* ou *relevância* afirma que tudo o que for dito deve ter relação clara com os tópicos da conversa até o momento e ser relevante ao objetivo dos interlocutores. Finalmente, a máxima de *modo* ou *clareza* pede para evitar a prolixidade e ambiguidade, buscar a concisão e ordenar adequadamente a conversa.

Em linha com a máxima de *quantidade*, Nielsen (1993) defende o *projeto estético e minimalistico*. Ele afirma que os diálogos não devem conter informações que sejam irrelevantes ou raramente necessárias. Cada unidade extra de informação em um diálogo compete com as unidades relevantes de informação e reduz sua visibilidade relativa.

Cooper (1999) alerta que, quando o sistema sonega informação, ele obscurece seu comportamento. Para que os usuários gostem do software e tenham uma experiência agradável, ele sugere projetar uma interação respeitosa, generosa e prestativa. Ele considera uma boa interação mais importante do que a composição da interface concreta propriamente dita.

Tognazzini (2003) oferece uma série de recomendações relacionadas à *redação* em interfaces gráficas. As mensagens de instrução e ajuda devem ser concisas e informativas sobre problemas que ocorrerem. Os rótulos de menus e botões devem ser claros e livres de ambiguidade. Entretanto, nem sempre um termo mais preciso é melhor. Por exemplo, considere um editor de texto com os seguintes itens de menu: Inserir Quebra de Página, Acrescentar Nota de Rodapé e Construir Tabela. Embora precisos, eles trazem menos benefícios do que os itens: Inserir Quebra de Página, Inserir Nota de Rodapé e Inserir Tabela. Nessa segunda opção, os usuários conseguem varrer as opções disponíveis mais rapidamente, pois a variedade dos verbos utilizados aumenta o tempo que leva para decodificá-los, sem ser suficientemente informativa de modo a compensar o tempo despendido.

Além de cuidar do conteúdo, o designer deve se certificar de que o texto também seja *legível*. Para isso, deve ser apresentado com alto contraste e favorecer texto preto sobre fundo branco ou amarelo-claro, evitando fundos de cor cinza (Tognazzini, 2003). Os tamanhos de fonte devem ser suficientemente grandes para serem lidos em monitores de tamanho e resolução padrão. E os dados podem ser apresentados



em fonte maior ou com mais destaque do que rótulos e instruções, principalmente quando os dados forem numéricos. E, sempre que possível, o designer deve permitir que usuários com deficiências visuais aumentem o tamanho da fonte.

Ao utilizar cores para transmitir alguma informação através da interface, é necessário utilizar dicas secundárias claras para transmitir a mesma informação àqueles que não conseguem distinguir as cores utilizadas, seja por limitações físicas, como daltonismo, que afeta cerca de 10% da população masculina, ou por limitações do dispositivo utilizado (Tognazzini, 2003). Dicas secundárias incluem variações na escala em tons de cinza, diferentes ilustrações ou rótulos associados a cada cor apresentada.

Como visto na Seção 1.4, profissionais oriundos de diversas áreas devem colaborar em atividades de IHC. Dentre eles, os designers gráficos são os principais responsáveis pela identidade visual do sistema, incluindo o *layout*, que define a disposição espacial dos elementos de interface, e a escolha de fontes, formas, cores, texturas, imagens e outros símbolos não tipográficos (Mullet e Sano, 1995). Ao elaborar esboços de interface ou protótipos, deve-se buscar fazer uso de alguns princípios básicos, para que a atenção do usuário não se desvie para detalhes da interface que não sejam o foco do design naquele momento, mas que, por violarem princípios básicos de design gráfico, estejam lhe incomodando.

Mullet e Sano (1995) organizam princípios de design visual de interfaces em torno dos seguintes temas: elegância e simplicidade; escala, contraste e proporção; organização e estrutura visual; módulo e programa; imagem e representação; e estilo. Os esboços de interface se beneficiam principalmente dos princípios relacionados com a estrutura geral da interface, como, por exemplo, o uso de *grid* para alinhar os elementos e o uso de espaço para guiar a disposição de elementos conforme a direção natural de leitura dos usuários.

8.2.9 Projeto para Erros

Norman (1988) recomenda *projetar para o erro*, ou seja, assumir que qualquer erro potencial será cometido. O designer deve ajudar o usuário a se recuperar de um erro, informando-lhe sobre o que ocorreu, as consequências disso e como reverter os resultados indesejados. Como visto, os sistemas devem ser exploráveis, ou seja, deve ser fácil reverter as operações e difícil realizar ações irreversíveis. Além disso, Cooper (1999) recomenda não colocar controles de funções utilizadas com frequência adjacentes a controles perigosos ou que raramente são utilizados. Por exemplo, na Figura 8.8, um botão de inspeção de Propriedades está posicionado bem próximo ao botão

para Desabilitar a conexão de rede que, inclusive, efetua a operação sem pedir confirmação do usuário.

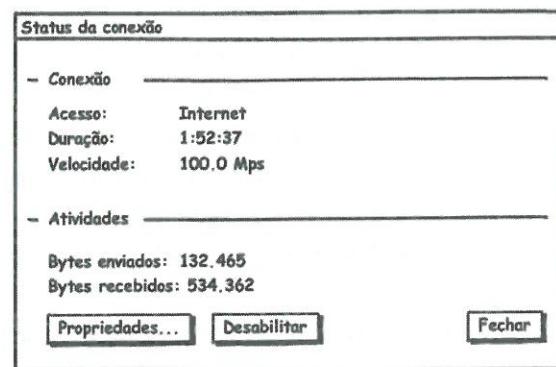


Figura 8.8 Posicionamento de um botão de uso frequente (Propriedades) próximo a um botão de uso infrequente e consequências graves (Desabilitar).

Nielsen (1993) e Shneiderman (1998) recomendam que o designer tente, em primeiro lugar, *evitar que os erros ocorram*, caso possível. Se um erro for cometido, o sistema deve ser capaz de detectá-lo e oferecer mecanismos simples e inteligíveis para tratá-lo. O designer deve *ajudar os usuários a reconhecerem, diagnosticarem e se recuperarem de erros* (Figura 8.9). As mensagens de erro devem ser expressas em linguagem simples e inteligível (sem códigos indecifráveis), indicar precisamente o problema e sugerir uma solução de forma construtiva.

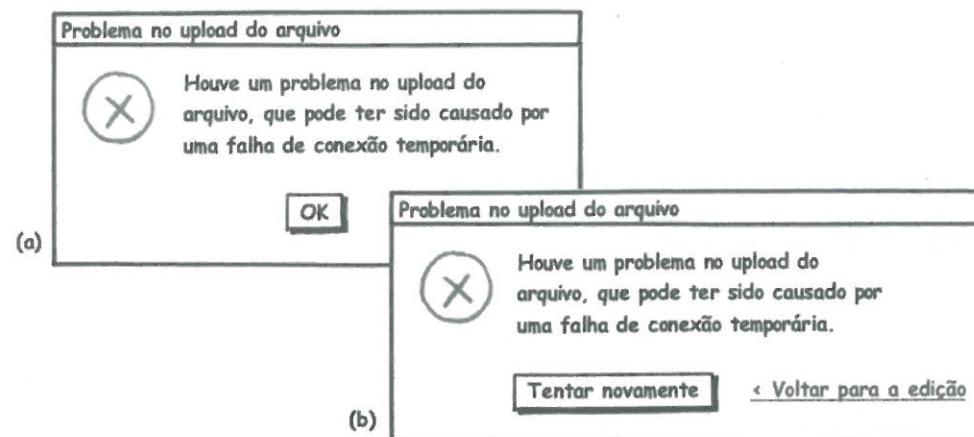


Figura 8.9 Exemplo de mensagens de erro (a) sem e (b) com oportunidade de recuperação e retomada de um caminho de interação produtivo.

Além de erros, também devemos apoiar os usuários a esclarecerem suas dúvidas durante a interação. Para isso, precisamos elaborar *ajuda e documentação* de alta qualidade. Tais informações devem ser facilmente encontradas, focadas na tarefa do usu-



ário, enumerar passos concretos a serem realizados e não ser muito extensas (veja Seção 7.5).

8.3 Padrões de Design de IHC

Padrões de design (*design patterns*) são descrições de melhores práticas num determinado domínio de design (Tidwell, 2006; Borchers, 2001). O conceito de padrões foi proposto na década de 1970 por Christopher Alexander (1979), no domínio de arquitetura e urbanismo, e apresentados à comunidade de Computação na década de 1990 na forma de padrões de arquitetura de software orientado a objetos utilizados na Engenharia de Software (Gamma *et al.*, 1995). Padrões capturam soluções comuns a certos interesses ou tensões de design (chamadas de “forças” na literatura de padrões). Alexander objetivava recriar o conhecimento compartilhado sobre soluções de design boas e adequadas tornando-o explícito, documentado, testado e gradualmente aperfeiçoado (Borchers, 2001).

Tidwell (2006) aponta como vantagens do uso de padrões, além da captura da sabedoria coletiva de designers experientes em IHC, o fornecimento de um vocabulário de design comum e divulgação de boas soluções para a comunidade de design. Ela alerta, no entanto, que padrões não são soluções prontas, nem regras ou heurísticas. Cada aplicação de um padrão difere ligeiramente uma da outra. Padrões de design em IHC também não são descrições passo a passo sobre como projetar uma interface inteira, mas sim descrições para problemas pontuais. O uso de padrões não substitui o processo criativo envolvido num projeto de design, nem assegura por si só a qualidade do produto final. Em outras palavras, padrões devem ser utilizados como recursos em um processo de design reflexivo (veja Seção 4.2), e não como soluções prontas.

Os padrões não são isolados; eles estão relacionados com outros padrões de diversas maneiras. Cada padrão pode ser descrito em maior ou menor nível de detalhes e pode ser adequado a apenas um certo contexto (Borchers, 2001). Um conjunto completo de padrões de design relacionados é chamado de linguagem de padrões (*pattern language*), que definem um extenso vocabulário de elementos utilizados em design, além de formas de articulá-los.

Borchers (2001) utiliza uma estrutura semelhante à de Alexander e sugere que os padrões sejam descritos através dos seguintes elementos:

- o *nome* do padrão, para transmitir em poucas palavras a ideia do padrão, de modo a facilitar sua memorização e a menção a ele em uma reflexão ou discussão durante o design;

- uma *avaliação* de sua validade (indicada por zero, um ou dois asteriscos), indicando o grau de confiança que os autores tinham no padrão;
- uma *imagem* como exemplo de aplicação do padrão;
- o *contexto* em que o padrão pode ser usado, com referência a padrões “maiores” que este ajuda a implementar;
- uma *breve descrição do problema*, um resumo da situação geral que o padrão endereça;
- uma *descrição detalhada do problema*, com base empírica e citando as forças conflitantes que o padrão almeja resolver ou equilibrar;
- a *solução central* do padrão, um conjunto claro mas relativamente genérico de instruções que possam ser aplicadas a diversas situações;
- um *diagrama* ilustrando a solução, geralmente um esboço gráfico da solução e seus principais constituintes;
- *referências a padrões “menores”*, recomendações do autor sobre como implementar e desdobrar ainda mais a solução representada no padrão atual.

Tidwell (2006), por sua vez, descreve seus padrões utilizando, além do título do padrão (e.g., “painéis colapsáveis”), as seguintes informações:

- *o que*, resumindo a solução em um parágrafo curto;
- *usar quando*, indicando as situações em que o padrão se aplica e as forças em atuação;
- *por que*, fornecendo dados que justificam a adequação do padrão à situação descrita;
- *como*, detalhando a solução e as formas como ela pode ser implementada;
- *exemplos*, incluindo diagramas e imagens de interfaces concretas que ilustram o uso do padrão.

Van Welie⁴ representa padrões de forma semelhante a Tidwell, através dos seguintes elementos: *problema*, *solução* (texto e imagem ilustrando uso real), *usar quando*, *como*, *por que*, *mais exemplos*, *implementação* e *literatura*.

O Exemplo 8.1 apresenta um padrão de design comumente descrito: o padrão de Acordeão, adaptado da biblioteca de van Welie. Os textos em itálico representam ligações com outros padrões, sejam da mesma biblioteca ou de outras bibliotecas, como a biblioteca de Tidwell.⁵

⁴ <http://www.welie.com/patterns/index.php>.

⁵ <http://designinginterfaces.com/>.



Exemplo 8.1 – Padrão de Design de Interface

Título: Acordeão

Problema: O usuário precisa encontrar um item dentre as opções de navegação.

Solução: Empilhar painéis vertical ou horizontalmente, abrindo um painel de cada vez, enquanto colapsa os demais.

Usar quando: Como mecanismo de navegação, sendo conceitualmente equivalente a *Guias* e uma solução alternativa a *Árvores de navegação*. Embora seja utilizado como parte de um *Assistente*, isso não é recomendado, pois apresenta pior qualidade de uso do que implementações tradicionais. Pode ser uma boa forma de implementar uma seção de *Perguntas Frequentes (FAQ)*, em que cada pergunta é aberta de uma vez. Um outro uso seria para gerenciar configurações de preferências. O número de painéis deve ser reduzido, em geral menor que dez.

Exemplo:

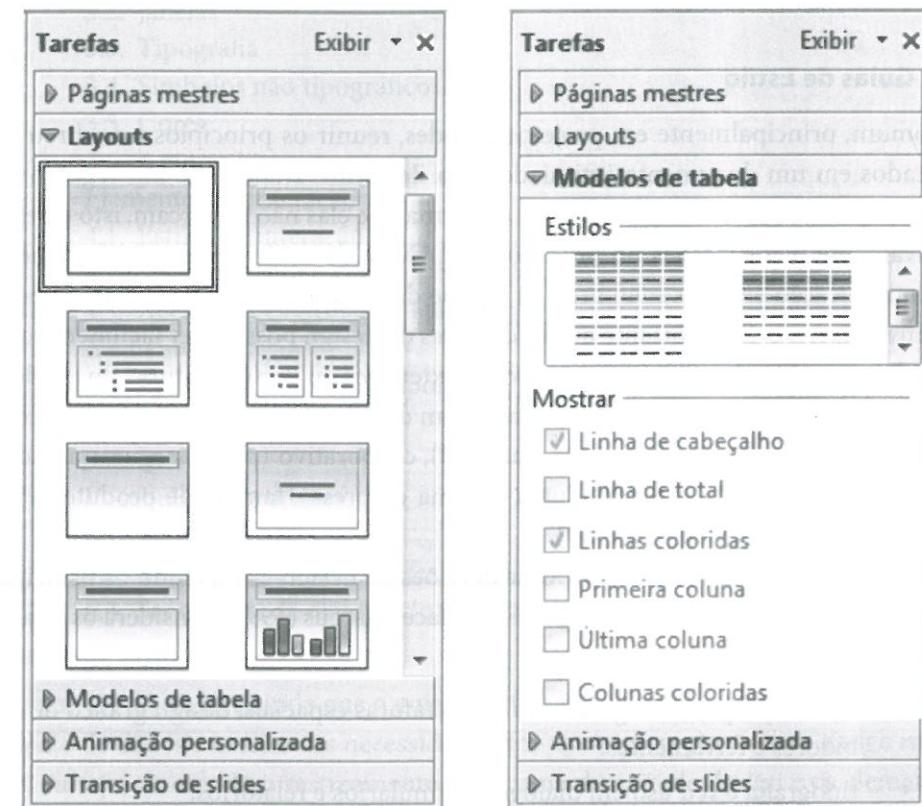


Figura 8.10 Exemplo do padrão de interface Acordeão, com o painel *Layouts* expandido (esq.) e com o painel *Modelos de tabela* expandido (dir.).

Como: Os painéis podem ser dispostos vertical ou horizontalmente. Apenas um painel é mantido aberto de cada vez. Quando mais do que um painel pode ser mantido aberto, trata-se do padrão *Painéis Colapsáveis*. Em geral, painéis verticais são destinados a submenus, enquanto painéis hori-

zontais revelam grandes áreas de conteúdo. Os seguintes cuidados devem ser tomados na implementação do padrão Acordeão:

- Anime a abertura dos painéis para fornecer aos usuários *feedback* sobre o que está acontecendo. A animação deve ser sutil e durar no máximo 250 ms.
- Permita que a navegação seja feita através das setas do teclado.
- Destaque o painel atual para que o usuário possa facilmente diferenciar o cabeçalho do painel aberto dos cabeçalhos dos painéis fechados.
- Certifique-se de que o tamanho total do Acordeão pode aumentar ou diminuir para acomodar o conteúdo adequadamente.

Por quê: Um acordeão é útil para comprimir muitos elementos num espaço de tela compacto. Os elementos podem ser propriedades, perguntas ou simples itens de navegação. A desvantagem óbvia é que os elementos dos outros painéis ficam ocultos.

8.4 Guias de Estilo

É comum, principalmente em projetos grandes, reunir os princípios e as diretrizes adotados em um documento intitulado **guia de estilo**. Trata-se de um registro das principais decisões de design tomadas, de forma que elas não se percam, isto é, sejam efetivamente incorporadas no produto final. Guias de estilo servem de ferramenta de comunicação entre os membros da equipe de design e também com a equipe de desenvolvimento. É importante que as decisões de design possam ser facilmente consultadas e reutilizadas nas discussões sobre extensões ou versões futuras do produto.

Um guia de estilo pode ser elaborado com diferentes escopos: plataforma (composição de dispositivo e sistema operacional), corporativo (para assegurar a padronização e consistência entre produtos de uma empresa), família de produtos e um produto específico (Mayhew, 1999).

Um guia de estilo deve incorporar decisões de design envolvendo os principais elementos e considerações de design de interface. Marcus (1992) considera os seguintes elementos:

- *layout*: proporção e grids; uso de metáforas espaciais; design gráfico de exibidores e ferramentas;
- tipografia e seu uso em diálogos, formulários e relatórios;
- simbolismo: clareza e consistência no design de ícones;
- cores: os dez mandamentos sobre o uso de cores;
- visualização de informação: design de gráficos, diagramas e mapas;
- design de telas e elementos de interface (*widgets*).



Uma estrutura comum de guia de estilo é a seguinte (Marcus, 1992; Mayhew, 1999):

1. Introdução
 - 1.1. Objetivo do guia de estilo
 - 1.2. Organização e conteúdo do guia de estilo
 - 1.3. Público-alvo do guia de estilos (programadores, gerentes, equipe de suporte)
 - 1.4. Como utilizar o guia (em produção e manutenção)
 - 1.5. Como manter o guia
2. Resultados de análise
 - 2.1. Descrição do ambiente de trabalho do usuário
3. Elementos de interface
 - 3.1. Disposição espacial e *grid*
 - 3.2. Janelas
 - 3.3. Tipografia
 - 3.4. Símbolos não tipográficos
 - 3.5. Cores
 - 3.6. Animações
4. Elementos de interação
 - 4.1. Estilos de interação
 - 4.2. Seleção de um estilo
 - 4.3. Aceleradores (teclas de atalho)
5. Elementos de ação
 - 5.1. Preenchimento de campos
 - 5.2. Seleção
 - 5.3. Ativação
6. Vocabulário e padrões
 - 6.1. Terminologia
 - 6.2. Tipos de tela (para tarefas comuns)
 - 6.3. Sequências de diálogos (e.g., para *feedback* ou confirmação de uma operação)

Mayhew (1999) sugere ainda que o guia de estilo inclua todos os produtos do levantamento de dados e análise das necessidades dos usuários, registrando o *design rationale*, ou seja, mantendo o rastreamento entre uma decisão de design e os elementos de discussão que culminaram naquela decisão.

Quando elaboramos um guia de estilo, não basta simplesmente produzi-lo. Deveremos comunicar adequadamente sua existência e importância para os demais designers e desenvolvedores, oferecer treinamento, facilitar o acesso ao documento como um todo ou a um tópico específico e investir em uma mudança na cultura de design

e desenvolvimento da equipe. Além disso, não devemos tratar o guia de estilo como um conjunto de regras, mas sim uma ferramenta prática de apoio ao trabalho e à criatividade. Em outras palavras, um guia de estilo deve ser utilizado como parte de um *processo reflexivo de design*, e não como um conjunto de soluções prontas ou fórmulas geradoras de soluções (veja Seção 4.2).

Atividades

Para as atividades abaixo, considere a agenda pessoal integrada com lista de afazeres e correio eletrônico utilizada no capítulo anterior, projetada para uso em dois dispositivos: um computador desktop e um *smartphone*. Retomando as atividades do capítulo anterior, examine os esboços produzidos.

1. *Uso de princípios e diretrizes.* Quais princípios e diretrizes foram aplicados nos esboços? Algum princípio ou diretriz foi violado? Como pode ser resolvido? Caso não seja resolvido, quais as possíveis consequências da violação?
2. *Uso de padrões.* Examine as bibliotecas de padrões citadas neste capítulo. Quais padrões se encaixam para a solução sendo projetada? Que adaptações precisam ser feitas?
3. *Guia de estilo.* A partir dos modelos e esboços elaborados, destaque os elementos que podem compor um guia de estilo para futuras funcionalidades do mesmo sistema ou para sistemas complementares produzidos para o mesmo cliente.

9

Planejamento da Avaliação de IHC

Objetivos do Capítulo

- Discutir a importância de avaliar a qualidade de uso de um sistema interativo.
- Descrever o planejamento e a execução da avaliação de IHC envolvendo ou não usuários.
- Caracterizar os objetivos de avaliação e contextos de projeto que auxiliam na escolha do método de avaliação a ser utilizado.