13. Usabilidade

As pessoas ignoram o design que ignora as pessoas. -Frank Chimero

A usabilidade se preocupa com a facilidade com que o usuário consegue realizar uma tarefa desejada e com o tipo de suporte que o sistema oferece. Ao longo dos anos, o foco na usabilidade demonstrou ser uma das maneiras mais baratas e fáceis de melhorar a qualidade de um sistema (ou, mais precisamente, a percepção de qualidade do usuário) e, consequentemente, a satisfação do usuário final.

A usabilidade abrange as seguintes áreas:

- **Aprender as funcionalidades do sistema.** Se o usuário não está familiarizado com um sistema específico ou um aspecto particular dele, o que o sistema pode fazer para facilitar a tarefa de aprendizado? Isso pode incluir o fornecimento de recursos de ajuda.
- **Usar um sistema de forma eficiente.** O que o sistema pode fazer para tornar o usuário mais eficiente em sua operação? Isso pode incluir permitir que o usuário redirecione o sistema após emitir um comando. Por exemplo, o usuário pode desejar suspender uma tarefa, realizar várias operações e, em seguida, retomar essa tarefa.
- **Minimizar o impacto de erros do usuário.** O que o sistema pode fazer para garantir que um erro do usuário tenha um impacto mínimo? Por exemplo, o usuário pode desejar cancelar um comando emitido incorretamente ou desfazer seus efeitos.
- **Adaptar o sistema às necessidades do usuário.** Como o usuário (ou o próprio sistema) pode se adaptar para facilitar a tarefa do usuário? Por exemplo, o sistema pode preencher automaticamente URLs com base nas entradas anteriores de um usuário.
- **Aumentar a confiança e a satisfação.** O que o sistema faz para dar ao usuário a confiança de que a ação correta está sendo tomada? Por exemplo, fornecer feedback que indica que o sistema está realizando uma tarefa de longa duração, juntamente com a porcentagem de conclusão até o momento, aumentará a confiança do usuário no sistema.

Pesquisadores focados em interações humano-computador usaram os termos **iniciativa do usuário**, **iniciativa do sistema** e **iniciativa mista** para descrever qual dos pares humano-computador toma a iniciativa na execução de certas ações e como a interação prossegue. Cenários de usabilidade podem combinar iniciativas de ambas as perspectivas. Por exemplo, ao cancelar um comando, o usuário emite um cancelamento (iniciativa do usuário) e o sistema responde. Durante o cancelamento, no entanto, o sistema pode exibir um indicador de progresso (iniciativa do sistema). Assim, a operação de cancelamento pode compreender uma iniciativa mista. Neste capítulo, usaremos essa distinção entre iniciativa do usuário e iniciativa do sistema para discutir as táticas que o arquiteto usa para alcançar os vários cenários.

PÁGINA 2

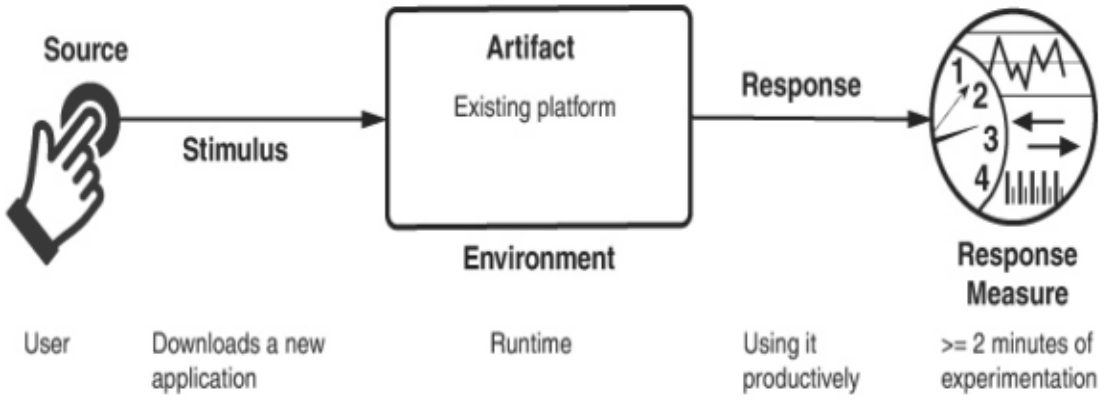
Existe uma forte conexão entre a obtenção de usabilidade e a modificabilidade. O processo de design da interface do usuário consiste em gerar e testar um design de interface do usuário. É altamente improvável que você acerte na primeira vez, então você deve planejar iterar esse processo e, portanto, deve projetar sua arquitetura para tornar essa iteração menos dolorosa. É por isso que a usabilidade está fortemente conectada à modificabilidade. Conforme você itera, as deficiências no design são — espera-se — corrigidas e o processo se repete.

Essa conexão resultou em padrões padrão para suportar o design da interface do usuário. De fato, uma das coisas mais úteis que você pode fazer para alcançar a usabilidade é modificar seu sistema, repetidamente, para torná-lo melhor à medida que aprende com seus usuários e descobre melhorias a serem feitas.

PÁGINA 3

Porção do Cenário	Descrição	Valores Possíveis
Fonte	De onde vem o estímulo?	O usuário final (que pode estar em um papel especializado, como um administrador de sistema ou de rede) é a principal fonte do estímulo para a usabilidade. Um evento externo chegando a um sistema (ao qual o usuário pode reagir) também pode ser uma fonte de estímulo.
Estímulo	O que o usuário final deseja?	O usuário final deseja: • Usar um sistema de forma eficiente • Aprender a usar o sistema • Minimizar o impacto de erros • Adaptar o sistema • Configurar o sistema
Ambiente	Quando o estímulo alcança o sistema?	As ações do usuário com as quais a usabilidade se preocupa sempre ocorrem em tempo de execução ou no momento da configuração do sistema.

Artefatos	Qual parte do sistema está sendo estimulada?	Exemplos comuns incluem: • Uma GUI • Uma interface de linha de comando • Uma interface de voz • Uma tela de toque
Resposta	Como o sistema deve responder?	O sistema deve: • Fornecer ao usuário os recursos necessários • Antecipar as necessidades do usuário • Fornecer feedback apropriado ao usuário
Medida da Resposta	Como a resposta é medida?	Um ou mais dos seguintes: •Tempo da tarefa • Número de erros • Tempo de aprendizado • Razão entre o tempo de aprendizado e o tempo da tarefa



A Figura 13.1 fornece um exemplo de um cenário concreto de usabilidade que você poderia gerar usando a Tabela 13.1: O usuário baixa um novo aplicativo e o utiliza produtivamente após 2 minutos de experimentação.

(Descrição da Figura 13.1: Cenário de usabilidade de amostra) Um diagrama de fluxo mostrando:

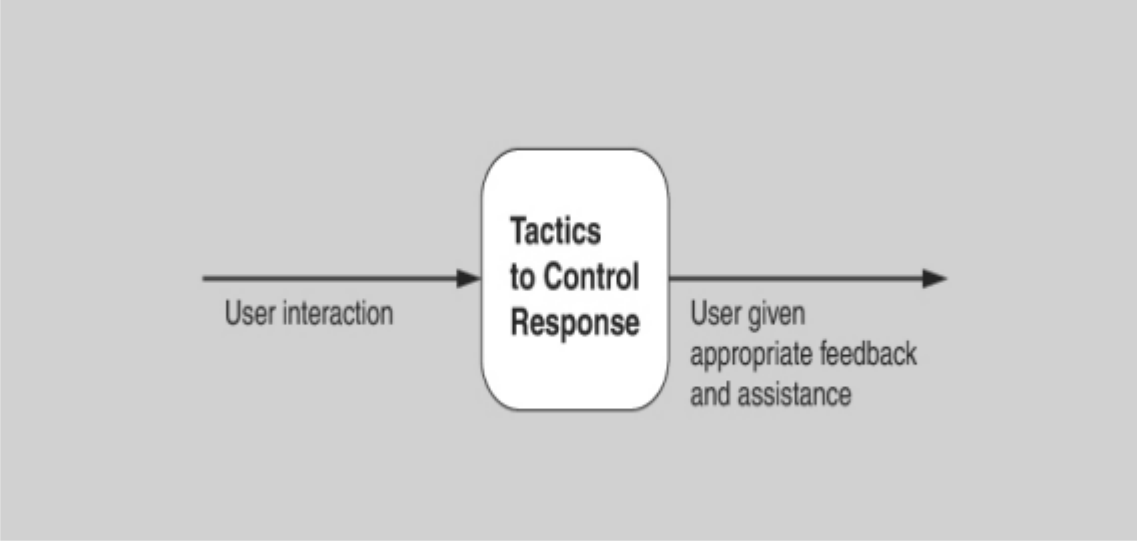
- **Fonte:** Usuário
- **Estímulo:** Baixa um novo aplicativo
- **Artefato:** Plataforma existente
- **Ambiente:** Tempo de execução
- **Resposta:** Usando-o produtivamente
- **Medida da Resposta:** >= 2 minutos de experimentação

PÁGINA 4

- Número de tarefas realizadas
- Satisfação do usuário
- Ganho de conhecimento do usuário

13.2 Táticas para Usabilidade

A Figura 13.2 mostra o objetivo do conjunto de táticas de usabilidade.



(Descrição da Figura 13.2: O objetivo das táticas de usabilidade) Um diagrama de fluxo mostrando:

- **Entrada:** Interação do usuário
- **Processo:** Táticas para Controlar a Resposta
- **Saída:** Usuário recebe feedback e assistência apropriados

PÁGINA 5

Apoiar a Iniciativa do Usuário

Uma vez que um sistema está em execução, a usabilidade é aprimorada ao dar ao usuário feedback sobre o que o sistema está fazendo e ao permitir que o usuário dê respostas apropriadas. Por exemplo, as táticas descritas a seguir — cancelar, desfazer, pausar/retomar e agregar — apoiam o usuário na correção de erros ou em ser mais eficiente. O arquiteto projeta uma resposta para a iniciativa do usuário enumerando e alocando as responsabilidades do sistema para responder ao comando do usuário. Aqui estão alguns exemplos comuns de táticas para apoiar a iniciativa do usuário:

- **Cancelar.** Quando o usuário emite um comando de cancelamento, o sistema deve estar ouvindo-o (portanto, há a responsabilidade de ter um ouvinte constante que não é bloqueado pelas ações do que quer que esteja sendo cancelado); a atividade que está sendo cancelada deve ser encerrada; quaisquer recursos que estão sendo usados pela atividade cancelada devem ser liberados; e os componentes que estão colaborando com a atividade cancelada devem ser informados para que também possam tomar as medidas apropriadas.
- **Desfazer.** Para suportar a capacidade de desfazer, o sistema deve manter uma quantidade suficiente de informações sobre o estado do sistema para que um estado anterior possa ser restaurado, a pedido do usuário. Tal registro pode assumir a forma de "instantâneos" de estado — por exemplo, pontos de verificação — ou um conjunto de operações reversíveis. Nem todas as operações podem ser facilmente revertidas. Por exemplo, alterar todas as ocorrências da letra "a" para a letra "b" em um documento não pode ser revertido alterando todas as instâncias de "b" para "a", porque algumas dessas instâncias de "b" podem ter existido antes da alteração original. Nesse caso, o sistema deve manter um registro mais elaborado da alteração. Claro, algumas operações não podem ser desfeitas de forma alguma: você não pode cancelar o envio de um pacote ou desfazer o disparo de um míssil, por exemplo. Desfazer vem em diferentes tipos. Alguns sistemas permitem um único desfazer (onde invocar o desfazer novamente o reverte para o estado em que você comandou o primeiro desfazer, essencialmente desfazendo o desfazer). Em outros sistemas, comandar múltiplas operações de desfazer o leva de volta por muitos estados anteriores, seja até um certo limite ou até o momento em que o aplicativo foi aberto pela última vez.
- **Pausar/retomar.** Quando um usuário inicia uma operação de longa duração — digamos, o download de um arquivo grande ou um conjunto de arquivos de um servidor — muitas vezes é útil fornecer a capacidade de pausar e retomar a operação. Pausar uma operação de longa duração pode ser feito para liberar temporariamente recursos para que possam ser realocados para outras tarefas.

PÁGINA 6

- **Agregar.** Quando um usuário está realizando operações repetitivas, ou operações que afetam um grande número de objetos da mesma maneira, é útil fornecer a

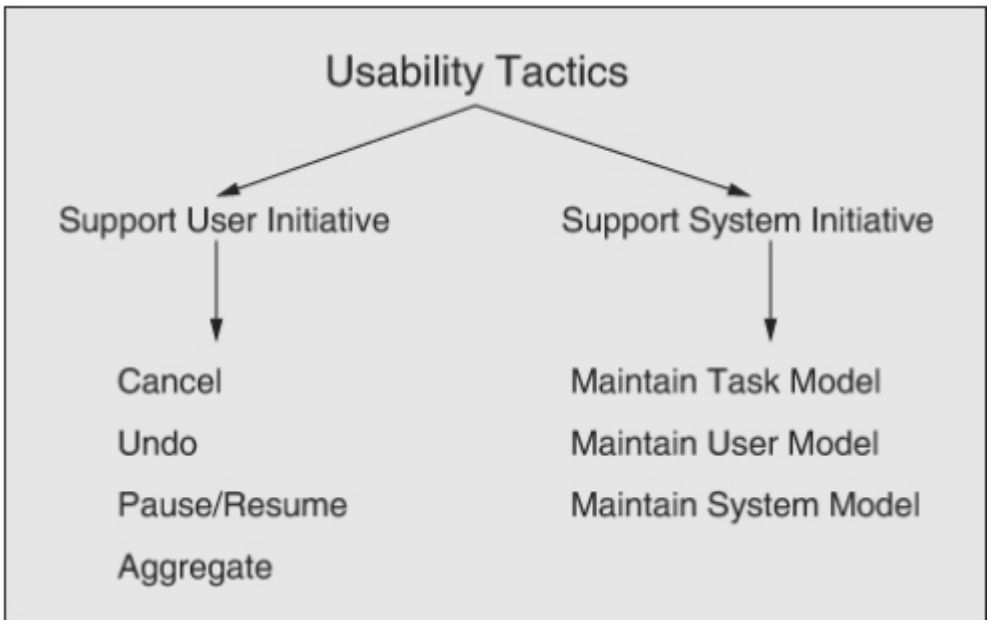
capacidade de agregar os objetos de nível inferior em um único grupo, para que a operação possa ser aplicada ao grupo, liberando assim o usuário da monotonia, e do potencial de erros, de fazer a mesma operação repetidamente. Um exemplo é agregar todos os objetos em um slide e alterar o texto para a fonte de 14 pontos.

Apoiar a Iniciativa do Sistema

Quando o sistema toma a iniciativa, ele deve se basear em um modelo do usuário, um modelo da tarefa que está sendo realizada pelo usuário, ou um modelo do estado do sistema. Cada modelo requer vários tipos de entrada para realizar sua iniciativa. As táticas de apoio à iniciativa do sistema identificam os modelos que o sistema usa para prever seu próprio comportamento ou a intenção do usuário. Encapsular essa informação tornará mais fácil personalizá-la ou modificá-la. A personalização e a modificação podem ser baseadas dinamicamente no comportamento passado do usuário ou ocorrer offline durante o desenvolvimento. As táticas relevantes são descritas aqui:

- **Manter modelo da tarefa.** O modelo da tarefa é usado para determinar o contexto para que o sistema possa ter alguma ideia do que o usuário está tentando fazer e fornecer assistência. Por exemplo, muitos motores de busca fornecem capacidades preditivas de digitação antecipada, e muitos clientes de e-mail fornecem correção ortográfica. Ambas as funções são baseadas em modelos de tarefa.
- **Manter modelo do usuário.** Este modelo representa explicitamente o conhecimento do usuário sobre o sistema, o comportamento do usuário em termos de tempo de resposta esperado e outros aspectos específicos de um usuário ou uma classe de usuários. Por exemplo, aplicativos de aprendizado de idiomas estão constantemente monitorando áreas onde um usuário comete erros e, em seguida, fornecendo exercícios adicionais para corrigir esses comportamentos. Um caso especial dessa tática é comumente encontrado na personalização da interface do usuário, onde um usuário pode modificar explicitamente o modelo de usuário do sistema.
- **Manter modelo do sistema.** O sistema mantém um modelo explícito de si mesmo. Isso é usado para determinar o comportamento esperado do sistema para que um feedback apropriado possa ser dado ao usuário. Uma manifestação comum de um modelo de sistema é uma barra de progresso que prevê o tempo necessário para completar a atividade atual.

A Figura 13.3 resume as táticas para alcançar a usabilidade.



Descrição da Figura 13.3: Táticas de Usabilidade) Um diagrama em árvore:

- **Topo:** Táticas de Usabilidade
  - **Ramo Esquerdo:** Apoiar Iniciativa do Usuário
    - Cancelar
    - Desfazer
    - Pausar/Retomar
    - Agregar
  - **Ramo Direito:** Apoiar Iniciativa do Sistema
    - Manter Modelo da Tarefa
    - Manter Modelo do Usuário
    - Manter Modelo do Sistema

13.3 Questionário Baseado em Táticas para Usabilidade

Com base nas táticas descritas na Seção 13.2, podemos criar um conjunto de perguntas inspiradas em táticas de usabilidade, conforme apresentado na Tabela 13.2. Para obter uma visão geral das escolhas arquitetônicas feitas para suportar a usabilidade, o analista faz cada pergunta e registra as respostas na tabela. As respostas a essas perguntas podem então se tornar o foco de atividades futuras: investigação de documentação, análise de código ou outros artefatos, engenharia reversa de código e assim por diante.

Tabela 13.2 Questionário Baseado em Táticas para Usabilidade

Grupo de Táticas	Pergunta sobre Táticas	Suportado? (S/N)	Risco	Decisões de Design e Localização	Justificativa e Pressupostos
Apoiar Iniciativa do Usuário	O sistema é capaz de ouvir e responder a um comando de cancelamento?				
	É possível desfazer o último comando, ou os últimos vários comandos?				
	É possível pausar e depois retomar operações de longa duração?				
	É possível agregar objetos da UI em um grupo e aplicar operações no grupo?				
Apoiar Iniciativa do Sistema	O sistema mantém um modelo da tarefa?				
	O sistema mantém um modelo do usuário?				
	O sistema mantém um modelo de si mesmo?				

13.4 Padrões para Usabilidade

Discutiremos brevemente três padrões de usabilidade: model-view-controller (MVC) e suas variantes, observer e memento. Esses padrões promovem principalmente a usabilidade ao promover a separação de responsabilidades, o que, por sua vez, facilita a iteração do design de uma interface de usuário. Outros tipos de padrões também são possíveis, incluindo padrões usados no design da própria interface do usuário, como breadcrumbs, carrinho de compras ou divulgação progressiva — mas não os discutiremos aqui.

Model-View-Controller

MVC é provavelmente o padrão mais conhecido para usabilidade. Ele vem em muitas variantes, como MVP (model-view-presenter), MVVM (model-view-view-model), MVA (model-view-adapter) e assim por diante. Essencialmente, todos esses padrões estão focados em separar o modelo — a lógica de "negócios" subjacente do sistema — de sua realização em uma ou mais visões (views) da UI. No modelo MVC original, o modelo enviava atualizações para uma visão, que um usuário veria e com a qual interagiria. As interações do usuário — pressionamentos de tecla, cliques de botão, movimentos do mouse e assim por diante — são transmitidas para o controlador, que as interpreta como operações no modelo e, em seguida, envia essas operações para o modelo, que altera seu estado em resposta. O caminho inverso também era uma porção do padrão MVC original. Ou seja, o modelo pode ser alterado e o controlador enviaria atualizações para a visão. O envio de atualizações depende se o MVC está em um único processo ou está distribuído por processos (e potencialmente pela rede). Se o MVC está em um processo, as atualizações são enviadas usando o padrão observer (discutido na próxima subseção). Se o MVC está distribuído por processos, o padrão publish-subscribe é frequentemente usado para enviar atualizações (ver Capítulo 8).

Benefícios:

- Como o MVC promove uma clara separação de responsabilidades, as alterações em um aspecto do sistema, como o layout da UI (a visão), muitas vezes não têm consequências para o modelo ou o controlador.
- Além disso, como o MVC promove a separação de responsabilidades, os desenvolvedores podem trabalhar em todos os aspectos do padrão — modelo, visão e controlador — de forma relativamente independente e em paralelo. Esses aspectos separados também podem ser testados em paralelo.
- Um modelo pode ser usado em sistemas com diferentes visões, ou uma visão pode ser usada em sistemas com diferentes modelos.

Desvantagens:

- O MVC pode se tornar oneroso para UIs complexas, pois a informação muitas vezes está espalhada por vários componentes. Por exemplo, se houver múltiplas visões do mesmo modelo, uma alteração no modelo pode exigir alterações em vários componentes que, de outra forma, não estariam relacionados.
- Para UIs simples, o MVC adiciona uma complexidade inicial que pode não compensar em economias futuras.

PÁGINA 10

- O MVC adiciona uma pequena quantidade de latência às interações do usuário. Embora isso seja geralmente aceitável, pode ser problemático para aplicações que exigem latência muito baixa.

Observer

O padrão observer é uma maneira de vincular alguma funcionalidade a uma ou mais visões. Este padrão tem um sujeito — a entidade sendo observada — e um ou mais observadores desse sujeito. Os observadores precisam se registrar com o sujeito; então, quando o estado do sujeito muda, os observadores são notificados. Este padrão é frequentemente usado para implementar o MVC (e suas variantes), por exemplo, como uma forma de notificar as várias visões sobre as alterações no modelo.

Benefícios:

- Este padrão separa alguma funcionalidade subjacente da preocupação de como, e quantas vezes, essa funcionalidade é apresentada.
- O padrão observer facilita a alteração das ligações entre o sujeito e os observadores em tempo de execução.

Desvantagens:

- O padrão observer é excessivo se múltiplas visões do sujeito não forem necessárias.
- O padrão observer exige que todos os observadores se registrem e cancelem o registro com o sujeito. Se os observadores se esquecerem de cancelar o registro, sua memória nunca será liberada, o que efetivamente resulta em um vazamento de memória. Além disso, isso pode afetar negativamente o desempenho, já que observadores obsoletos continuarão a ser invocados.
- Os observadores podem precisar fazer um trabalho considerável para determinar se e como refletir uma atualização de estado, e esse trabalho pode ser repetido para cada observador. Por exemplo, suponha que o sujeito esteja mudando seu estado em uma granularidade fina, como um sensor de temperatura que reporta flutuações de 1/100 de grau, mas a visão só atualiza as mudanças em graus inteiros. Em tais casos onde há uma "incompatibilidade de impedância", recursos de processamento substanciais podem ser desperdiçados.

Memento

O padrão memento é uma maneira comum de implementar a tática de desfazer. Este padrão apresenta três componentes principais: o originador, o zelador e o memento. O originador está processando algum fluxo de eventos que mudam seu estado (originados da interação do usuário). O zelador está enviando eventos ao originador que fazem com que ele mude seu estado. Quando o zelador está prestes a mudar o estado do originador, ele

pode solicitar um *memento* — um instantâneo do estado existente e pode usar este artefato para restaurar esse estado existente se necessário, simplesmente passando o *memento* de volta para o originador. Desta forma, o zelador não sabe nada sobre como o estado é gerenciado; o *memento* é simplesmente uma abstração que o zelador emprega.

Benefícios:

- O benefício óbvio deste padrão é que você delega o complicado processo de implementar o desfazer, e de descobrir qual estado preservar, para a classe que está realmente criando e gerenciando esse estado. Em consequência, a abstração do originador é preservada e o resto do sistema não precisa conhecer os detalhes.

Desvantagens:

- Dependendo da natureza do estado que está sendo preservado, o memento pode consumir quantidades arbitrariamente grandes de memória, o que pode afetar o desempenho. Em um documento muito grande, tente cortar e colar muitas seções grandes e, em seguida, desfazer tudo isso. É provável que isso resulte em uma lentidão perceptível do seu processador de texto.
- Em algumas linguagens de programação, é difícil impor o memento como uma abstração opaca.

13.5 Para Leitura Adicional

Claire Marie Karat investigou a relação entre usabilidade e vantagem comercial [Karat 94].

Jakob Nielsen também escreveu extensivamente sobre este tópico, incluindo um cálculo do ROI da usabilidade [Nielsen 08].

Bonnie John e Len Bass investigaram a relação entre usabilidade e arquitetura de software. Eles enumeraram aproximadamente duas dúzias de cenários de usabilidade que têm impacto arquitetônico e forneceram padrões associados para esses cenários [Bass 03].

Greg Hartman definiu a atenção como a capacidade do sistema de apoiar a iniciativa do usuário e permitir o cancelamento ou pausa/retomada [Hartman 10].

13.6 Questões para Discussão

1. Escreva um cenário concreto de usabilidade para o seu automóvel que especifique quanto tempo você leva para definir suas estações de rádio favoritas. Agora considere outra parte da experiência do motorista e crie cenários que testem outros aspectos das medidas de resposta da tabela de cenário geral (Tabela 13.1).
2. Como a usabilidade pode se contrapor à segurança? Como ela pode se contrapor ao desempenho?
3. Escolha alguns de seus sites favoritos que fazem coisas semelhantes, como redes sociais ou compras online. Agora escolha uma ou duas respostas apropriadas do

cenário geral de usabilidade (como "antecipar a necessidade do usuário") e uma medida de resposta correspondente apropriada. Usando a resposta e a medida de resposta que você escolheu, compare a usabilidade dos sites.

4. Por que em tantos sistemas o botão de cancelar em uma caixa de diálogo parece não responder? Quais princípios arquitetônicos você acha que foram ignorados nesses sistemas?
5. Por que você acha que as barras de progresso frequentemente se comportam de maneira errática, movendo-se de 10 para 90 por cento em um passo e depois ficando presas em 90 por cento?
6. Pesquise o acidente do voo 296 da Air France na floresta de Habsheim, França, em 1988. Os pilotos disseram que não conseguiram ler o visor digital do rádio altímetro ou ouvir sua leitura audível. Neste contexto, discuta a relação entre usabilidade e segurança.