Princípios de design

“As pessoas ignoram designs que ignoram pessoas.” - Frank Chimero

Estudar o trabalho de outras pessoas pode inspirar nossa criatividade, porém estudar e entender os princípios de design nos protegerá de cometer erros. Os princípios de design correspondem às leis científicas do mundo da usabilidade, muito semelhantes às leis da gravidade e da relatividade no mundo da física. Os princípios de design são relativamente constantes e foram concebidos ao longo de vários anos a partir do estudo da cognição e do comportamento humano. Eles nos ajudam a oferecer diretrizes baseadas na compreensão do ser humano e na interpretação do mundo que o cerca.

Ter um bom domínio dos princípios de design e de modelos previsíveis pode ajudar você a criticar eficientemente o seu trabalho. É a linguagem perfeita para expressar o que está correto ou o que está errado em seus usuários, que, com frequência, têm dificuldades em expressar suas necessidades. Ajude-os a encontrar a terminologia correta explicando o significado dos vários princípios de design. Isso proporcionará a ambos uma linguagem comum e correta com a qual trabalhar.

É impossível descrever todos os princípios de usabilidade neste livro, pois muitos deles estão além do escopo de nossa discussão. Considere que essa será uma visão geral de alguns dos princípios mais comuns.

Princípio da proximidade (Princípio da Gestald)

O princípio da proximidade é um dos muitos princípios definidos nos *Princípios de percepção da Gestald.* Embora você deva estudar todos os princípios da Gestald, acho que o *princípio da proximidade* tem o impacto potencial mais significativo sobre seus aplicativos e exige o menor volume de esforços.

O princípio estabelece que nós percebemos relacionamentos entre objetos que estão mais próximos. Inversamente, objetos que estão mais distantes, aparentemente, teriam menos relação.

Em virtude desse fato, você poderá ver o princípio de proximidade sendo referenciado como princípio de agrupamento. Basicamente, é mais fácil ver padrões de operação quando os itens estão agrupados de acordo com suas funções, conforme mostrado na figura 7.1.

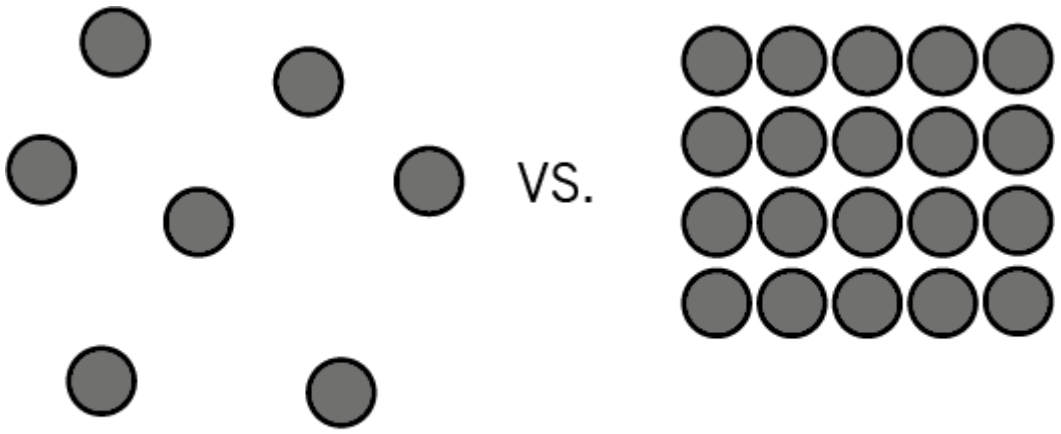

Figura 7.1 - Um exemplo do princípio de proximidade, com o grupo à direita parecendo estar relacionado.

É por isso que o princípio da proximidade pode exercer o impacto mais significativo. Ao simplesmente organizar e agrupar itens de uma maneira que a sua função seja descrita. você poderá melhorar significativamente a experiência de usuário com seu aplicativo. Um layout organizado faz com que seja mais fácil aprender a usar seu aplicativo e exige menos do usuário para que ele possa encontrar os itens. Muitos desenvolvedores ignoram esse princípio simples porque não reservam tempo para refletir sobre como seus aplicativos deveriam estar organizados. Eles acham que o layout de seus aplicativos faz sentido; enquanto isso, seus usuários ficam confusos e frustrados. O princípio da proximidade pode ser usado como um poderoso indicador de que determinados recursos devem estar juntos.

Considere o pacote de aplicativos Office da Microsoft. Em 2007, a Microsoft introduziu a interface Ribbon, que é um agrupamento de funções do Office na parte superior da janela de um aplicativo. Essa interface resultou da confusão cada vez mais crescente que muitos usuários sentiam em relação à localização de determinados recursos do Office. A Microsoft introduziu o Ribbon como uma maneira de colocar funções semelhantes em uma relação de proximidade umas com as outras.

Por exemplo, no Ribbon do Word, apresentado na figura 7.2, as funções que alteram o estilo do texto são colocadas bem próximas, assim como as funções que manipulam as imagens, as funções que alteram o layout do documento, e assim por diante. Além do mais, a Microsoft fez o Ribbon ser contextual, de modo que ele se altera conforme o item selecionado no documento. Isso ajuda os usuários ao destacar os recursos mais importantes, de acordo com o conteúdo que estão manipulando.

Figura 7.2 — Interface Ribbon do Microsoft Word.

Nada é mais frustrante do que um aplicativo desorganizado. Isso exige que o usuário percorra menus complexos e opções, procurando uma agulha virtual em um palheiro. Esse processo reduz nossa eficiência e acaba com nossa paciência. Organizar um aplicativo por proximidade ajuda os usuários a entender como seu aplicativo funciona e permite que eles avaliem rapidamente as opções disponíveis.

Visibilidade, feedback visual e proeminência visual

A visibilidade corresponde a tudo o que você utilizar para concentrar o foco visual em um elemento ou uma ação na interface de usuário de seu aplicativo. Há diversas maneiras de fazer isso:

Tipo de letra

Diferentes estilos e tamanhos de texto podem atrair a atenção do usuário.

Opacidade

Ajustar a opacidade de um item ajuda a reduzir ou aumentar sua visibilidade.

Proeminência

Elemento que são maiores que outros terão mais visibilidade, conforme mostrado na figura 7.3 .

Status

Indica que o aplicativo está processando uma solicitação ou que recebeu dados de entrada do usuário.

Cor/Contraste

Tradicionalmente, itens com cores mais fortes ou contrastantes atrairão mais atenção.

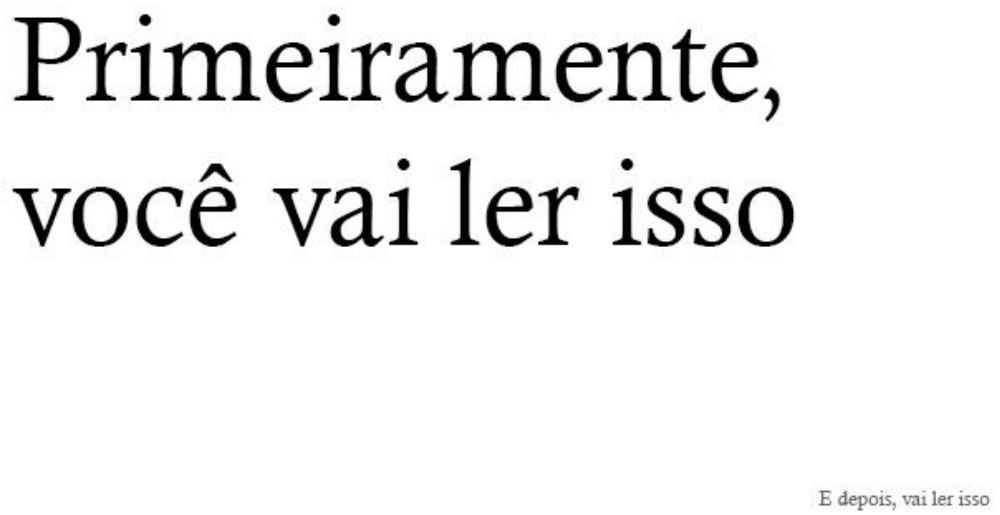

Figura 7.3 - Exemplo de proeminência, um dos princípios de visibilidade.

O princípio da visibilidade pode ser usado para indicar o status de um aplicativo. Por exemplo, o cliente de mensagens que uso no trabalho possui um ícone que se torna cinza quando não estou mais logado, como mostrado na figura 7.4. Quando faço login, ele se torna verde. Essa pequena mudança ajuda a me manter informado sobre meu *status* durante a utilização do serviço

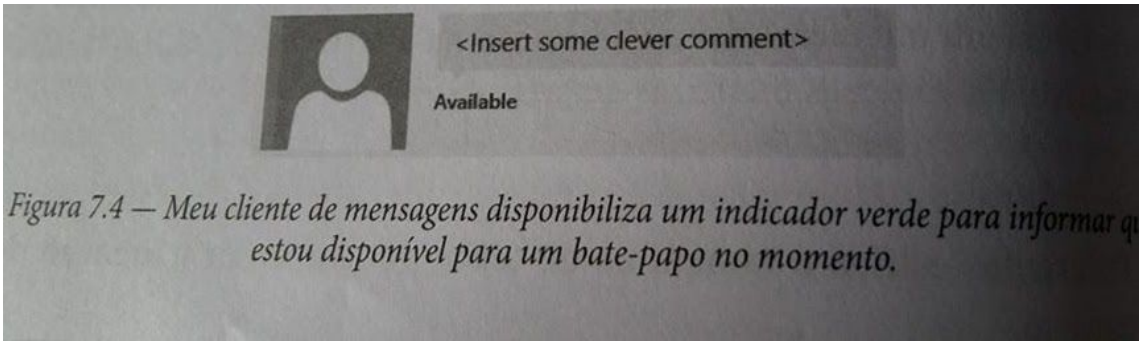

Figura 7.4 — Meu cliente de mensagens disponibiliza um indicador verde para informar q estou disponível para um bate-papo no momento.

Outro aspecto do princípio de visibilidade é proporcionar um *feedback* visual. O princípio do *feedback* visual estabelece que os aplicativos devem responder à entrada de dados do usuário. Em outras palavras, seu aplicativo deve apresentar algum tipo de indicação de que recebeu informações do usuário. Um exemplo simples desse caso seria disponibilizar um ícone giratório ou uma mensagem de “procurando...” quando um usuário submeter uma solicitação de busca. A questão geral do princípio de *feedback* visual é notificar o usuário de que houve uma interação. Sem essa confirmação, o usuário fica confuso a respeito de a sua ação ter sido recebida ou não pelo aplicativo.

A maioria dos aplicativos fornece *feedback* ao usuário. Contudo, já vi algumas implementações precárias desse recurso. Por exemplo, a figura 7.5 mostra como o

sistema de catálogo de cursos de minha universidade responde quando procuro uma determinada disciplina.

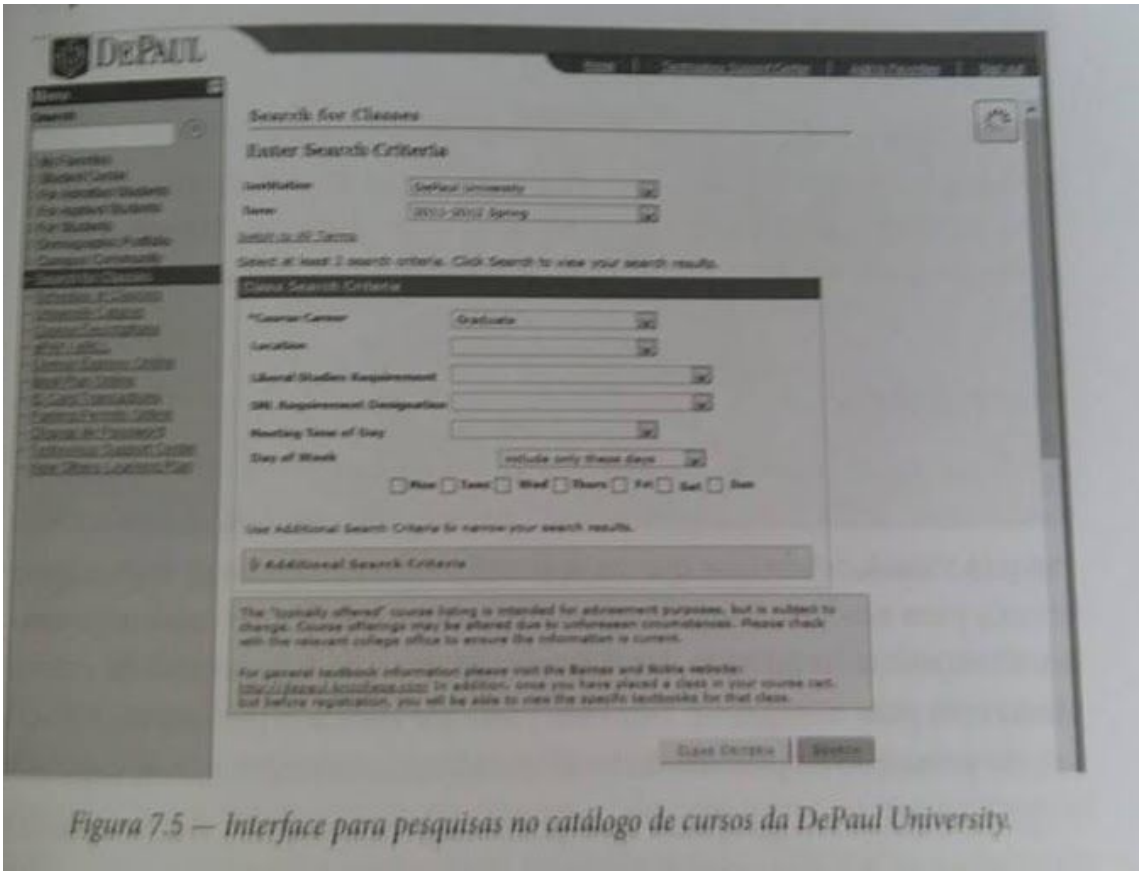

Figura 7.5 — Interface para pesquisas no catálogo de cursos da DePaul University.

No primeiro dia de matrícula, estava convencido de que a ferramenta de buscas não estava funcionando porque eu submetia uma solicitação de busca e nada acontecia. Alguns minutos se passaram até eu perceber o ícone giratório no canto superior direito. Nesse caso, o sistema estava fornecendo *feedback* visual, mas, como esse estava posicionado de maneira prec´ria, eu não havia notado.

Além do mais, em virtude do alto tráfego causado por todos se matriculando ao mesmo tempo, o sistema estava lento para responder. Isso significa que o *feedback* visual seria mais importante ainda porque as consultas estavam consumindo mais tempo que o normal. Um posicionamento mais adequado do ícone giratório seria mais próximo ao botão de pesquisas, local em que meus olhos estavam focados quando fiz minha solicitação. Com esse posicionamento mais apropriado, eu teria clicado no botão de pesquisas e visto imediatamente minha solicitação sendo processada.

Problemas com visibilidade e *feedback* visual adequado são os mais comuns de usabilidade que vejo nos aplicativos. Sempre que ouço um usuário reclamar dizendo que uma interface é confusa ou difícil de ser compreendida, começo a examinar maneiras pelas quais eu poderia estar violando os princípios de visibilidade.

Seu aplicativo deve fornecer mensagens apropriadas de *status* continuamente. Nunca permita que seu usuário questione se o seu aplicativo continua funcionando. Se o seu aplicativo exigir algum tempo de processamento para atender a uma solicitação, informe isso ao usuário.

Hierarquia

Ao desenvolver sistemas mais complexos, pode ficar mais difícil organizar todos os recursos de seu aplicativo. O princípio da hierarquia, ou da hierarquia visual, estabelece que os aplicativos devem fornecer indicadores visuais para ajudar o usuário a perceber como o aplicativo está organizado. Com muita frequência, isso assume a forma de submenus e de outros elementos para navegação. Também pode ser aplicado por intermédio do uso do princípio de proximidade, discutido anteriormente neste capítulo.

Já trabalhei em projeto que possuíam hierarquias incrivelmente complicadas. A parte mais difícil para o desenvolvedor é tentar organizar seu aplicativo de uma maneira que faça sentido. Você pode gastar horas tentando determinar o local ao qual um recurso particular pertence ou como ele deveria se chamar.

Uma ferramenta que tem sido de valor inestimável para esses tipos de desafios é o *diagrama de afinidade,* que consiste no processo de posicionar os recursos de seu aplicativo (tipicamente usando notas adesivas, do tipo “post-it”) e organizá-los em grupos que façam sentido.

Gosto de usar notas adesivas de cores fortes, como mostrado na figura 7.6, porque tornam meu agrupamento mais visual. Também uso marcadores para desenhar pontos na notas e indicar outros aspectos que desejo ver. As cores dos marcadores e das notas adesivas possibilitam perceber padrões rapidamente, e o adesivo das notas permite que eu tente obter diferentes organizações.

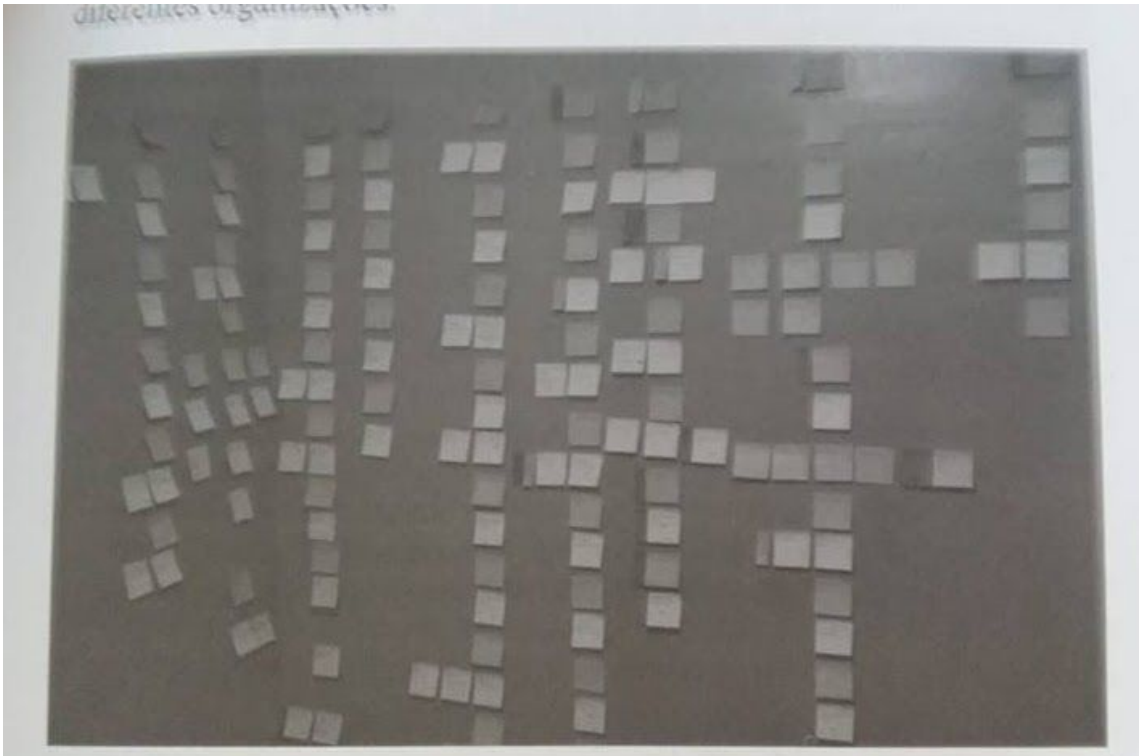

Figura 7.6 — Um exemplo de um diagrama de afinidades usando notas adesivas coloridas.

Ao organizar o portal de nossa empresa, esse tipo de diagramação foi útil para observar todos os recursos que queríamos que estivessem disponíveis aos usuários. Havia centenas de seções, políticas, aplicativos e sites. Com o diagrama de afinidades, o desafio torna-se mais plausível e ele proporciona uma maneira de tentarmos obter diferentes padrões de organização rapidamente.

Modelos mentais e metáforas

Independentemente de percebermos ou não, quando nos deparamos com um novo aplicativo ou produto, aplicamos nossos conhecimentos provenientes de outros produtos para entender como ele poderia funcionar. Com efeito, nossas experiências anteriores moldam nossa compreensão acerca de como o mundo funciona.

Por exemplo, as funções de computador Cortar e Colar baseiam-se em nossa familiaridade com cortar papéis em pedaços e colá-los. Com efeito, a maioria de nós sabe como uma tesoura funciona. Se não tivéssemos usado o recurso de Cortar em um aplicativo antes, poderíamos ver o ícone com a tesoura e fazer uma suposição segura acerca de seu propósito.

Porém o que acontece quando nosso modelo mental nos engana?

No livro *O design do dia-a-dia* (Rocco), Donald Norman explica o desafio representado pelos modelos mentais causadores de equívocos, ao descrever eletrodomésticos:

> Aquecedores domésticos, aparelhos de ar-condicionado e até a maioria dos fornos domésticos possuem somente dois níveis de operação: capacidade máxima ou desligado. Sendo assim, eles estão sempre aquecendo ou resfriando para atingir a temperatura desejada o mais rapidamente possível. Nesses casos, ajustar o termostato a uma temperatura muito elevada não faz nada além de causar desperdício de energia quando a temperatura exceder o alvo.

Aqui está um exemplo: uma mulher se registra em um hotel e encontra seu quarto insuportavelmente quente. Ela caminha até o controle do ar-condicionado, e ele está marcando uma temperatura sufocante de 30ºC! Em sua tentativa desesperada de se refrescar, ela pressiona a seta para baixo no controle até atingir a temperatura mínima de 10ºC. Seu modelo mental de ar-condicionado está incorreto. Ela acredita que, ao configurar o controle para o valor mínimo de temperatura, ela fará o quarto atingir temperatura desejada mais rapidamente. Na verdade, o ar-condicionado pode refrigerar somente a uma taxa fixa - geralmente alta ou baixa. Ao configurar o ar-condicionado com o valor mínimo de temperatura, ela somente garantiu que ficará muito mais frio do que ela desejava.

Como desenvolvedores, devemos ter ciência dos modelos mentais que os usuários estão utilizando em nossos aplicativos. Os ícones e a linguagem devem representar, com precisão, o funcionamento de nossos aplicativos. Já vi muitos desenvolvedores escolherem ícones inadequados para os aplicativos. Sem perceber, eles deduziram um determinado propósito, e quando os usuários clicam no ícone, ficam confusos com o resultado.

Por exemplo, se você estivesse desenvolvendo um site para viagens, poderia pensar que ter um coco como o botão usado para pesquisar seria bonito e divertido.

Infelizmente, os usuários não possuem um modelo conceitual de como um coco se aplica. Uma lente de aumento tem uma relação mais próxima com pesquisar algo. Isso ocorre porque muitos de nós sabemos que, no mundo real, lentes de aumento são usadas para procurar textos minuciosamente em livros e em periódicos. Embora eu o estimule a desafiar a sabedoria convencional e buscar inovações, alguns modelos devem ser deixados intactos.

Outro modelo mental interessante está na noção de Salvar nos aplicativos. Alguns de nós estamos familiarizados com o disquete tradicional como sendo o ícone para salvar arquivos em um computador. Esse modelo teve origem nos computadores mais antigos que utilizavam drives de disquetes de 3.5 polegadas para salvar documentos.

Como você acha que poderíamos melhorar essas metáforas? Realmente, isto é interessante!

Revelação progressiva

A revelação progressiva é uma ótima maneira de ajudar os usuários a entender quais recursos estão disponíveis a eles em seu aplicativo. Ao simplesmente ocultar opções que não são possíveis, você poderá reduzir a carga cognitiva dos usuários e orientá-los de maneira mais eficiente durante suas tarefas. O princípio da revelação progressiva é muito fácil de ser empregado e é especialmente útil em aplicativos mais complexos com menus carregados de recursos.

Por exemplo, o Adobe Photoshop, um software profissional para edição de fotografias, está cheio de recursos e de ferramentas para designers. Se a Adobe não incorporasse a revelação progressiva, todos os seus recursos apareceriam como disponíveis, independentemente do que eu estivesse fazendo no aplicativo. Isso representaria um fardo significativo para mim, à medida que tentasse descobrir o que é e o que não é possível fazer. Em vez disso, a Adobe deixa em cinza e desabilita os itens que não são aplicáveis à minha situação corrente, conforme pode ser visto na figura 7.7. Esse indicador sutil proporciona uma ajuda para navegar por entre as várias possibilidades do Photoshop.

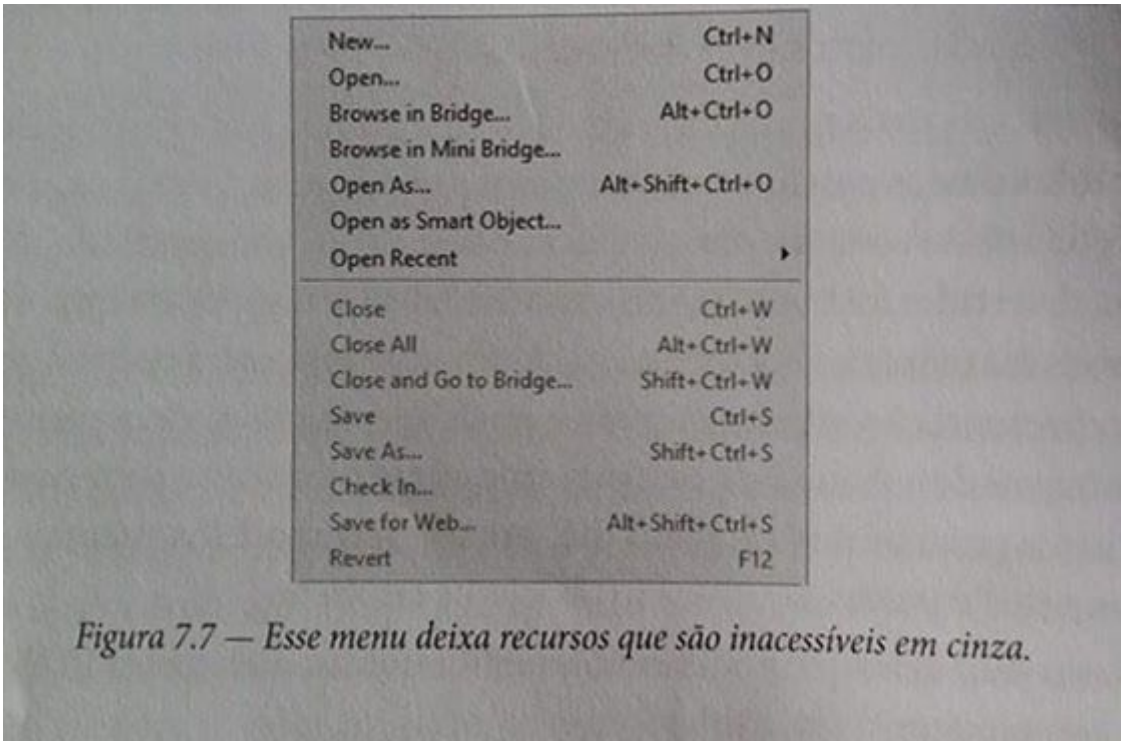

Figura 7.7 — Esse menu deixa recursos que são inacessíveis em cinza.

Consistência

O princípio da consistência pode parecer óbvio, mas vejo-o ser ignorado por muitos desenvolvedores. Esse princípio estabelece que os usuários aprendem e compreendem aplicativos com mais facilidade quando eles são consistentes com aquilo que já é conhecido. Já vi desenvolvedores introduzirem novos métodos para a execução de tarefas que já estão bem consolidadas.

Por exemplo, lembro-me de um aplicativo que exigia a criação de uma visualização prévia de meu documento antes de imprimi-lo. Em todo aplicativo que eu havia usado até então, nunca fui obrigado a criar uma visualização prévia de um documento antes de imprimi-lo. Os desenvolvedores podem ter tido boas intenções para criar esse fluxo de tarefas. Talvez achassem que seria melhor que eu fizesse uma visualização prévia de meu documento e, desse modo, seria menos provável que imprimisse algo que não quisesse. No entanto esse design não estava consistente em relação aquilo que eu já conhecia. Posso apreciar o fato de os desenvolvedores tentarem melhorar o processo de impressão, mas, nesse caso, o ato de criar uma visualização prévia antes de imprimir não era óbvio para mim e gerou uma confusão desnecessária.

Mais uma vez, sempre tento procurar maneiras novas de executar tarefas em um aplicativo. No entanto você deve ser criterioso na introdução de novos fluxos de tarefas que possam ser inconsistentes com o entendimento geral. E se você introduzir algo novo, tenha certeza de que será melhor do que aquilo que já conhecemos!

Um exemplo disso é a maneira pela qual a FiftyThree implementou a ação Undo(Desfazer) no Paper para iPad.

O usuário posiciona dois dedos na tela e começa a movê-los no sentido anti-horário (Undo, ou Desfazer) ou horário (Redo, ou Refazer). Outros aplicativos

disponibilizam um botão para Undo. No entanto a FiftyThree decidiu que um botão não se enquadrava em sua visão para o Paper. A empresa acreditava que procurar e pressionar um botão seria desnecessário e interromperia o fluxo criativo dos usuários.

Desse modo, a FiftyThree decidiu que iria melhorar o processo de Undo. Para isso, ela teve de pensar em novas maneiras de implementá-lo. A FiftyThree estudou outros mercados para obter novas ideias sobre como seria possível desfazer um trabalho. Foi assim que ela descobriu como cineastas desfaziam seu trabalho, como explica Petschnigg:

> Nosso designer de interação Andrew Allen - que é cineasta - trabalhou bastante com dials de jog[1] em videocassetes. E, para ele, a ideia era: “Precisamos de retrocesso. Não precisamos de um Undo, precisamos de Rewind(Retrocesso)!”.
>
> E foi meio daí que o gesto surgiu. E isso realmente se enquadra em nossa maneira de pensar - de certo modo - como seria a criação em dispositivos móveis. Como um app deveria funcionar quando você está em trânsito? Da forma como mantemos as pessoas em suas tarefas, em vez de fazê-las acionar um menu e encontrar um pequeno botão… parece fluir mais naturalmente.

Os desenvolvedores da FiftyThree poderiam simplesmente ter dado uma olhada no que seus concorrentes estavam fazendo e assumir que um botão tradicional de Undo seria o padrão que todos os usuários iriam entender. Poderiam ter sido conservadores e ter economizado tempo ao deixar de avaliar suas noções preconcebidas sobre como o Undo deveria funcionar.

Em vez disso, a equipe focou em seu objetivo de criar um aplicativo que estimulasse a criatividade. Da maneira como viam, o paradigma do Undo representava uma ameaça à sua missão. Portanto, eles conduziram intensas discussões sobre o que algumas pessoas teriam considerado uma função trivial. Eles procuraram entender como a maneira atual de desfazer uma tarefa não estava articulada, ao exigir a descoberta e o uso de menus e de botões. Exploraram outros mercados e tentaram descobrir como eles lidavam com a manipulação de trabalhos criativos. Ao longo de todo esse processo, eles chegaram a uma percepção inacreditável: não precisávamos de Undo; precisávamos de Rewind.

Desse modo, assim como acontece em várias situações, não há uma resposta fácil. Consistência em seus aplicativos é crucial porque reduz o fardo cognitivo de seus usuários e deixa-os mais à vontade para aprender funções básicas porque o desenvolvedor achou que seria melhor fazer as tarefas de maneira diferente.

Às vezes é bom quando um aplicativo se comporta conforme esperado, quando um item de menu está exatamente onde você espera que ele esteja ou uma ação produz o resultado adequado. Outras vezes, como no caso de desfazer tarefas no Paper, é agradável ser surpreendido e experimentar algo diferente.

Balancear a consistência em seu design pode ser desafiador, mas, se isso for feito corretamente, será possível criar um aplicativo fácil de ser aprendido e agradável de ser usado.

---

[1] N.T.: Um dial de jog é uma espécie de controle com formato circular, usado em geral para controle de áudio e de vídeo, comum em equipamentos profissionais.

Disponibilidade e restrições

Muitos objetos, como ferramentas e eletrodomésticos, foram projetados para proporcionar um uso adequado e nos impedir de usá-los inadequadamente. Isso corresponde aos princípios de disponibilidade e de restrição. Um exemplo está no plugue de três pinos e na tomada. Esses objetos foram projetados não só para se complementarem , mas também para funcionarem de uma determinada maneira. É virtualmente impossível conectar um plugue de três pinos, mostrado na figura 7.8, de maneira incorreta. Com os pinos achatados e um arredondado, o plugue deixa imediatamente claro para as pessoas como elas devem usá-lo. E se não estiver claro, ele evita que elas o conectem incorretamente e acabem se ferindo!

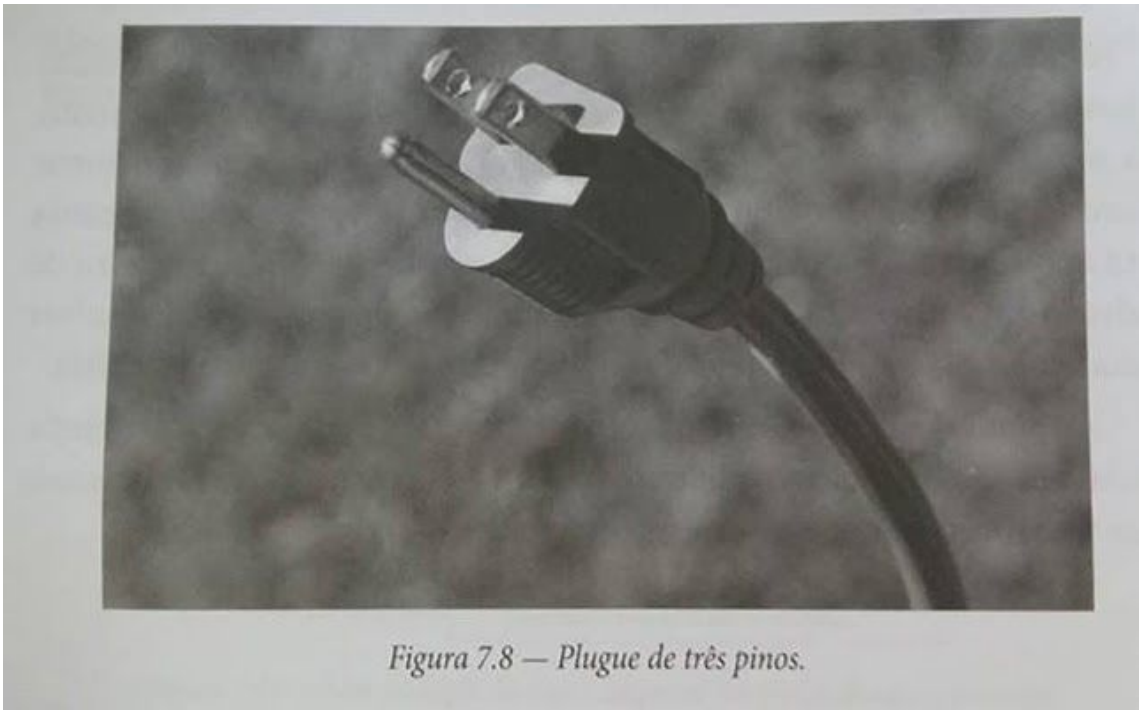

Figura 7.8 — Plugue de três pinos.

No hospital, temos um ditado: “Faça com que seja fácil fazer *o que é certo* e difícil fazer *o que é errado.*” A disponibilidade faz com que seja fácil fazer o que é certo, enquanto as restrições dificultam fazer o que é errado.

Se você observar usuários cometendo erros em seu aplicativo, considere limitar as opções ou antecipar seus fluxos de trabalho. Desenvolva ações que funcionem de modo que seja impossível fazer o que é errado. Os usuários apreciarão o fato de você estar tomando conta deles e terão mais confiança em seu aplicativo.

Confirmação

Uma maneira de evitar que os usuários cometam erros é pedir confirmação. O princípio da confirmação estabelece que um aplicativo deve evitar ações indesejadas ao solicitar uma verificação, conforme mostrado na figura 7.9.

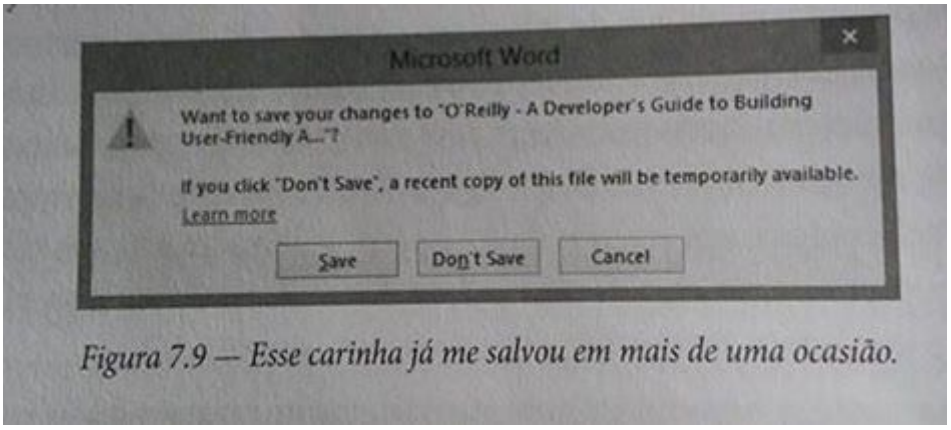

Figura 7.9 — Esse carinha já me salvou em mais de uma ocasião.

Na maioria dos aplicativos, se estou trabalhando com um documento e tento fechá-lo sem salvar, uma pergunta será mostrada. Geralmente, ela indaga se quero salvar antes de sair do programa.Se eu selecionar Cancel (Cancelar) e tentar fechar o aplicativo novamente a pergunta será apresentada de novo. Essencialmente, não há nenhuma maneira de fechar o aplicativo sem antes lidar com a questão de querer ou não salvar o documento. Isso me protege de cometer erros e perder meu trabalho.

Certifique-se de que seu aplicativo antecipará uma ação indesejada. Nada fará seus usuários o odiarem mais do que permitir que eles percam seus trabalhos de maneira não intencional.

A Lei de Hick

A Lei de Hick consistem em um modelo prescritivo que ajudará você a calcular o tempo necessário para os usuários tomarem uma decisão, como resultado da quantidade de opções que possuem. Também é conhecida como tempo de reação, ou TR, e é matematicamente representada da seguinte maneira:

$$RT = a + b\text{Log}_2 N$$

O modelo pode se provar útil na avaliação de seus menus para garantir que não estejam sobrecarregados. Uma pergunta comum no design de um aplicativo é: “Quantos itens devem estar presentes em um menu, e como devem ser organizados?”.

Por exemplo, o esquema de navegação em um portal de uma empresa pode ser extremamente difícil de administrar. Considere o portal de empresa que discutimos anteriormente no princípio de hierarquia. É mais do que provável que os usuários queiram que aquilo que estejam procurando seja o primeiro item no menu. Afinal de contas, é o item mais importante, porque eles estão procurando por ele! Veja a figura 7.10 para analisar um exemplo.

Obviamente, pode haver somente um primeiro item, portanto, pode identificar e dar prioridade aos itens em um menu pode ser um verdadeiro cabo de guerra.

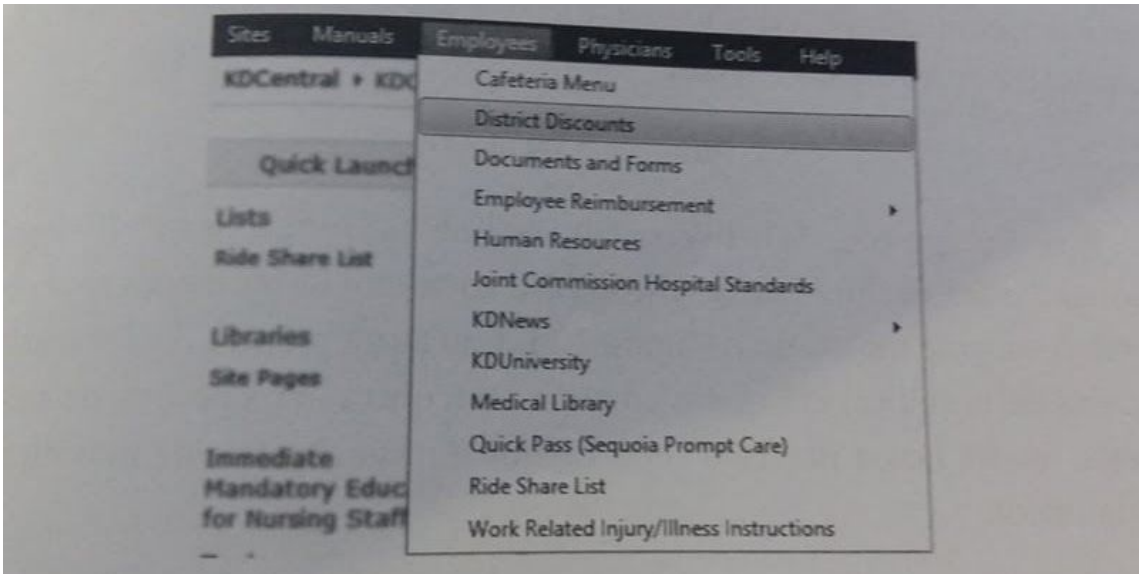

Figura 7.10 — Um exemplo de menu suspenso no portal de nossa empresa.

Desse modo, em virtude de sua natureza linear, a Lei de Hick sugere que processamos as informações a uma taxa constante. Resumindo, quanto mais itens você colocar diante dos usuários, mais tempo será necessário para que eles encontrem o que estão procurando.

Parece óbvio, mas ainda vejo desenvolvedores criarem aplicativos ou sites com menus de navegação incrivelmente complexos. Acho que é fácil perder o controle sobre nossos aplicativos. Ficamos acrescentando mais e mais itens e, antes que tenhamos percebido, não é mais possível administrá-los. Então movemos e reorganizamos os itens para evitar a difícil tarefa de decidir do que devemos nos livrar.

Ao aplicar o princípio da hierarquia e a Lei de Hick como modelo prescritivo, podemos tomar melhores decisões sobre o valor de cada item em nossos menus.

Lei de Fitt

A Lei de Fitt pode ajudar você a determinar o tamanho dos elementos-alvo, tais como botões, menus etc., em sua interface, com base na distância que o dispositivo apontador do usuário deve percorrer. Esse modelo prescritivo é expresso em termos de tempo de movimento, ou ™, e prova que, quanto maior distância que o usuário tiver de percorrer entre dois elementos, menor será a precisão com que o usuário alcançará o alvo. Se sua intenção for fazer o usuário clicar em um botão, o tamanho desse botão será determinado pela distância entre o botão e o cursor do usuário. A equação é apresentada a seguir:

$$MT = a + b \log_2 (2A/W)$$

Suponha que o Google tivesse feito os botões Google Search (Pesquisa Google) e I'm Feeling Lucky (Estou com sorte) menores e mais para a lateral, conforme mostrado na figura 7.11. Isso aumentaria a distância entre a caixa de

pesquisa, em que o cursor se encontra, e os botões, ou alvos. Desse modo, nossa precisão seria menor e nosso tempo de movimento seria maior.

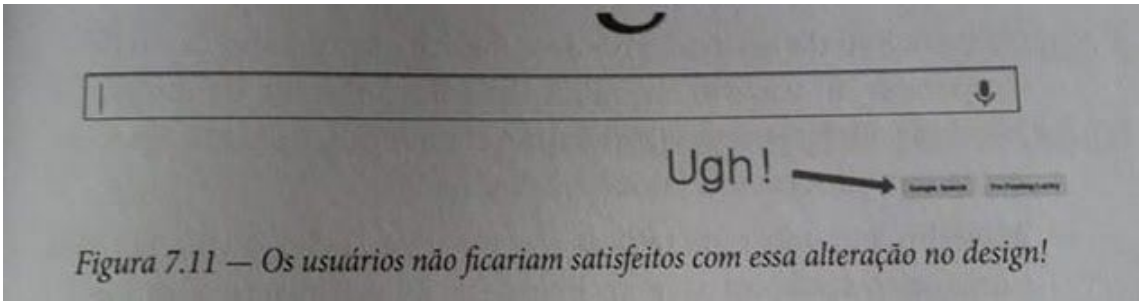

Figura 7.11 — Os usuários não ficariam satisfeitos com essa alteração no design!

A distância que os usuários devem percorrer a partir de um objeto deve determinar o tamanho do objeto para o qual estão se dirigindo. A Lei de Fitt é correlativa. Em outras palavras, quanto maior a distância que um usuário deve percorrer, maiores deverão ser os objetos-alvo. O tamanho exato será determinado pelo tempo de movimento aceitável.

A Lei de Fitt é útil no caso de interfaces orientadas a mouse; porém, há alguns estudos novos que ajustaram o modelo para acomodar interfaces sensíveis ao toque também.

A versão resumida

- É importante aprender e estudar os princípios de design. Há vários princípios e, ao simplesmente seguir suas diretrizes, você poderá melhorar incrivelmente a experiência em seus aplicativos.
- O princípio de proximidade estabelece que os seres humanos percebem um relacionamento entre objetos que estejam mais próximos. Utilize esse princípio agrupando funções. Isso fará com que seja mais fácil aprender a usar e entender seu aplicativo.
- O princípio da visibilidade estabelece que indicadores visuais devem estar presentes para ajudar os usuários a compreender o *status* de seu aplicativo.
- O princípio da proeminência visual estabelece que a atenção de um usuário será atraída para objetos que sejam maiores, que tenham cores mais fortes ou que sejam mais proeminentes.
- O princípio do *feedback* visual estabelece que um aplicativo deve responder ao usuário ao indicar que uma entrada de dados foi recebida. Os aplicativos também devem indicar ao usuário quando estiverem processando uma solicitação.
- Modelos mentais e metáforas são maneiras pelas quais os seres humanos transferem o conhecimento do mundo real para o mundo da computação. Certifique-se de que sua iconografia, a linguagem e outras metáforas estejam calcadas em conhecimentos que os usuários compreendam.
- A revelação progressiva remove e desabilita recursos que não são aplicáveis ao estado corrente dos usuários.

- O princípio da consistência estabelece que as tarefas em um aplicativo devem funcionar conforme esperado. Você não deve inventar novos fluxos de trabalho para executar tarefas que já são compreendidas pelo usuário.
- A disponibilidade ajuda os usuários a fazer o que é certo, e as restrições evitam que façam o que é errado.
- O princípio da confirmação estabelece que os aplicativos devem exigir verificação para evitar que os usuários executem ações indesejadas. O diálogo de confirmação para salvar um documento é um bom exemplo.
- A Lei de Hick consiste em um modelo prescritivo usado para determinar o tempo necessário para selecionar um item, com base na quantidade de itens disponíveis para seleção. Quanto mais itens você incluir em um menu, maior será o tempo de resposta do usuário.
- A Lei de Fitt consiste em outro modelo prescritivo, usado para determinar o tempo necessário para um usuário mover o cursor de um local para o local-alvo. Quanto maior for a distância que um usuário percorrer com seu cursor, menor será a precisão com que o usuário alcançará o objeto-alvo.