

CAPÍTULO 4

AVALIAÇÃO DE INTERFACES

For a company to say 'we don't need evaluation' is just the same as saying 'our system designers don't need to see anything when they are driving: they can drive with their eyes shut and achieve the goal that they want'. You can't possibly produce a good product blindfolded.

Brian Shackel apud Preece *et al.*, 1994, p. 600

INTRODUÇÃO

Avaliação não deve ser vista como uma fase única dentro do processo de design e muito menos como uma atividade a ser feita somente no final do processo e se "der tempo". Idealmente, avaliação deve ocorrer durante o ciclo de vida do design e seus resultados utilizados para melhorias gradativas da interface. É claro que não se pode pretender efetuar extensivos testes experimentais durante todo o processo de design, mas técnicas informais e analíticas devem ser utilizadas. Nesse sentido existe uma forte correlação entre avaliação e as técnicas de modelagem e construção de protótipos discutidas no Capítulo 3, pois essas técnicas garantem que o design está constantemente sob avaliação. E na maioria de modelos de desenvolvimento de interfaces usáveis a avaliação tem um papel central, como no modelo Estrela (Hix e Hartson, 1993) apresentado no capítulo anterior (Figura 4.1).

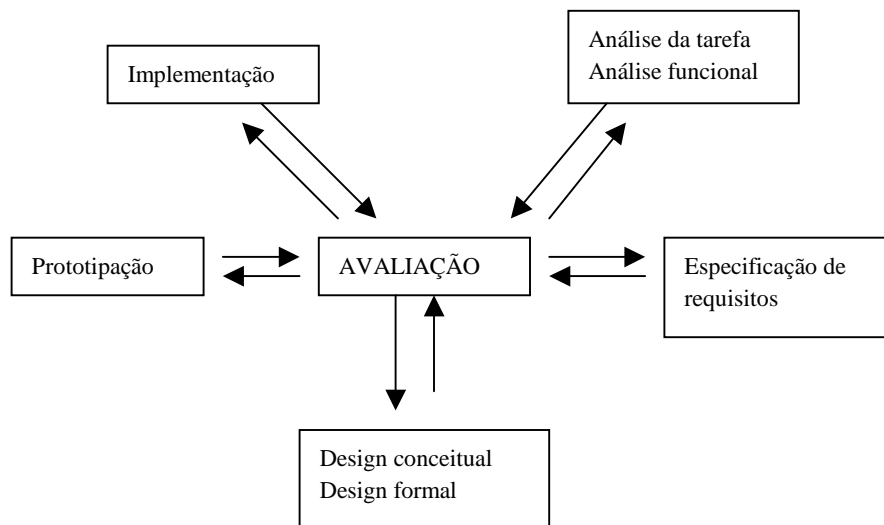


FIGURA 4.1 - O CICLO DE VIDA ESTRELA (ADAPTADO DE HIX E HARTSON, 1993)

Diferentes tipos de avaliação são necessárias em diferentes estágios do design. Nos estágios bem iniciais onde idéias estão sendo exploradas e tentadas, muitas vezes testes bastante informais são suficientes. Por exemplo, depois de uma sessão de discussão (*brainstorming*) para explorar diferentes metáforas, o conjunto inicial de opções certamente estará bem reduzido. Outras vezes, principalmente em estágios um pouco mais avançados do processo, avaliações mais formais devem ser planejadas.

Os fatores determinantes de um plano de avaliação incluem (Nielsen, 1993; Hix and Hartson, 1993; Preece *et al.*, 1994; Schneiderman, 1998):

- estágio do design (início, meio ou fim)
- quão pioneiro é o projeto (bem definido *versus* exploratório)
- número esperado de usuários
- quão crítica é a interface (por exemplo, um sistema de controle de tráfego aéreo *versus* um sistema de orientação de um shopping)
- custo do produto e orçamento alocado para o teste
- tempo disponível
- experiência dos designers e avaliadores

O domínio do plano de avaliação pode estar entre um ambicioso teste de dois anos, com múltiplas fases, de um novo sistema de controle de tráfego aéreo até um discreto teste de três dias com seis usuários de um sistema interno de contabilidade. Da mesma forma, o custo pode variar de 10 % até 1 % do custo total do projeto.

Mesmo considerando que é uma atitude irresponsável não efetuar alguma forma de avaliação, deve-se ter claro que um certo grau de incerteza sempre permanece mesmo após exaustivos testes com múltiplos métodos. Perfeição não é possível, portanto qualquer planejamento deve prever métodos de avaliação contínua e reparo de problemas durante todo ciclo de vida de uma interface.

Deve-se ter em vista ao analisar métodos de avaliação que eles apresentam resultados bastante satisfatórios para grande parte dos sistemas em condições normais de uso, mas a performance de sistemas com grande complexidade de entradas, como por exemplo, as emergências de um sistema de controle de um reator nuclear ou de tráfego aéreo, são muito difíceis de serem testadas.

O objetivo deste capítulo não é apresentar todas as possíveis técnicas de avaliação e nem prescrever exatamente quando e como usá-las. Estaremos tratando com mais detalhe três técnicas que julgamos representativas: uma delas a mais tradicional que é o teste com usuários e as outras duas, avaliação heurística e percurso cognitivo, por serem as que apresentam mais e melhores resultados práticos, além de poderem ser aprendidas com certa facilidade. E discutiremos ao longo do capítulo as razões da existência de diferentes abordagens de avaliação.

Ressaltamos que na maioria das situações práticas uma ou mais técnicas são adotadas e adaptadas de forma a atender as necessidades específicas da interface sob avaliação. Geralmente os recursos disponíveis têm um grande impacto no tipo de avaliação a ser feita. Portanto, selecionar a técnica de avaliação adequada envolve *escolher, misturar e adaptar* técnicas a partir do conjunto de técnicas disponíveis.

OBJETIVOS DA AVALIAÇÃO

De forma geral, se faz avaliação para conhecer o que os usuários querem e os problemas que eles experimentam, pois quanto melhor informados sobre seus usuários os designers estiverem, melhor serão os design de seus produtos.

Muitas questões poderiam ser objetivo de uma avaliação. O setor de marketing poderia estar interessado em como o produto de sua empresa se compara com produtos de outros competidores do mercado. Por exemplo, se a funcionalidade e a aceitação do produto é melhor, ou pelo menos igual, à do principal competidor. Produtos também podem ser avaliados no sentido de verificar se estão de acordo com padrões específicos, como as normas ISO, por exemplo.

Avaliações são necessárias para responder dúvidas que surgem durante o processo de design e desenvolvimento de um produto. Em muitos pontos do processo de design as pessoas de desenvolvimento necessitam respostas a questões de modo a verificar se suas idéias são realmente o que os usuários necessitam ou desejam. Desse modo, avaliação direciona e se mescla ao design, auxiliando na criação de um produto útil e usável, como visto no Capítulo 3. Em contraste, podemos ter as avaliações que ocorrem depois do produto desenvolvido e que se preocupam em fazer julgamentos sobre o produto final: sua performance considerando produtos competitivos, sua adequação à uma determinada família de produtos, etc. Como um dos focos desse livro é em design estaremos neste capítulo nos atendo mais às avaliações feitas durante o processo de design e seus resultados.

Resumidamente, podemos dizer que avaliação tem três grandes objetivos: avaliar a funcionalidade do sistema, avaliar o efeito da interface junto ao usuário e identificar problemas específicos do sistema.

A **funcionalidade** do sistema é importante no sentido de estar adequada aos requisitos da tarefa do usuário, ou seja, o design do sistema deve permitir ao usuário efetuar a tarefa pretendida e de modo mais fácil e eficiente. Isso inclui não somente ter a funcionalidade adequada disponível, mas também torná-la usável, na forma de ações que o usuário precisa efetuar para executar a tarefa. Avaliação nesse nível envolve também medir a performance do usuário junto ao sistema, ou seja, avaliar a eficiência do sistema na execução da tarefa pelo usuário.

Adicionalmente, é preciso medir o **impacto do design junto ao usuário**, ou seja, avaliar sua usabilidade. Isso inclui considerar aspectos tais como: avaliar quão fácil é aprender a usar o sistema; a atitude do usuário com relação ao sistema; identificar áreas do design as quais sobrecarregam o usuário de alguma forma, por exemplo, exigindo que uma série de informações sejam lembradas; etc. Muitos dos métodos concentram a avaliação sobre aspectos padrão de usabilidade, como o uso de guidelines como apresentado no Capítulo 3.

O terceiro objetivo da avaliação é **identificar problemas específicos com o design**, ou seja, identificar aspectos do design os quais quando usados no contexto alvo, causam resultados inesperados ou confusão entre os usuários. Isso é claro está correlacionado tanto com a funcionalidade quanto com a usabilidade do design.

Atendendo a esses objetivos pode-se classificar os métodos de avaliação em duas dimensões: se usuários reais estão ou não envolvidos e se a interface está ou não implementada. Considera-se implementação qualquer protótipo executável (Whitefield *et al.*, 1991). Nessas dimensões estaremos neste capítulo tratando de dois grupos de métodos:

- **inspeção de usabilidade** (*predictive evaluation*) - sem envolver usuários e podendo ser usado em qualquer fase do desenvolvimento de um sistema (implementado ou não)
- **testes de usabilidade** - métodos de avaliação centrados no usuário que incluem métodos experimentais ou empíricos, métodos observacionais e técnicas de questionamento (como nos métodos etno-gráficos vistos no Capítulo 3). Para se usar esses métodos é necessária a existência de uma implementação real do sistema em algum formato que pode ser desde uma simulação da capacidade interativa do sistema, sem nenhuma funcionalidade, um protótipo básico implementando, um cenário, ou até a implementação completa.

Outros grupos de métodos que não serão tratados neste capítulo incluem:

- **Experimentos controlados** - envolvem efetuar um bem projetado e controlado experimento de laboratório. Tem-se sempre definida uma hipótese a ser testada e todas as variáveis de interesse necessitam ser controladas. Conhecimento estatístico é necessário para validar os resultados. Controlar todas as variáveis dentro de interações complexas envolvendo humanos além de difícil é de validade muito questionável. A metodologia experimental é seguida, tendo-se o experimentador controlando certas variáveis enquanto examina outras. Os experimentos são feitos em laboratórios especialmente construídos para esse fim e existe muito rigor na observação e monitoramento do uso do sistema. Os dados coletados são analisados quantitativamente de modo a produzir métricas que guiem o design (Preece *et al.*, 1994; Dix *et al.*, 1998).
- **Métodos de avaliação interpretativos** - o objetivo desse tipo de avaliação é possibilitar aos designers um maior entendimento de como os usuários se utilizam dos sistemas em seu ambiente natural e como o uso desses sistemas se integra com outras atividades. Portanto, alguma forma de participação do usuário na coleta, análise ou interpretação dos dados é bastante comum. Os métodos que pertencem a esse grupo incluem avaliação participativa e conceitual que são dois métodos desenvolvidos especialmente para avaliar IHC, e avaliação etnográfica, uma técnica emprestada da antropologia, conforme discutido no Capítulo 3. Formas de registro como vídeos e áudio podem ser feitas como em outros métodos mas a forma de análise é bastante diferenciada (Preece *et al.*, 1994; Monk *et al.*, 1993; Greenbaum e Kying, 1991).

INSPEÇÃO DE USABILIDADE

Define-se inspeção de usabilidade como um conjunto de métodos baseados em se ter avaliadores inspecionando ou examinando aspectos relacionados a usabilidade de uma interface de usuário. Os avaliadores podem ser especialistas em usabilidade, consultores de desenvolvimento de software, especialistas em um determinado padrão de interface, usuários finais, etc.

Diferentes métodos de inspeção têm objetivos diferentes, mas normalmente inspeção de usabilidade é proposta como um modo de avaliar design de interfaces baseado no julgamento de avaliadores e são sustentados pela confiança depositada em seus julgamentos. Os métodos variam no sentido de como os julgamentos são efetuados e em quais critérios se espera que o avaliador baseie seus julgamentos.

Pode-se contrastar os métodos de inspeção com outros modos de se obter dados de usabilidade: **automaticamente**, onde medidas de usabilidade são computadas executando-se um software de avaliação que recebe como entrada uma especificação formal da interface; **empiricamente**, testando a interface com usuários reais; **formalmente**, usando modelos exatos e fórmulas para calcular as medidas de usabilidade; e **informalmente**, usando a habilidade e experiência de avaliadores. Inspeções de usabilidade correspondem a categoria de métodos informais.

No estado atual da arte, métodos automáticos não funcionam e métodos formais são muito difíceis de serem aplicados não funcionando bem quando se tem interfaces complexas e altamente interativas (Kahan e Prail, 1994).

Métodos empíricos ou testes de usabilidade são o principal modo de avaliar interfaces e certamente o mais tradicional (estaremos discutindo mais profundamente no decorrer deste capítulo). Mas geralmente, usuários reais são difíceis e caros para serem recrutados de forma a se poder testar todas as fases do desenvolvimento evolutivo de uma interface. Muitos estudos demonstram que muitos problemas encontrados por métodos de inspeção não são detectados com testes de usuários e vice-versa. Esses estudos sugerem que os melhores resultados são obtidos combinando testes com usuários e inspeções.

OBJETIVOS DA INSPEÇÃO

Inspeção de usabilidade objetiva encontrar problemas de usabilidade em um design de uma interface de usuário e com base nesses problemas fazer recomendações no sentido de eliminar os problemas e melhorar a usabilidade do design. Isso significa que inspeções de usabilidade são feitas em um estágio onde a interface está sendo gerada e a sua usabilidade (e utilidade) necessita ser avaliada

Problemas de usabilidade podem ser definidos como aspectos da interface do usuário que podem causar uma usabilidade reduzida ao usuário final do sistema.

Usabilidade, como já visto em capítulos anteriores, é um termo bastante amplo que se refere a quão fácil é para o usuário final aprender a usar o sistema, quão eficientemente ele irá utilizar o sistema assim que aprenda como usar e quão agradável é o seu uso. Também, a frequência e a severidade dos erros do usuário são consideradas como partes constituintes da usabilidade. Entretanto, um usuário pode achar um elemento da interface problemático por muitas razões: torna o sistema difícil de aprender, torna-o lento na execução de suas tarefas, causa erros de uso, ou pode ser simplesmente feio e desagradável. Muito do trabalho da inspeção de usabilidade é classificar e contar o número de problemas de usabilidade. Esta análise depende da exata definição do que é um problema de usabilidade e julgamentos de como diferentes fenômenos constituem manifestações de um único problema. Frequentemente, é muito difícil fazer essas distinções, mas na maioria dos casos bom senso é suficiente para determinar o que é um problema de usabilidade. Para uma definição geral de problema de usabilidade, pode-se dizer que é qualquer aspecto de um design onde uma mudança pode melhorar uma ou mais medidas de usabilidade.

A identificação dos problemas na interface é importante, mas é apenas uma parte de um grande processo. Depois de ser gerada a lista de problemas de usabilidade, a equipe de desenvolvimento precisa fazer um redesign da interface de modo a tentar corrigir a maior quantidade possível de problemas. Para que isso seja feito serão precisos outros tipos de informações e análises dentro do contexto mais amplo da engenharia de usabilidade.

Métodos de inspeção de usabilidade são geralmente melhores na detecção de problemas do que na direção de como melhorar a interface, mas tipicamente relatórios gerados a partir dos métodos contêm sugestões para redesign. Em muitos casos, conhecendo sobre o problema de usabilidade, é clara a maneira de corrigi-lo. Além disso, muitos dos métodos sugerem encontros entre a equipe de avaliadores e a equipe de desenvolvimento, quando esta é distinta, para discutir soluções de redesign.

O uso efetivo de uma lista de problemas de usabilidade irá requerer que esses problemas sejam priorizados com relação à gravidade de cada problema. Prioridades são necessárias para não se dispendem esforços desproporcionais corrigindo problemas que não irão alterar em muito a interação do usuário com a interface. **Graus de severidade** são geralmente derivados do impacto gerado pelo problema tanto no usuário quanto no mercado. Por serem muitas vezes critérios dependentes da aplicação, a definição de graus de severidade não é muito bem estabelecida na literatura, mas alguns autores dão exemplos de como atribuir graus de severidade à problemas de usabilidade (Nielsen, 1994; Karat, 1994; Desurvire, 1994).

Precisa ser estimado o custo associado à implementação das sugestões de redesign. Certamente, problemas de usabilidade com alto grau de severidade devem ser corrigidos não interessando o quanto custem. Frequentemente, muitos dos problemas não muito graves podem ser corrigidos com alterações mínimas de código. Esse compromisso não pode ser considerado como parte do método de inspeção de

usabilidade, pois é preferível ter uma relação completa de todos os problemas sem o pré julgamento da viabilidade ou não da sua correção.

Concluindo, ressaltamos que métodos de inspeção podem ser aplicados em fases iniciais ou finais do design e o resultado é um relatório formal dos problemas identificados com recomendações para mudanças. Alternativamente, e fortemente recomendável, a inspeção pode resultar em uma discussão ou apresentação para os designers e gerentes do projeto. Avaliadores precisam estar sensíveis à habilidade profissional e ao alto grau de envolvimento da equipe de desenvolvimento, e as sugestões devem ser feitas com cuidado: é difícil para alguém que tem um contato inicial com o sistema efetuando uma inspeção entender todas as razões de design e história de desenvolvimento de uma interface. Daí os revisores anotarem os possíveis problemas para discussão, mas as decisões de como corrigir devem ser deixadas sob responsabilidade dos designers.

Uma inspeção pode demorar de 12hs até 40hs, dependendo da complexidade da interface e de seus procedimentos operacionais, além do contexto da tarefa que necessariamente deverá ser conhecido pelo avaliador.

MÉTODOS DE INSPEÇÃO

Dentre os diversos métodos de inspeção existentes podemos destacar:

- **Avaliação Heurística:** é feita a inspeção da interface tendo como base uma pequena lista de heurísticas de usabilidade. Esse método será discutido com mais detalhe na próxima seção deste capítulo.
- **Revisão de *Guidelines*:** a interface é analisada no sentido de verificar se está de acordo com uma lista de *guidelines* de usabilidade. Geralmente essa lista contém uma sequência de cerca de 1.000 *guidelines*, o que torna o uso desse método muito raro dada a expertise que é exigida de um revisor.
- **Inspeção de Consistência:** o avaliador verifica a consistência dentro de uma família de interfaces, quanto à terminologia, cores, *layout*, formatos de entrada e saída, e tudo o mais dentro da interface. Também é avaliado o material *online* de treinamento e de ajuda .
- **Percorso Cognitivo:** o avaliador simula o usuário "caminhando" na interface para executar tarefas típicas. Tarefas mais frequentes são o ponto inicial de análise, mas tarefas críticas, tais como recuperação de erro, também são percorridas. Percorso cognitivo foi desenvolvido para interfaces que podem ser aprendidas de forma exploratória, mas também são úteis em interfaces que

exigem muito treinamento. Esse método será discutido em seção próxima deste capítulo.

Segundo Nielsen (1993), os métodos de inspeção de usabilidade não exigem muito esforço de quem pretende usá-los e podem ser facilmente integrados aos mais variados esquemas de produção de software. Não é necessário modificar fundamentalmente o modo como sistemas são desenvolvidos e gerenciados de modo a obter grandes benefícios da inspeção de usabilidade. Os resultados são rápidos e fornecem concretas evidências de quais aspectos da interface devem ser aperfeiçoados.

Percurso cognitivo e avaliação heurística são os precursores dos métodos de inspeção de usabilidade e geralmente não exigem uma grande experiência ou longo treinamento para que possam ser utilizados. Além disso, a utilização deles proporciona uma relevante experiência educacional para designers novatos. A exposição a preocupações adicionais de usabilidade tem se mostrado um meio efetivo de incrementar a perspectiva e valor do desenvolvimento de software orientado para o usuário. Esses dois métodos estarão sendo apresentados com detalhe nas próximas seções deste capítulo.

AValiação HEURÍSTICA

A maioria dos métodos de inspeção terão um efeito significativo na interface final somente se forem usados durante o ciclo de vida do projeto. Infelizmente, isso ainda não é uma realidade. Muitos desenvolvedores consideram os métodos intimidadores, muito caros, difíceis e que necessitam muito tempo para serem aplicados (Nielsen, 1994). No sentido de inverter essa tendência Nielsen propõe a denominada engenharia econômica de usabilidade - *discount usability engineering* - (Nielsen, 1989; Nielsen, 1993) com métodos que são baratos, rápidos e fáceis de serem usados. Avaliação heurística é o principal método dessa proposta. Segundo o autor, é fácil (pode ser ensinada em 4hs); é rápida (cerca de 1 dia para a maioria das avaliações); e tão barata quanto se deseje.

COMO CONDUZIR UMA AVALIAÇÃO HEURÍSTICA

Deve ser vista como parte do processo de design interativo de uma interface. Ela envolve um pequeno conjunto de avaliadores examinando a interface e julgando suas características em face de reconhecidos princípios de usabilidade, denominados heurísticas.

De modo geral, é difícil de ser feita por um único avaliador, porque uma única pessoa nunca é capaz de encontrar todos os problemas de usabilidade de uma interface. A experiência tem mostrado que diferentes pessoas encontram diferentes problemas, e portanto se melhora significativamente os resultados da avaliação heurística utilizando múltiplos avaliadores. A recomendação é que se use de três a cinco avaliadores.

A avaliação heurística é feita em um primeiro momento individualmente. Durante a sessão de avaliação cada avaliador percorre a interface diversas vezes (pelo menos duas) inspecionando os diferentes componentes do diálogo e ao detectar problemas os relata associando-os claramente com as heurísticas de usabilidade que foram violadas. As heurísticas (Tabela 4.1 e Tabela 4.2), são regras gerais que objetivam descrever propriedades comuns de interfaces usáveis (Nielsen, 1994).

1. Diálogo simples e natural

- simples significa informação não irrelevante ou raramente utilizada
- natural refere-se à adequação à tarefa

2. Falar na linguagem do usuário

- usar conceitos do mundo do usuário
- não usar termos computacionais específicos

3. Minimizar a carga de memória do usuário

- não fazer com que o usuário tenha que relembrar coisas de uma ação em uma próxima ação
- deixar informação na tela até ela não ser mais necessária

4. Ser consistente

- seqüência de ações aprendidas em uma parte do sistema devem poder ser aplicadas em outras partes

5. Prover feedback

- dar conhecimento aos usuários do efeito que suas ações têm sobre o sistema

6. Saídas claramente marcadas

- se o usuário entra em uma parte do sistema que não lhe interessa, ele deve ser capaz de sair rapidamente sem estragar nada
- não colocar o usuário em armadilhas

7. Prover Shortcuts

- auxiliar o usuário experiente a evitar extensos diálogos e mensagens de informações que ele não quer ler

8. Mensagens de erro construtivas e precisas

- informar ao usuário qual foi o problema e como corrigí-lo

9. Prevenir erros

- sempre que encontrar uma mensagem de erro, verificar se aquele erro poderia ser evitado

TABELA 4.1 - LISTA ORIGINAL DE HEURÍSTICAS DE USABILIDADE (NIELSEN, 1990)

1. Visibilidade do status do sistema

- sistema precisa manter os usuários informados sobre o que está acontecendo, fornecendo um feedback adequado dentro de um tempo razoável

2. Compatibilidade do sistema com o mundo real

- sistema precisa falar a linguagem do usuário, com palavras, frases e conceitos familiares ao usuário, ao invés de termos orientados ao sistema. Seguir convenções do mundo real, fazendo com que a informação apareça numa ordem natural e lógica

3. Controle do usuário e liberdade

- usuários frequentemente escolhem por engano funções do sistema e precisam ter claras saídas de emergência para sair do estado indesejado sem ter que percorrer um extenso diálogo. Prover funções *undo* e *redo*.

4. Consistência e padrões

- usuários não precisam adivinhar que diferentes palavras, situações ou ações significam a mesma coisa. Seguir convenções de plataforma computacional

5. Prevenção de erros

- melhor que uma boa mensagem de erro é um design cuidadoso o qual previne o erro antes dele acontecer

6. Reconhecimento ao invés de relembração

- tornar objetos, ações e opções visíveis. O usuário não deve ter que lembrar informação de uma para outra parte do diálogo. Instruções para uso do sistema devem estar visíveis e facilmente recuperáveis quando necessário

7. Flexibilidade e eficiência de uso

- usuários novatos se tornam peritos com o uso. Prover aceleradores de formar a aumentar a velocidade da interação. Permitir a usuários experientes "cortar caminho" em ações freqüentes.

8. Estética e design minimalista

- diálogos não devem conter informação irrelevante ou raramente necessária. Qualquer unidade de informação extra no diálogo irá competir com unidades relevantes de informação e diminuir sua visibilidade relativa

9. Ajudar os usuários a reconhecer, diagnosticar e corrigir erros

- mensagens de erro devem ser expressas em linguagem clara (sem códigos) indicando precisamente o problema e construtivamente sugerindo uma solução.

10. Help e documentação

- embora seja melhor um sistema que possa ser usado sem documentação, é necessário prover *help* e documentação. Essas informações devem ser fáceis de encontrar, focalizadas na tarefa do usuário e não muito extensas.

TABELA 4.2 - VERSÃO REVISADA DAS HEURÍSTICAS (NIELSEN, 1993)

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

Adicionalmente ao conjunto de heurísticas gerais a ser considerada em cada componente do diálogo, o avaliador pode também considerar heurísticas específicas da categoria do produto sendo analisado, por exemplo, heurísticas derivadas da análise de produtos similares e seus resultados de uso.

Por exemplo, um sistema altamente dependente de menus ou com diálogo centrado na entrada de informações via formulários pode definir heurísticas que avaliem especificamente a usabilidade desses componentes mais relevantes. Um exemplo, é o mencionado por Romani e Baranauskas (1998) quando apresentam o resultado da avaliação heurística de um sistema denominado *Sistema de Acompanhamento e Avaliação de Rebanhos Leiteiros (ProLeite)*, que é usado na organização das informações de desempenho produtivo e reprodutivo dos animais de rebanhos leiteiros. O sistema possui grande quantidade de formulários para entrada de dados, possibilita consultas e emissão de relatórios. A partir dos resultados da avaliação heurística do sistema é mostrada a necessidade de um conjunto de heurísticas de categorias específicas, para captar aspectos específicos do domínio considerado.

- | |
|---|
| 1. Opções de menu significativas e agrupadas logicamente: O agrupamento e os nomes das opções do menu são pistas para o usuário encontrar a opção requerida. |
| 2. Facilidade no modo de operação: O sistema deve prover um modo de operação facilitado, principalmente para sistemas de entrada de dados. Tais sistemas são construídos a base de formulários de entrada de dados sendo importante minimizar a quantidade de toques do usuário. |
| 3. Agrupamento lógico e seqüencial dos campos: O sistema deve dispor os campos de forma lógica e seqüencial nos formulários. O agrupamento lógico facilita a entrada de dados tanto para usuários iniciantes quanto experientes. |
| 4. Diferenciação entre campos não editáveis, obrigatórios e opcionais: O sistema deve prover cor de fundo diferenciada para campos não editáveis sinalizando para o usuário que determinados campos em um formulário não precisam ser digitados. O sistema deve fornecer alguma forma de identificação para campos obrigatórios para auxiliar na entrada de dados pois evidencia quais campos devem ser preenchidos e quais podem ficar em branco. Facilita a entrada dos dados e acaba evitando erros. |
| 5. Permite identificação do tipo de dado e quantidade de caracteres: No sistema deve |

	estar em evidência para o usuário o tipo de dado para cada campo, através de mensagem explicativa (hint), formatação do campo (por ex. --/--/---- para datas) ou no próprio rótulo. O sistema deve alertar sobre a quantidade de caracteres possível por campo.
6.	Agilidade na movimentação do cursor: O sistema deve facilitar a mudança de um campo para outro nos formulários através de teclas de atalho além do mouse, como o TAB ou ENTER.
7.	Facilidade na correção de erros durante a entrada de dado: O sistema deve permitir a correção rápida de erros durante a entrada de dados através de teclas como DEL, BACKSPACE ou outra. Além disso, ele deve possibilitar overtyping nos campos e a remoção de um campo inteiro.
8.	Aproveitamento de dados entrados anteriormente: Para sistemas de entrada de dados, é extremamente importante o aproveitamento de dados na digitação de novos registros. Esta característica diminui consideravelmente a quantidade de toques por registro.
9.	Localização de informação rapidamente: O sistema deve permitir a localização de um registro específico durante a entrada do dado. Esta opção de localização deve possibilitar buscas elaboradas não apenas por um código ou chave que identifique o registro na base de dados.

TABELA 4.3 – HEURÍSTICAS ESPECÍFICAS (ROMANI E BARANAUSKAS, 1998)

Dentre as heurísticas específicas definidas listamos algumas na Tabela 4.3. Como pode ser observado as heurísticas específicas são um refinamento das heurísticas gerais, provendo somente meios de uma avaliação mais específicas dos componentes de um diálogo.

Como durante a avaliação heurística os avaliadores não estarão usando o sistema de fato, ou seja, para efetuar tarefas reais, é possível efetuar a avaliação em interfaces que existam apenas em papel e que ainda não foram implementadas.

Se o sistema é de domínio não específico ou para ser usado pela população em geral ou os avaliadores são peritos no domínio, nenhuma assistência adicional é necessária. Caso contrário, será preciso prover meios de auxiliar o avaliador de forma a que ele seja capaz de usar o sistema adequadamente. Uma forma é ter sempre uma pessoa da equipe de desenvolvimento disponível para responder perguntas dos avaliadores. Outra maneira, é prover cenários típicos de uso (ver Capítulo 3), listando os vários passos que um usuário deveria efetuar para realizar um conjunto de tarefas reais. Tal cenário deve ser construído com base na análise da tarefa dos usuários reais de modo a ser tão realístico quanto possível. Esta é uma abordagem usada com frequência e com bastante sucesso.

Como dissemos, o resultado de uma avaliação heurística é uma lista de problemas de usabilidade da interface com referências aos princípios de usabilidade que foram violados. O avaliador não pode simplesmente dizer que não gosta de um determinado aspecto, tem que justificar com base nas heurísticas e tem também que ser o mais específico possível e listar cada problema encontrado separadamente. Geralmente a avaliação heurística não objetiva prover meios de corrigir os problemas ou um modo de avaliar a qualidade de um redesign. Entretanto, como ela explica cada problema encontrado referenciando as respectivas heurísticas que foram violadas, geralmente não é difícil gerar um design revisado baseado nas diretrizes que foram providas pelo princípio de usabilidade violado.

EXEMPLOS DE PROBLEMAS ENCONTRADOS NA AVALIAÇÃO HEURÍSTICA

Estaremos apresentando alguns problemas isolados detectados em avaliações heurísticas, de forma a deixar mais claro o tipo de resultados que podem ser obtidos pelo método. Importante observar que muitas vezes um único componente de diálogo fere mais que uma heurística e isso será indicado em nossos exemplos.

Exemplo 1

Em um sistema para oferecimento de cursos a distância denominado TelEduc¹, a ferramenta de correio eletrônico não permite o envio de mensagens sem o preenchimento do campo assunto (*subject*). Essa característica é uma fonte de erros constante, pois a maioria dos sistemas de correio permite isso, emitindo apenas um aviso. A heurística violada é a de **prevenção de erros**. Também deve ser observada a mensagem de erro inadequada - **Você não informou o assunto da correspondência** - usando o termo correspondência que não é natural no contexto de mensagens (Figura 4.2). Heurística violada - **consistência e padrões**. A recuperação do erro também é difícil, pois ao retornar à tela de composição da mensagem ela tem que ser completamente refeita. Portanto, mais uma heurística violada - **ajudar os usuários a reconhecer, diagnosticar e corrigir erros**.

Exemplo 2

Outro exemplo, extraído do componente **Lista de Discussão** da mesma ferramenta de ensino a distância do exemplo anterior. No sistema TelEduc as telas têm sempre o mesmo formato: dividida em dois *frames*, o da esquerda sempre está presente e contém a relação de todas as ferramentas disponíveis no ambiente e o da direita muda conforme a ferramenta escolhida. Quando se seleciona Lista de Discussão, a direita aparece uma

¹ http://www.nied.unicamp.br/tele_educ

Ao se selecionar a opção Ver que possibilita ler as mensagens de uma determinada lista a tela é substituída e se tem acesso às mensagens trocadas. E nessa tela temos uma coleção de heurísticas não respeitadas, algumas delas:

- **Estética e design minimalista; consistência e padrão** - existem dois componentes que têm exatamente a mesma função. Um é a flecha no topo direito da tela e outro o botão em baixo a direita. Um deles deve ser eliminado. A flecha tem o problema adicional de confundir com a flecha de *back* da maioria dos *browsers* (**prevenção de erros**). E o botão Voltar tem que ser mais específico indicando para onde vai voltar (ou então pode-se pensar que simplesmente é um *back* análogo ao do *browser*, o que também é ocasiona a quebra da heurística **prevenção de erros**).
- **Compatibilidade do sistema com o mundo real** - as mensagens podem ser vistas em diversas ordens (cronológica, alfabética por remetente, etc.) mas dificilmente um usuário não especialista vai saber o que é em *ordem de árvore* com relação ao título da mensagem
- **Reconhecimento ao invés de relembração; consistência e padrão**- a pergunta mais freqüente dos usuários é em como fazer para alterar a ordem das mensagens. Dificilmente se apercebem que basta um clique no rótulo do campo da mensagem que a ordem é alterada. Deveria ser provida uma forma que tornasse visível essa funcionalidade. Também importante que outras ferramentas análogas (como o correio exemplificado anteriormente) consistentemente apresentassem a mesma funcionalidade (pois depois de aprendida na lista, a funcionalidade vira fonte de erro dada a inconsistência entre as ferramentas análogas).

Exemplo 3

O antivírus Norton 2000 para NT Server é um software projetado para proteger servidores de rede Windows NT de arquivos infectados por vírus. O software é executado no servidor NT e sempre que se tenta gravar/abrir algum arquivo infectado no servidor, o programa apresenta uma mensagem na tela do servidor avisando que o arquivo está infectado, mas os usuários em estações cliente NT não recebem esse aviso. Tem-se portanto a heurística **visibilidade do status do sistema** violada. Consequentemente, quando o usuário trabalhando em uma estação cliente tenta gravar um arquivo infectado no servidor o antivírus impede a gravação e não emite nenhuma mensagem para a estação cliente e o usuário pode então perder seu trabalho sem saber que isso está ocorrendo. O resultado é perda do arquivo, o que teria sido facilmente prevenido se alguma mensagem de alerta fosse enviada à estação cliente. Claramente a **heurística ajudar os usuários a reconhecer, diagnosticar e corrigir erros** também é violada.

Exemplo 4

No sistema Windows quando se quer instalar um novo componente de hardware é iniciado um processo de busca e o indicador de detecção pode ficar parado por muito tempo, como indicado na janela de diálogo (Figura 4.4). O usuário fica perdido na maioria das vezes por não saber o que significa esse muito tempo e não sabe se deve ou não reiniciar o computador, mesmo porque usuários de sistemas semelhantes sabem quão pouco confiável é a relação da barra de detecção com o real andamento da operação (**visibilidade e status do sistema; ajudar os usuários a reconhecer, diagnosticar e corrigir erros**).



FIGURA 4.4 - DETECÇÃO DE HARDWARE DO SISTEMA WINDOWS

Exemplo 5

No Dia das Crianças o provedor ZAZ (<http://www.zaz.com.br>) fez uma página especial que dava entrada para uma página de jogos (Figura 4.5). Nessa página uma coleção de termos técnicos é encontrada e as crianças, potencialmente usuários novatos, não conseguem entender o que tem que fazer para atingir seu objetivo, que é jogar (**compatibilidade do sistema com o mundo real**).



FIGURA 4.5 - SITE DO PORTAL ZAZ NO DIA DA CRIANÇA DE 1999

Exemplo 6

O software Winzip em uma mesma versão gratuita (não registrada) tem uma tela de abertura onde os botões aparecem em ordem aleatória a cada execução (Figura 4.6). Arbitrariamente os botões Accept e Quit aparecem em ordem trocada levando o usuário a errar sem saber porque (**consistência e padrões; prevenção de erros**).



FIGURA 4.6 - TELA DE ABERTURA DA CÓPIA PARA AVALIAÇÃO DO SOFTWARE WINZIP

Exemplo 7

No sistema de entregas da declaração de Imposto de Renda ao tentar desistir do programa de instalação o usuário recebe uma caixa de diálogo onde o Não está a direita do Sim (Figura 4.7). Isso foge completamente ao padrão de diálogo de toda aplicação Windows em caixas de diálogo do tipo Sim/Não sendo fonte constante de erro (**consistência e padrões; prevenção de erros**). Vale a pena observar que esse programa é da categoria de softwares de uso eventual e mesmo que internamente esteja sendo adotado o padrão de colocar a escolha “mais provável” em primeiro lugar, o padrão Windows deveria ter sido respeitado, pois o usuário não irá usar o software

muito tempo de forma a poder aprender esse padrão interno específico. Outro aspecto a ser observado é o uso indevido da palavra abortar na caixa de diálogo (**compatibilidade do sistema com o mundo real**).



FIGURA 4.6 - PÁGINA DE INSTALAÇÃO DO SOFTWARE DE ENVIO ELETRÔNICO DA DECLARAÇÃO DE IMPOSTO DE RENDA

Exemplo 8

No Windows Explorer ao tentarmos excluir um arquivo que está em uso uma caixa de diálogo é aberta (Figura 4.7). Nessa caixa aparece a mensagem de que não foi possível acessar o arquivo, e recomenda ao usuário que verifique se o disco está cheio ou protegido, e finalmente se o arquivo não está sendo usado. Não há usuário que não se confunda: o que tem a ver disco cheio com excluir um arquivo! (**ajudar os usuários a reconhecer, diagnosticar e corrigir erros**)

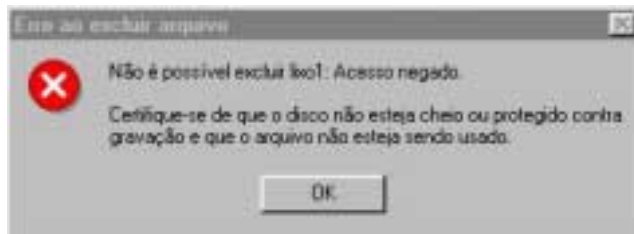


FIGURA 4.7 MENSAGEM DE ERRO DO WINDOWS EXPLORER

Exemplo 9

A página de entrada do site do AltaVista² tem uma enorme redundância de links para a função de busca (Figura 4.8). O usuário efetivamente não sabe qual usar, quando na verdade todas conduzem ao mesmo resultado (**estética e design minimalista**).



FIGURA 4.8 - PÁGINA DE ENTRADA DO SITE ALTA VISTA EM 10/04/2000

Exemplo 10

O Microsoft Word possui, no canto inferior esquerdo da tela, 4 botões, que servem para selecionar o modo de exibição do texto: normal, *layout online*, *layout* da página, e estrutura de tópicos (Figura 4.9).

Através desses botões o usuário pode alterar facilmente o modo de exibição de seu documento. Mas quando o modo de *layout online* é selecionado, os botões desaparecem, e o usuário fica sem saber como retornar ao modo de exibição anterior (Figura 4.10). O retorno só pode ser feito indo no menu Exibir e selecionando um dos outros modos (**reconhecimento ao invés de relembração**)

² <http://www.altavista.com>

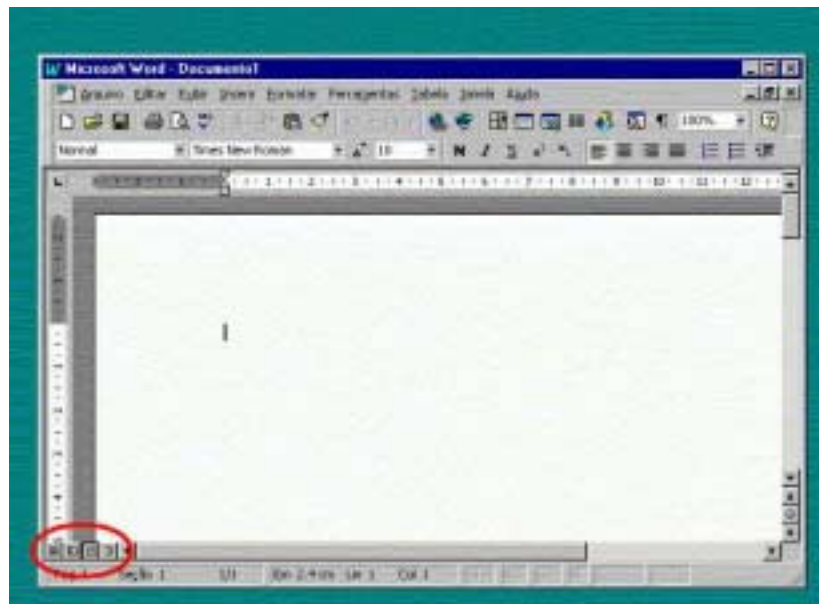


FIGURA 4.9 - TELA DO EDITOR DE TEXTOS WORD PARA WINDOWS

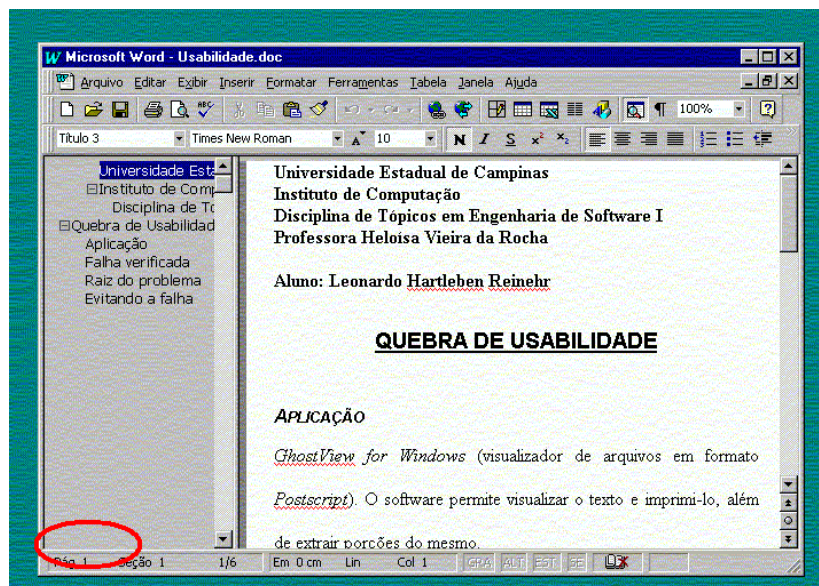


FIGURA 4.10 - TELA DO EDITOR DE TEXTOS WORD PARA WINDOWS MOSTRANDO TEXTO NO FORMATO LAYOUT

Exemplo 11

No Editor do Netscape quando se está editando um documento html, constantemente é necessária a visualização do documento no *browser*, para se testar links ou visualizar a forma do documento. O mesmo ocorre quando se está no *browser* e se deseja editar o documento (Figura 4.11). Portanto estas duas opções são bastante usadas por desenvolvedores de páginas, logo sempre deveriam ser oferecidos atalhos para essas opções (a partir do *browser* e a partir do editor), o que não ocorre no produto em questão (**flexibilidade e eficiência de uso**)

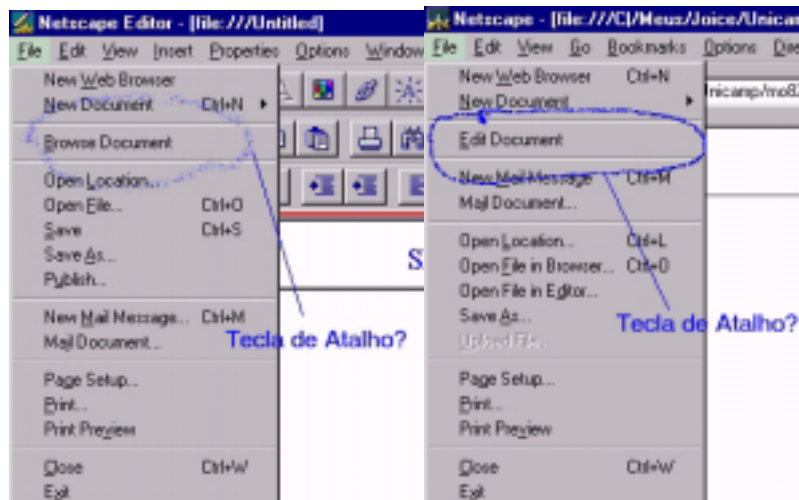


FIGURA 4.11 - TELAS DO BROWSER NETSCAPE COM MENUS DO MODO COMPOSER E NAVIGATOR

Exemplo 12

O Help do sistema Netscape não é muito extenso (Figura 4.12). Ele não preserva nenhum contexto, ou seja, sempre é aberta a mesma tela, independente do usuários estar no Navigator ou Mail, por exemplo (**help e documentação**). No índice à esquerda da tela, o usuário não sabe quando deve dar um clique na palavra ou na bola (que é o link) ao lado da palavra (três primeiras palavras da tela), ou seja, o usuário não identifica onde está o link, inclusive na parte de baixo da tela as palavras é que são os links (**consistência e padrões; prevenção de erros**). Ainda nessa tela tem-se a opção *Find* e a opção *Look For* que têm exatamente a mesma funcionalidade. O mesmo acontece com o X na barra em baixo da tela cuja funcionalidade é a mesma do X na barra inicial (padrão para fechar janelas). Isso confunde o usuário achando que o X em baixo é para sair de tudo e não simplesmente fechar a janela(**estética e design minimalista**).



FIGURA 4.12 - SISTEMA DE HELP DO BROWSER NETSCAPE

Com esses exemplos, podemos de forma mais clara perceber que o diagnóstico de um problema associado com as heurísticas que foram consideradas fonte do problema efetivamente traz possibilidades concretas de redesign, apesar desse não ser o objetivo da avaliação.

Como dissemos, uma possibilidade para estender o método de avaliação heurística, de forma a prover efetivas soluções de redesign, é fazer uma sessão de discussão final envolvendo a equipe de avaliadores e representantes da equipe de desenvolvimento. Essa discussão não precisa ser formalmente conduzida e deve estar focalizada nos principais problemas de usabilidade. Outro ponto que fortalece a existência dessa reunião é que nela podem ser levantados os aspectos positivos do design, pois a avaliação heurística não trata desse importante aspecto. Aspectos positivos são todas as características importantes e boas em uma interface que de forma alguma deveriam ser alteradas ou eliminadas em um redesign.

GRAUS DE SEVERIDADE

Adicionalmente à lista de problemas de usabilidade detetados, a avaliação heurística pode ser usada para avaliar a gravidade de cada problema. Esta informação é importante no momento em que forem alocados recursos para corrigir os problemas mais sérios e se necessário deixar os menos graves para uma nova versão.

A gravidade de um problema é a combinação de três fatores:

- **a frequência** com que ele ocorre: se é comum ou raro
- **impacto** do problema quando ele ocorre: se é fácil ou difícil para o usuário superá-lo
- **a persistência** do problema: problema que ocorre uma única vez e que o usuário pode superar desde que saiba que ele existe, ou se os usuários serão repetidamente incomodados por ele

Finalmente, é claro que é preciso considerar o **impacto do problema no mercado**, pois muitos problemas simples de serem superados tem um efeito importante na popularidade de um produto.

Dada essa diversidade de componentes, usualmente se faz uma tabela onde todos aparecem combinados de modo bastante subjetivo como pode ser visto na Tabela 4.4.

-
1. eu não concordo que isso é um problema de usabilidade
 2. é um problema cosmético somente - precisa ser corrigido somente se sobrar algum tempo no projeto
 3. problema de usabilidade menor - corrigí-lo deve ter prioridade baixa
 4. problema de usabilidade grave - importante corrigí-lo, deve ser dada alta prioridade
 5. catástrofe de usabilidade - a sua correção é imperativa antes do produto ser liberado

TABELA 4.4 - LISTA DE GRAUS DE SEVERIDADE DE PROBLEMAS ENCONTRADOS EM UMA AVALIAÇÃO HEURÍSTICA (ADAPTADO DE NIELSEN, 1994, P. 49)

Difícilmente se terá uma tabela de valores que expresse de maneira objetiva os componentes de gravidade de um problema para todos os sistemas. Elas são construídas quase que caso-a-caso e será a experiência dos avaliadores que irá determinar a coerência na atribuição de valores. Nielsen (1994) apresenta uma série de resultados procurando certificar a confiança nos julgamentos de severidade. Ele conclui que a taxação de severidade feita por um único avaliador não é confiável. Quanto mais avaliadores julgam a gravidade de problemas a qualidade cresce rapidamente, e taxação feita por três ou quatro avaliadores é satisfatória para a maioria dos problemas práticos.

Como a atribuição de graus de severidade não é possível de ser feita enquanto não se tem uma relação completa dos problemas, depois da sessão de avaliação é apresentado ao conjunto de avaliadores uma lista de graus de severidade e a relação completa de todos os problemas encontrados e cada avaliador atribui individualmente graus de severidade aos problemas.

CARACTERÍSTICAS DE PROBLEMAS DE USABILIDADE ENCONTRADOS PELA AVALIAÇÃO HEURÍSTICA

Avaliação heurística tem se mostrado um bom método para determinar tanto problemas graves como problemas menores de usabilidade. Certamente os problemas mais sérios são mais fáceis e os mais importantes de serem identificados, mas um grande valor da avaliação heurística está na detecção dos chamados problemas menores - como inconsistência tipográfica entre componentes do diálogo - que dificilmente seriam detectados usando outros métodos de avaliação e que muitas vezes prejudicam a interação do usuário (tornando-a mais lenta por exemplo no caso da inconsistência tipográfica).

Problemas de usabilidade em um diálogo podem ser localizados de quatro maneiras diferentes: em um único local da interface, em dois ou mais locais que devem ser comparados para se detectar o problema, um problema da estrutura geral da interface e como algo que precisa ser incluído na interface. Não existem diferenças significativas na quantidade de problemas localizados de alguma das maneiras, o que indica que avaliadores têm igual facilidade de localizar qualquer um dos tipos de problemas. A diferença que existe é no estágio de implementação da interface e o tipo de problema que pode ser detectado. Por exemplo, componentes de diálogo que precisariam ser incluídas (que faltam) são difíceis de serem detectadas em interfaces em estágio muito primário de desenvolvimento (em papel, por exemplo).

Concluindo, pode-se dizer que avaliação heurística é o método básico da engenharia de usabilidade e é relativamente fácil de ser usado e de ser aprendido. A descrição que fizemos apresenta alguns aspectos como a avaliação de interfaces de domínio

muito específico e atribuição de graus de gravidade que podem parecer um pouco complicadas de início por exigirem uma certa expertise dos avaliadores. Mas o mais importante é que se pode começar a fazer avaliação heurística sem considerar esses aspectos e sem muita experiência com avaliação.

Segundo Nielsen (1994) os principais componentes de uma avaliação heurística podem ser resumidos como:

- avaliadores devem percorrer a interface pelo menos duas vezes. Na primeira vez devem se concentrar no fluxo e na segunda nas componentes individuais do diálogo.
- a interface deve ser inspecionada com base em uma lista de princípios de usabilidade, as denominadas heurísticas, e todos os problemas devem ser justificados e detalhados o máximo possível.
- combinar os problemas encontrados por 3 a 5 avaliadores e fazer com que trabalhem individualmente (sem que um influencie o outro)

Depois do trabalho individual o ideal é ter uma reunião final de discussão, incluindo representantes da equipe de desenvolvimento de forma a se ter sugestões para redesign. E graus de severidade podem ser atribuídos aos problemas caso haja necessidade de priorizar a correção de problemas.

Usar o método, não somente melhora a interface sob análise, como também beneficia futuros projetos, o que é um efeito colateral da inspeção que julgamos extremamente importante.

PERCURSO COGNITIVO

Percurso cognitivo (Lewis *et al.* 1990; Polson *et al.* 1992a) é um método de inspeção de usabilidade que tem como foco principal avaliar o design quanto à sua facilidade de aprendizagem, particularmente por exploração.

O foco na aprendizagem foi motivado por resultados de estudos que apontavam que usuários preferem aprender a usar um software por exploração (Carroll and Rosson, 1987; Fisher, 1991). Ao invés de investir tempo em treinamento formal ou leitura de extensivo material de apoio, usuários preferem aprender sobre um software enquanto trabalham em suas tarefas usuais, adquirindo conhecimento sobre as características do software à medida que delas necessitem. Esta abordagem de aprendizagem incremental de certa forma assegura que o custo da aprendizagem de uma determinada característica é em parte determinado pelo seu benefício imediato ao usuário.

O que estamos pretendendo nesta seção é prover uma descrição detalhada de como se faz um percurso cognitivo, em que fase do desenvolvimento do sistema, etc. Portanto, não estaremos explorando os aspectos teóricos que subsidiaram o desenvolvimento do método, que podem ser vistos em Polson *et al.* (1992a).

UMA PRIMEIRA DESCRIÇÃO

Percurso cognitivo é um processo de revisão no qual o autor de um aspecto do design apresenta uma proposta para um grupo de pares. Os pares então avaliam a solução usando critérios apropriados ao design específico.

Os revisores avaliam a interface proposta no contexto de uma ou mais tarefas do usuário. A entrada para uma sessão de percurso inclui uma descrição detalhada da interface (na forma de um protótipo executável ou uma maquete em papel), o cenário da tarefa, suposições explícitas sobre a população de usuários e o contexto de uso, e a seqüência de ações que o usuário terá que fazer para executar corretamente a tarefa. O processo de percurso pode ser dividido em duas fases básicas, fase preparatória e fase de análise, cujas características podem ser vistas resumidamente na Tabela 4.5.

Fase preparatória

- Analistas definem tarefas, seqüências de ações para cada tarefa, população de usuários e a interface a ser analisada
1. Quem serão os usuários do sistema?
 2. Qual tarefa (ou tarefas) devem ser analisadas?
 3. Qual é a correta seqüência de ações para cada tarefa e como pode ser descrita ?
 4. Como é definida a interface?

Fase de análise

- Objetiva contar uma estória verossímil que informe sobre o conhecimento do usuário e objetivos, e sobre o entendimento do processo de solução de problemas que leva o usuário a "adivinhar" a correta solução. Analistas respondem 4 questões:
 1. Os usuários farão a ação correta para atingir o resultado desejado?
 2. Os usuários perceberão que a ação correta está disponível?
 3. Os usuários irão associar a ação correta com o efeito desejado?
 4. Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação a tarefa desejada?
- uma estória verossímil de fracasso será contada se algumas das questões acima tiver resposta negativa

TABELA 4.5 - PROCESSO DE PERCURSO COGNITIVO (ADAPTADO DE WARTON, C. ET AL . 1994, P.106)

Durante o processo de percurso o grupo de avaliadores considera, em seqüência, cada uma das ações necessárias para completar a tarefa. Para cada ação, os analistas tentam *contar uma estória* sobre interações típicas de usuários com a interface. Eles perguntam o que o usuário tentaria fazer nesse ponto a partir das ações que a interface deixa disponíveis. Se o design da interface for bom, a intenção do usuário fará com que ele selecione a ação apropriada e tenha conhecimento disso, ou seja, em seguida à ação a interface deverá apresentar uma resposta clara indicando que progresso foi feito na direção de completar a tarefa.

DESCRIÇÃO DETALHADA DO PROCEDIMENTO DE PERCURSO

Como já dissemos o percurso cognitivo tem duas fases principais: fase preparatória e fase de análise. Na fase preparatória, os analistas definem as condições de entrada para o percurso: as tarefas, seqüência de ações para cada tarefa, população de usuários e a interface que será objeto de análise. O principal trabalho analítico é feito na segunda fase, durante o qual os analistas trabalham em cada ação de cada uma das tarefas escolhidas. Os detalhes de cada fase, em particular as informações a serem registradas, dependem fortemente de como o percurso vai ser usado no processo de desenvolvimento.

Formalmente, o percurso cognitivo é um método de inspeção para avaliar um design quanto ao aspecto de facilidade de aprendizagem por exploração. Ele pode ser efetuado depois de uma especificação detalhada da interface do usuário, a qual acontece depois da análise de requisitos e definição da funcionalidade de uma aplicação. Um percurso também pode ser efetuado em uma simulação em papel da interface, ou em um protótipo mínimo construído com qualquer ferramenta de prototipação ou ainda em um protótipo completo de um design.

O percurso pode ser um processo individual ou em grupo. Em grupo, o designer apresenta a interface para um grupo de pares, tipicamente após fases significativas do design como a construção de um primeiro protótipo, e usa o feedback da avaliação para modificar ou fortalecer a próxima revisão. Os pares podem incluir outros designers, engenheiros de software, e representantes de outras unidades organizacionais como marketing, documentação e treinamento. Adicionalmente pode ser incluído ao grupo um especialista em avaliação de interfaces. Cada membro do grupo de avaliadores tem um papel específico: o escriba, o facilitador e os demais contribuem com um conhecimento específico: conhecimento do mercado potencial, análise das necessidades do usuário, etc.

Um indivíduo pode também usar o percurso cognitivo para avaliar um design pessoal. Como o percurso é baseado no modelo explícito do processo de aprendizagem por exploração, desenvolvedores participando de seu próprio (ou de outro grupo) percurso também têm a possibilidade de internalizar o conhecimento

sobre o processo associado ao modelo teórico que o embasa. Esse conhecimento pode influenciar em próximas decisões de design que o desenvolvedor tenha que tomar. Portanto, o processo de avaliação pode ser usado em fases bastante iniciais do processo de design por designers individuais, desenvolvedores ou grupos de designers.

De modo geral, percurso cognitivo tem um impacto positivo em todas as fases do design e do processo de desenvolvimento. Muitas vezes as tarefas do usuário, centrais na aplicação, escolhidas para avaliação são utilizadas como carros-chefe de teste de campo e propaganda.

DEFININDO AS ENTRADAS PARA O PERCURSO - FASE PREPARATÓRIA

Antes da fase de análise quatro aspectos devem estar plenamente acordados:

- *Quem são os usuários do sistema?*

Pode ser uma descrição simples e geral tal como, "*peças que usam LINUX*". Mas o processo de percurso é mais revelador se a descrição inclui mais especificamente a experiência e conhecimento técnico que podem influenciar os usuários na interação com uma nova interface. Por exemplo, usuários podem ser "*Usuários de Windows que trabalham com o Microsoft Word*". No processo de percurso são considerados o conhecimento do usuário com relação à tarefa e com relação à interface.

- *Qual tarefa (ou tarefas) será analisada?*

O percurso envolve a detalhada análise de uma ou várias tarefas. É possível fazer a análise de todas as tarefas associadas a um sistema com funcionalidade simples, como um sistema de compactação de arquivos, por exemplo. Também para um sistema uma única tarefa pode ser analisada, por exemplo a que se mostrou problemática em versões anteriores. Em geral, para sistemas com alguma complexidade, a análise deverá ser limitada a uma razoável, mas representativa, coleção de tarefas.

A questão crítica é como selecionar essas tarefas. Seleção de tarefas deverá ser baseada em resultados de estudo de mercado, análise de necessidades, análise de conceito e análise de requisitos. Algumas tarefas devem ser escolhidas como exemplo a partir da funcionalidade central da aplicação, isto é, operações básicas que se pretende que o sistema deva suportar. Adicionalmente, outras tarefas que requeiram uma combinação dessas funções básicas podem ser consideradas.

As tarefas selecionadas devem ser o mais concretas e realistas possível. A descrição da tarefa deve conter o contexto necessário, por exemplo, o conteúdo das bases de dados que se espera que o usuário vá utilizar. Esse contexto reflete condições típicas sob as quais o sistema irá operar. Por exemplo, em uma tarefa de recuperação de

dados de uma base de dados, a base de dados usada como exemplo deverá ser grande ou pequena, dependendo do uso pretendido do sistema.

- *Qual a correta seqüência de ações para cada tarefa e como é descrita?*

Para cada tarefa, deve haver uma descrição de como se espera que o usuário veja a tarefa antes de aprender sobre a interface. Também deve haver uma descrição da seqüência de ações para resolver a tarefa na atual definição da interface. Essas ações podem incluir movimentos simples como "*pressionar a tecla ENTER*" ou "*mover o cursor para o menu File*", ou, podem ser seqüências de diversas ações simples que o usuário típico pode executar como um bloco, tais como "*Logar no sistema*" para usuários experientes em UNIX, ou selecionar o "*Salvar como do menu File*" para usuários experientes em Windows. A decisão sobre a granularidade da descrição depende muito da expertise do usuário alvo. A grosso modo, pode-se definir que a descrição deve ser equivalente a descrição que seria colocada em um tutorial eficiente.

- *Qual a interface definida?*

A definição da interface precisa descrever os *prompts* que precedem cada ação requerida para completar as tarefas que estão sendo analisadas como também a reação da interface para cada uma de suas ações. Se a interface já está implementada, toda informação estará disponível na implementação. Entretanto, algumas características importantes do sistema são difíceis de serem apreciadas como, por exemplo, o tempo de resposta, cores, temporização em interfaces com fala, e interações físicas.

Para uma interface descrita em papel, o detalhamento da definição da interface irá depender da experiência pretendida dos usuários alvo com sistemas que já existem. Por exemplo, a preparação para analisar uma aplicação Windows dirigida para usuários experientes em Windows, não precisa prover uma descrição detalhada da aparência padrão de menus Windows; uma descrição simples de seus conteúdos é suficiente.

PERCORRENDO AS AÇÕES - FASE DE ANÁLISE

A fase de análise do percurso consiste em examinar cada ação do caminho da solução e tentar contar uma *estória verossímil* de como o usuário iria escolher aquela ação. Estórias verossímeis são baseadas em suposições sobre objetivos e conhecimento do usuário, e no entendimento do processo de solução de problemas que possibilitará ao usuário escolher a ação correta.

O processo de solução de problemas foi descrito por Polson and Lewis em sua teoria sobre aprendizagem exploratória (Polson and Lewis, 1990). Brevemente, este processo de solução de problemas estabelece que usuários: (1) iniciam com uma

descrição grosseira da tarefa que tem que efetuar (2) exploram a interface e selecionam as ações que eles imaginam as mais adequadas para efetuar a tarefa ou parte dela (3) observam a reação da interface para verificar se suas ações tiveram o efeito desejado e (4) determinam qual ação efetuar a seguir.

A teoria também aponta algumas heurísticas específicas que os usuários aplicam quando tomam suas decisões. Em particular, usuários frequentemente seguem a estratégia de *"label-following"*, a qual os faz selecionar uma ação se o rótulo para essa ação combina com a descrição da tarefa (Engelbeck, 1986). Por exemplo, se o usuário deseja "imprimir um arquivo" poderá selecionar uma ação com o rótulo (ou ícone) "imprimir" ou "arquivo".

As características críticas da interface, são portanto, aquelas que provêm uma ligação entre a descrição do usuário para a tarefa e a ação correta, e aquelas que provêm feedback indicando o efeito da ação do usuário. Conforme o percurso vai avançando o analista aplica essa teoria ao relatar e avaliar sua estória de como o usuário escolheria a ação prevista pelo designer em cada passo. Os analistas ao contarem suas estórias devem responder a quatro questões:

- *Os usuários farão a ação correta para atingir o resultado desejado?*

Suponha que em uma determinada aplicação antes de mandar imprimir um documento é preciso selecionar uma determinada impressora. O usuário irá saber que tem que fazer isso antes de executar a tarefa de impressão?

- *Os usuários perceberão que a ação correta está disponível?*

Se a ação estiver disponível no menu e for facilmente identificada não há problema. Mas suponha que para imprimir um documento seja necessário dar um clique em um ícone com o botão esquerdo do mouse. O usuário pode não pensar nunca nisso.

- *Os usuários irão associar a ação correta com resultado desejado?*

Se existe um item de menu claro e facilmente encontrado informando "Selecionar Impressora" então não há problemas, mas se no menu só tem a opção "ImpSis" aí as coisas talvez não sejam tão evidentes.

- *Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação a tarefa desejada?*

Se após a seleção o usuário tiver um feedback informando "Impressora Laser XXX da sala YY selecionada" então sem problemas. O pior caso é a ausência de resposta.

Essas questões servem de guia para construir as estórias, não sendo requisitos obrigatórios, mas são as mais usadas. Podem ser definidos pelo analista outros critérios que o levem a contar estórias verossímeis. No caso de seguir o critério das quatro questões acima, a falha em qualquer uma das questões implicará em problemas com a interface.

REGISTRO DA INFORMAÇÃO DURANTE A AVALIAÇÃO

Durante o percurso é importante registrar toda informação gerada. Para uma avaliação feita em grupo, muito comum no caso de percurso cognitivo, é recomendado que sejam usados materiais visíveis ao grupo. Também é muito conveniente gravar na forma de vídeoteipe todo o processo de avaliação, incluindo comentários dos avaliadores. O vídeo é uma importante fonte de referência para eximir dúvidas, verificando comentários e decisões tomadas durante o processo de avaliação. O material visível ao grupo serve para registrar e resumir todas as decisões e informações chave para o grupo.

Existem diversas informações importantes que devem ser registradas de forma visível ao grupo durante o processo de avaliação. Indispensáveis são as informações sobre o conhecimento esperado do usuário, suposições sobre a população de usuários, notas sobre mudanças de design e a história verossímil desenvolvida durante o processo de percurso. São sugeridas três instâncias de registro: uma para os pontos chave da história do grupo, outra para catalogar toda informação sobre o usuário e uma terceira que registre notas sobre mudanças de design e outras observações paralelas importantes.

Quanto às informações sobre os usuários deve-se ter o registro para cada classe de usuários, das seguintes informações:

que o usuário precisa conhecer antes de executar a tarefa
que o usuário poderá aprender enquanto executa a tarefa

ESTÓRIAS DE SUCESSO E ESTÓRIAS DE FRACASSO

Para se ter uma idéia mais clara do tipo de histórias que analistas geram durante um percurso, iremos apresentar alguns pequenos exemplos de histórias de sucesso e de fracasso verossímeis, de interfaces que existem ou existiram efetivamente.

EXEMPLOS DE ESTÓRIAS DE SUCESSO

Estória de sucesso 1:

Um usuário experiente em Windows inicia uma tarefa dando um clique no ícone da aplicação para abrí-la.

Defesa da Credibilidade

- usuário abre a aplicação porque ele sabe que deve abrir uma aplicação para usá-la
- usuário conhece por experiência que pode dar clique sobre o ícone da aplicação

- usuário sabe por experiência que o clique é a ação a ser usada
- mudanças na tela ou na barra de menu sinalizam o início da aplicação

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

Estória de sucesso 2:

Um usuário experiente em Windows abre o menu Tabela para preparar uma tabela em um editor de textos.

Defesa da Credibilidade

- usuário está tentando preparar uma tabela pois essa é a tarefa
- usuário abre o menu Tabela pois o título Tabela é claramente relacionado com o que ele está querendo fazer
- usuário sabe que pode abrir um menu e essa é a ação a ser feita se o título parece bom, por experiência com o Windows
- usuário sabe que tudo está indo bem quando da leitura das opções do menu.

Estória de sucesso 3:

Um usuário de cartão de crédito está usando o sistema telefônico para obter informações sobre seu saldo. O sistema diz "entre com o número de seu cartão" e o usuário disca seu número.

Defesa de Credibilidade

- usuário entra com o número porque o sistema informou isso para ele
- usuário usa as teclas do seu telefone porque elas estão visíveis e não existe outra possibilidade para entrar o número
- usuário conhece o número do cartão pois é dele o cartão e ele tem então acesso ao número
- usuário sabe que as coisas estão indo bem quando o sistema começa a dizer os números de um menu de opções de serviço.

CARACTERÍSTICAS COMUNS DE SUCESSOS

Com esses exemplos em mente, podem ser resumidos os quatro pontos que os analistas consideram em cada passo e suas características mais relevantes:

1. Usuários irão conhecer **Qual resultado querem alcançar:**
 - Porque é parte da tarefa original, ou

- Porque eles têm experiência no uso do sistema, ou
 - Porque o sistema diz a eles o que devem fazer
2. Usuários irão saber que **Uma ação está disponível**:
 - Por experiência, ou
 - Observando algum dispositivo, ou
 - Observando a representação de uma ação
 3. Usuários irão saber **Qual ação é adequada** para o resultado que estão tentando obter:
 - Por experiência, ou
 - Porque a interface provê um *prompt* ou rótulo que conecta a ação ao que ele está tentando fazer, ou
 - Porque as outras ações não parecem corretas
 4. Usuários irão saber que **As coisas estão indo bem** depois da ação:
 - Por experiência, ou
 - Por reconhecer a conexão entre a resposta do sistema e o que ele está tentando fazer

Muitos desses pontos enfatizam a importância de se conhecer como o usuário descreve a tarefa. Quando o sistema usa a mesma terminologia que o usuário, ele provavelmente irá escolher a ação correta. O mesmo acontece com a resposta do sistema (feedback) quando é dada na linguagem do usuário. Quando termos não usuais são utilizados pelo sistema, o usuário poderá encontrar dificuldades em obter sucesso sem um conhecimento adicional.

EXEMPLOS DE ESTÓRIAS DE FRACASSOS

Estórias de sucesso requerem sucesso em todos os quatro critérios de análise, enquanto estórias de fracasso tipicamente falham em apenas um dos quatro critérios implicando em que uma estória verossímil não possa ser contada.

Vamos então agrupar nossos exemplos de estórias de fracasso de acordo com o critério em que falham.

- **Critério 1:** *Os usuários farão a ação correta para atingir o resultado desejado?*

Exemplo 1:

Considerando um usuário de Windows com familiaridade com editores gráficos básicos, que ao usar um editor de *gifs* animados (Figura 4.13) tenta dar um *zoom in* na figura sob edição. Para isso ele tem que alterar o número que está na caixa de diálogo à esquerda, no topo da tela (inicialmente com 100%).

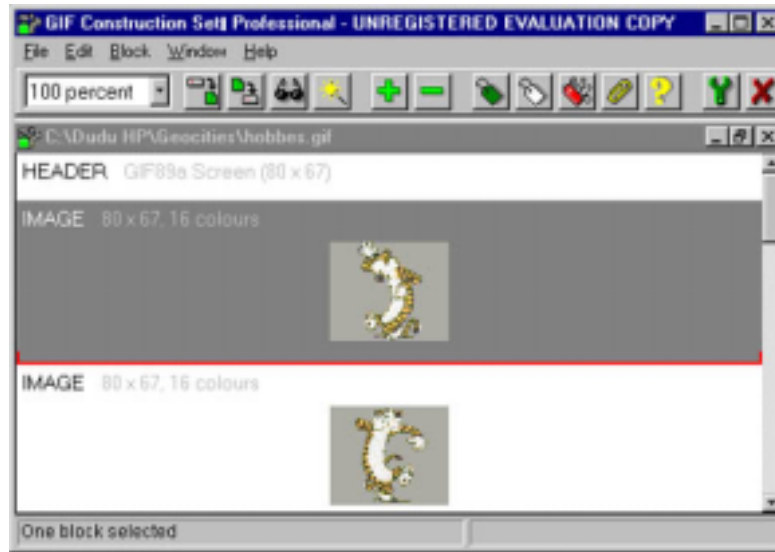


FIGURA 4.13 - PÁGINA DE SOFTWARE DE EDIÇÃO DE GIFS ANIMADOS

Estória de fracasso: Os usuários irão acionar diretamente o botão com o sinal de menos pois esse é o padrão de ícone usado na maioria dos editores para dar *zoom in*. Não obteriam portanto o resultado desejado pois o respectivo botão elimina o último *gif* editado da sequência de *gifs*.

Exemplo 2:

Considerando um usuário experiente em Windows e com conhecimentos básicos do editor de textos Word. Ele quer numerar as páginas de um texto colocando a numeração no seguinte formato *pg – número*, centralizado no topo da página. Para fazer isso ele terá que ir ao menu View, selecionar a opção Header and Footer e aí editar o formato desejado da numeração em uma caixa de entrada que é aberta no texto, conjuntamente com uma caixa de ferramentas nomeada Header e Footer (figura 4.14)

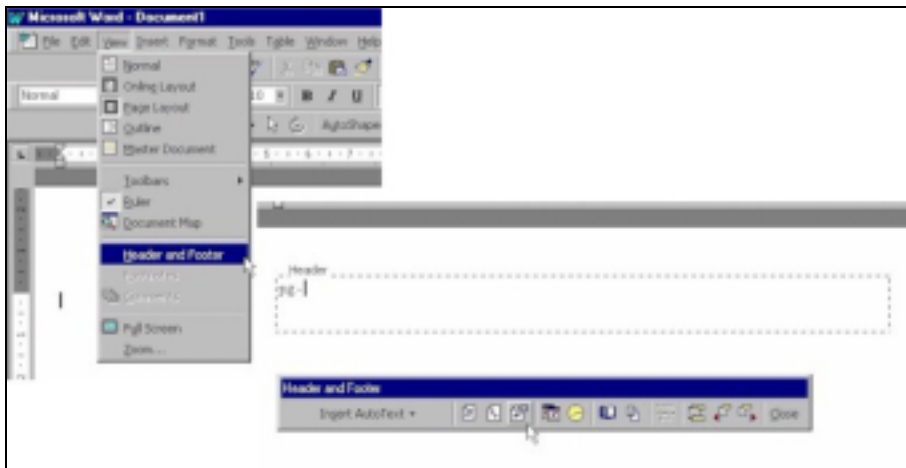


FIGURA 4.14 TELAS DO EDITOR DE TEXTOS WORD PARA WINDOWS COM MENUS VIEW E TELA DE EDIÇÃO DE HEADER

Estória de fracasso: Dificilmente o usuário irá associar a ação de numerar páginas em um determinado formato com a opção *View*. Ele irá certamente escolher o item de menu *Insert* com a opção *Page Numbers* e *Format Page Numbers* em seguida, falhando em sua ação.

- **Critério 2:** Os usuários perceberão que a ação correta está disponível?

Exemplo 1:

Em um particular programa que gera gráficos (pizza, barras, etc.), para se alterar a fonte e outras especificações do título de um gráfico deve-se dar um duplo clique no título do gráfico e somente dessa forma, irá abrir uma caixa de diálogo que permitirá a edição desejada. Considerando usuários experientes em interfaces gráficas.

Estória de fracasso: os usuários frequentemente não consideram a possibilidade de usar um duplo clique nesse contexto

Exemplo 2:

No sistema operacional Windows 95 para se desligar o computador deve-se dar um clique no botão *Iniciar*. Considerando-se usuários novatos em Windows. (Figura 4.15)

Estória de fracasso: Usuários principiantes não descobrem que essa é a ação a ser adotada para se desligar corretamente o computador.

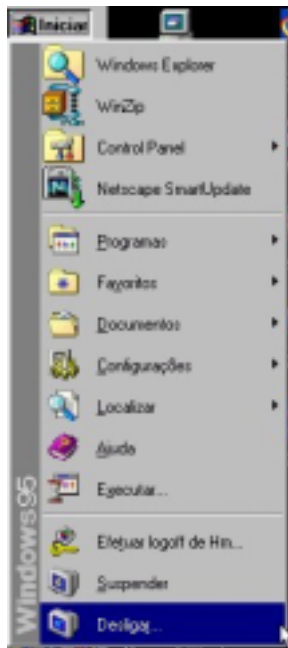


FIGURA 4.15 - MENU INICIAR DO WNDOWS 95

Com frequência, sistemas orientados por comando têm falhas com relação a esse critério. Usuários geralmente sabem o que querem fazer (por exemplo, abrir um diretório, mudar a proteção de um arquivo, listar um arquivo, etc), mas eles muitas vezes não sabem, e nem sempre conseguem encontrar, o nome do comando.

- **Critério 3:** *Os usuários irão associar a ação correta com o efeito desejado?*

Exemplo 1:

Em um processador de textos, orientado a menus, existem dois menus. Um ítem de menu é chamado FORMAT e outro é chamado FONT. Estilo de fontes são parte do menu FORMAT. Considerando usuários iniciantes.

Estória de fracasso: Os usuários não saberão qual menu escolher se quiserem colocar uma palavra em itálico.

Exemplo 2:

Considerando o exemplo mencionado anteriormente sobre a colocação de número de páginas no processador de textos Word do Windows no formato *pg - número da página*

Estória de fracasso: Mesmo considerando que os usuários saibam que esse formato de numeração tenha que ser colocado sob a forma de Header, dificilmente irão associar a operação de inserir um Header com o menu View.

- **Critério 4:** *Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação à tarefa desejada?*

Exemplo:

No sistema Windows NT, para desligar o computador deve ser acionado o menu iniciar da mesma forma que no sistema Windows (Figura 4.15). Depois de selecionada a opção correta aparece na tela uma caixa de diálogo para o usuário iniciar novamente o sistema.

Estória de fracasso: Usuários muitas vezes não percebem que saíram do sistema com sucesso. Ao invés disso, alguns usuários automaticamente dão um clique ou enter e se vêm presos em um *loop*.

Alguns outros problemas que devem ser observados durante um percurso e que poderão resultar em estórias de fracasso:

Time outs: alguns sistemas, especialmente os baseados em telefone ou que requisitam senhas (como os caixas automáticos por exemplo), dão ao usuário um tempo determinado para tomar a ação. Tentar determinar se o tempo dado está adequado considerando o usuário alvo.

Ações difíceis fisicamente: apertar teclas simultaneamente, especialmente quando devem ser pressionadas ou soltas em uma determinada ordem, é difícil. Também a dificuldade em selecionar um componente muito pequeno na tela com o mouse ou toque. Problemas desse tipo existem em editores gráficos que muitas vezes exigem um controle motor muito fino para desenhar uma simples linha.

Terminadores de ação: usuários geralmente esquecem ações que indicam que alguma parte da tarefa está concluída, como colocar um ";" ou "." após um comando em uma linguagem de programação. Isso porque para o usuário é óbvio o término. Isso ocorre mesmo com usuários muito experientes que sabem o que deve ser feito e mesmo assim estão sempre tendo que efetuar correções.

COMO USAR OS RESULTADOS DO PERCURSO PARA CORRIGIR PROBLEMAS

Observamos que falhas são tipicamente associadas com um dos quatro critérios de análise; sugestões gerais de como corrigi-las também podem ser organizadas da mesma forma. Consideremos, portanto, cada um dos critérios:

- *Os usuários farão a ação correta para atingir o resultado desejado?*
Se a interface falha nesse ponto - ou seja, o usuário não está fazendo a coisa correta - existem pelo menos três opções para tentar corrigir: (1) a ação precisa ser eliminada, ou o sistema a efetua (por exemplo, no caso de se ter obrigatoriamente que escolher uma impressora antes de efetuar uma impressão) ou deve ser combinada com outra ação mais adequada (2) um *prompt* deve ser dado ao usuário informando-o sobre qual ação deve ser feita (3) alguma outra parte da tarefa precisa ser mudada de forma a que o usuário entenda a necessidade da ação, talvez porque passe a ser consistente com alguma outra parte da sequência de ações.
- *Os usuários perceberão que a ação correta está disponível?*
Se o usuário tem os objetivos corretos mas não sabe que a ação está disponível na interface, a solução é associar a ação a um controle mais óbvio. Isso tipicamente envolve o uso de um menu ou *prompt*, ao invés de uma tecla de controle; ou pode envolver associar a ação a um controle mais escondido mas que pode ser facilmente descoberto por estar significativamente agrupado.
- *Os usuários irão associar a ação correta com o efeito desejado?*
Para corrigir essa falha os designers precisam conhecer seus usuários e a forma como descrevem a tarefa. Com essa informação o designer pode prover rótulos e descrições para ações que incluam palavras que os usuários freqüentemente usam para descrever suas tarefas. Pode ser necessário redefinir rótulos ou outros controles que os usuários selecionam em preferência aos corretos. Também pode envolver um reagrupamento de funções em menus de forma mais significativa com relação à visão que o usuário tem da tarefa.
- *Se a ação correta for executada os usuários perceberão que foi feito um progresso em relação à tarefa desejada?*
Claramente, na maioria das situações qualquer feedback é melhor que nenhum feedback. E feedback que indica o que aconteceu é sempre melhor que um feedback que indica que alguma coisa aconteceu. Respostas são sempre mais efetivas quando usam termos (ou gráficos) relacionados à descrição do usuário para a tarefa.

De maneira geral, lidar com o problema eliminando ou reagrupando ações é sempre preferível a tentar corrigir os problemas usando *prompts* ou feedback. Quando a

interface apresenta uma série de problemas que indicam uma concepção de design que não confere com a descrição que o usuário tem da tarefa, o designer precisa avaliar a possibilidade de corrigir esses problemas efetuando uma reorganização global da interface ao invés de ficar corrigindo problemas locais.

ESCOPO E LIMITAÇÕES DO MÉTODO

Percurso cognitivo, como já dissemos, enfoca apenas um atributo de usabilidade, a facilidade de aprendizagem. Mas deve ser considerado que outros atributos de usabilidade são fortemente conectados à facilidade de aprendizagem, como funcionalidade e facilidade de uso.

Entretanto, o uso de percurso cognitivo como o único método para avaliar uma interface pode conduzir o design à um forte compromisso com a facilidade de aprendizagem. Por exemplo, o processo de percurso pode conduzir à uma avaliação negativa de características que objetivem o aumento de produtividade se essas características tornarem mais difícil decidir como efetuar uma determinada tarefa.

Percurso cognitivo avalia cada passo necessário para a realização de uma tarefa com o objetivo de descobrir erros de design que podem dificultar a aprendizagem por exploração. O método encontra os conflitos entre designers e usuários sobre a concepção da tarefa, escolhas pobres de palavras de menus e rótulos de botões, e respostas inadequadas sobre conseqüências de ações. O método também mostra as suposições explícitas e implícitas feita pelos desenvolvedores sobre o conhecimento do usuário sobre a tarefa e das convenções da interface. O procedimento de avaliação toma a forma de uma série de questões que são feitas sobre cada passo dentro de uma tarefa derivadas da teoria de aprendizagem por exploração (Polson *et al.* 1992a). Não existem relatados dados estatísticos que informem o número de avaliadores a serem usados de forma a se obter confiabilidade nos resultados das avaliações. Por ser uma avaliação muito detalhada ela quase sempre é realizada durante o processo de design avaliando sempre pequenos cenários de interação que se apresentem críticos com relação a característica de aprendizagem. Resultados muito satisfatórios são conseguidos com avaliações feitas com grupos de 3 a 5 avaliadores trabalhando conjuntamente. Existem software, especificamente desenvolvidos para o processo de percurso cognitivo, que apresentam formulários pré-formatados e que são usados durante as avaliações, tanto na fase preparatória como durante a avaliação.

Mais uma vez, reforçamos que todos os métodos têm suas características fortes e fracas. O forte compromisso do percurso cognitivo com a facilidade de aprendizagem sacrifica a obtenção de informação válida sobre outros importantes atributos de usabilidade como, por exemplo, consistência global da interface. Isso reforça a necessidade de se usar diversos métodos complementares de forma a garantir uma ampla cobertura da interface em todas as suas características relevantes.

Portanto, todo método de avaliação não pode ser considerado completo e sim complementar suas características com outros métodos.

TESTE DE USABILIDADE

Teste com usuário é um método fundamental de usabilidade. Desenvolvedores tradicionais resistem à idéia, dizendo que teste de usabilidade sem dúvida alguma é uma boa idéia, mas limitações de tempo e de recursos os impedem de fazê-lo. Mas esse cenário está mudando muito rapidamente. Gerentes de desenvolvimento estão percebendo que ter agendado testes de usabilidade é um poderoso incentivo para o término da fase de design. E a surpresa é que resultados práticos têm demonstrado que testes de usabilidade não somente têm acelerado muitos projetos como também têm produzido uma significativa redução em seus custos (Gould and Lewis, 1985; Gould *et al.*, 1991; Karat, 1994).

O movimento em direção aos testes de usabilidade estimulou a construção de laboratórios de usabilidade (Dumas and Redisch, 1993; Nielsen, 1993). A IBM fez em Boca Raton, Flórida, uma sofisticada construção com 16 laboratórios dispostos em círculo com uma base de dados centralizada para registrar a performance e o *log* de uso de produtos testados. Muitas outras empresas seguiram o exemplo e abraçaram a idéia com bastante força. Isso demonstra a crescente preocupação com o usuário, que está permeando a maioria dos desenvolvimentos atuais.

Um laboratório de usabilidade geralmente abriga uma pequena equipe de pessoas com experiência em teste e design de interface de usuário. A equipe do laboratório geralmente entra em contato com representantes da equipe de desenvolvimento no início de um projeto, de forma a estabelecer um plano de teste com datas definidas e custos alocados. Ela também participa na fase inicial de análise da tarefa e revisão de design, fazendo sugestões e provendo informações, e ajudando no desenvolvimento do conjunto de tarefas para o teste de usabilidade.

A disponibilidade de um laboratório não deve ser considerada condição para a realização de um teste de usabilidade e sim como uma grande facilitação. Quase todas as formas de teste podem ser feitas nos mais diversos locais, desde que devidamente preparados. Também não deve ser considerada uma condição a existência de avaliadores experientes para se efetuar um teste. Bons resultados têm sido obtidos com experimentadores novatos que aprendem o método de teste (Nielsen, 1992; Wright and Monk, 1991)

OBJETIVOS E PLANO DE TESTE

Antes de qualquer teste ter início é preciso estabelecer seus objetivos pois isso tem um impacto significativo no tipo de teste a ser feito. A principal distinção é se o teste tem como objetivo obter uma ajuda no desenvolvimento ou é um teste que visa avaliar a qualidade global de uma interface. No primeiro caso interessa saber em detalhe quais aspectos da interface estão bons ou ruins, e como o design pode ser melhorado. É uma forma mais gradual de analisar a interface, e nesse caso usualmente se aplica o teste denominado *pensar em voz alta* (*thinking-aloud test*) que veremos a seguir. No segundo caso, como se quer uma visão mais global de uma interface em fase final de definição geralmente se utiliza testes que dêem medidas de performance que apresentaremos em seções a seguir. Em qualquer uma das situações deve ser desenvolvido um plano detalhado de teste onde, dentre outras mais específicas, as seguintes questões devem ser respondidas:

- O objetivo do teste: o que se deseja obter?
- Quando e onde o teste irá acontecer?
- Qual a duração prevista de cada sessão de teste?
- Qual o suporte computacional necessário?
- Qual software precisa estar a disposição?
- Qual deverá ser o estado do sistema no início do teste?
- Quem serão os experimentadores?
- Quem serão os usuários e como serão conseguidos?
- Quantos usuários são necessários?
- Quais as tarefas que serão solicitadas aos usuários?
- Qual critério será utilizado para definir que os usuários terminaram cada tarefa corretamente?
- Quanto o experimentador poderá ajudar o usuário durante o teste?
- Quais dados serão coletados e como serão analisados uma vez que tenham sido coletados?
- Qual o critério para determinar que a interface é um sucesso? (p. ex: *nenhum problema de usabilidade novo com severidade maior ou igual a 3*)

Deve-se sempre estar atento a dois problemas vinculados a um teste de usabilidade: a **confiabilidade** e a **validade**. Como confiabilidade entendemos o grau de certeza de que o mesmo resultado será obtido se o teste for repetido; e como validade, o fato dos resultados de teste refletirem os aspectos de usabilidade que se deseja testar.

No quesito **confiabilidade** deve-se estar atento às diferenças individuais entre os usuários. Por exemplo, cuidar do grau de confiança que se dá a afirmações do tipo:

“usuário A usando a interface X executa uma tarefa 40% mais rápido que o usuário B usando a interface Y”

que não necessariamente significam que a interface A tem melhor qualidade, pois não é incomum ter-se um grupo de usuários onde o melhor usuário é 10 vezes mais

rápido que o mais lento e que os 25% melhores são 2 vezes mais rápidos que os 25% piores (Egan, 1988)

Quanto à **validade**, o que se gostaria de assegurar é que o resultado obtido tenha realmente significado considerando-se o produto real em uso e fora da situação de laboratório. Deve-se então nesse ponto estar atento à escolha dos usuários, à escolha das tarefas e à diferença entre equipamentos (situação de teste e situação real)

A regra principal para se efetuar a **escolha dos usuários** é que sejam tão representativos quanto possível com relação aos usuários reais do sistema. O ideal seria envolver usuários reais do sistema, mas isso nem sempre é possível. Se o grupo de sujeitos não é composto de usuários reais, ele deve ter idade e nível educacional similar ao grupo de usuários alvo. Também similar deve ser sua experiência com computadores, com o tipo de sistema que está sendo testado e o conhecimento do domínio da tarefa. Certamente não é conveniente testar uma interface voltada para o público em geral e utilizar estudantes de computação como grupo de teste: eles certamente não são representativos da população de usuários alvo.

Os usuários devem ser tratados com respeito e principalmente serem informados de que é a interface e não eles que estão sendo testados. Geralmente se sentem sob muita pressão na situação de teste e isso os leva a aprenderem mais lentamente e a fazerem mais erros, sentindo-se estúpidos quando experimentam dificuldades.

Os experimentadores devem ser preparados no sentido de terem conhecimento extenso sobre a aplicação e a respectiva interface de usuário. Não precisam saber como o sistema foi implementado, mas devem estar prontos a lidar com problemas que afetem o teste, por exemplo, problemas que levem o sistema a cair. Nada impede que sejam os próprios designers desde que esses estejam preparados no sentido de manter uma certa isenção no sentido de não mascarar os resultados do teste. Geralmente eles tendem a ajudar muito os usuários e a antecipar situações de erro, dado seu extenso conhecimento da interface.

As tarefas a serem feitas durante um teste devem ser as mais representativas possíveis e devem dar uma cobertura razoável das partes mais significativas da interface. Devem poder ser completadas no tempo definido para uma sessão de teste (de 1 a 3 horas). Devem ter grau de dificuldade gradativa para dar mais confiança ao usuário e devem ser planejadas para que possam ser interrompidas a qualquer tempo, caso o usuário assim o deseje. A descrição de cada tarefa a ser efetuada deve ser feita por escrito e deve ser tão realista quanto possível e inserida em um cenário de uso.

Geralmente, um teste piloto é efetuado com um pequeno grupo (de 1 a 3) de usuários, para refinar todos os procedimentos definidos.

ETAPAS DE UM TESTE

Basicamente um teste é composto de quatro etapas:

- *Preparação*
Nessa etapa se garante que tudo estará pronto antes do usuário chegar. Muito cuidado deve ser tomado com relação aos equipamentos que serão utilizados, devem estar "limpos" (de resultados de outros teste, alarmes sonoros, etc).
- *Introdução*
É uma fase muito importante, onde os usuários são apresentados à situação de teste e de alguma forma colocados a vontade. Alguns pontos que devem ser falados aos usuários nessa introdução podem ser destacados:
 - O propósito do teste é avaliar o sistema e não o usuário
 - Não devem se preocupar em ferir sentimentos dos experimentadores (designers) com suas observações
 - Os resultados do teste servirão para melhorar a interface do usuário
 - Relembrar que o sistema é confidencial e não deve ser comentado com outros (que inclusive podem vir a ser futuros usuários em outros testes)
 - A participação no teste é voluntária e podem parar a qualquer tempo
 - Os resultados do teste não serão colocados publicamente e o anonimato do participante estará garantido
 - Explicar sobre o uso de gravações de vídeo ou audio que estarão sendo feitas (o ideal é não gravar a face do usuário)
 - Explicar que podem fazer qualquer pergunta durante o teste, mas que nem sempre o experimentador irá ajudá-los ou responder suas questões
 - Instruções específicas sobre a forma do teste (p. ex.: falar em voz alta, ou fazer as atividades o mais rápido que puder, etc.)
- *Teste*
Durante o teste deve ser escolhido somente um experimentador para falar com o usuário, para evitar confusão, e é importante que:
 - evite qualquer tipo de comentário ou expressões sobre a performance ou observações do usuário
 - evite ajudar o usuário, a não ser que ele esteja realmente em dificuldades muito graves
- *Sessão final*
Depois do tempo definido para completar as tarefas - usualmente de 1 a 3 horas - os participantes são convidados a fazerem comentários ou sugestões gerais, ou a responderem um questionário específico.

Gravar em vídeo os participantes efetuando as tarefas é sempre um recurso valioso para uma posterior revisão. Analisar vídeos é um trabalho extremamente difícil e

tedioso, portanto sempre devem ser feitas cuidadosas anotações ou coletados *logfiles* durante o teste de modo a reduzir o tempo dispendido em encontrar acontecimentos críticos (Harrison, 1991). A reação de designers vendo esses vídeos de usuários cometendo erros é muito poderosa e motivadora. Quando designers vêem usuários repetidamente acessando o menu errado, eles se convencem que rótulos e posições devem ser mudadas.

Uma técnica efetiva durante um teste de usabilidade é solicitar que os usuários *pensem em voz alta* sobre o que estão fazendo. A atmosfera informal de uma sessão que usa essa técnica é extremamente agradável, e freqüentemente leva à muitas sugestões espontâneas de melhorias.

PENSANDO EM VOZ ALTA

É uma técnica muito valiosa utilizada originalmente como um método de pesquisa psicológico. Solicita-se ao usuário que verbalize tudo que pensa enquanto usa um sistema e a expectativa é que seus pensamentos mostrem como o usuário interpreta cada item da interface (Lewis, 1982). Certamente é uma técnica não adequada quando se deseja medidas de performance. Geralmente os usuários ficam mais lentos e cometem menos erros quando pensam em voz alta.

O experimentador tem que ser bem preparado no sentido de levar o usuário a falar sempre e nunca interferir no uso do sistema pelo usuário. Formas de questionamento usuais podem ser relacionadas:

- O que você está pensando agora?
- O que você acha que essa mensagem significa (depois do usuário notar a mensagem)?
- Se o usuário pergunta se pode fazer alguma coisa: O que você acha que vai acontecer se fizer isso?
- Se o usuário se mostra surpreso: Era isso que você esperava que iria acontecer? O que esperava?

Os comentários dos usuários devem ser criteriosamente analisados e nunca aceitos indiscriminadamente pois podem dar falsa impressão das razões de um determinado problema. Os usuários têm teorias nem sempre verdadeiras.

A principal força dessa técnica é mostrar *o que* os usuários estão fazendo e *porque* estão fazendo *enquanto* estão fazendo, evitando as racionalizações posteriores.

No sentido de incentivar o pensar em voz alta muitas vezes se coloca usuários trabalhando aos pares de forma a produzirem mais conversas a medida que um

participante explica para o outro seus procedimentos, sem a inibição de estar falando com alguém que "sabe mais" sobre o sistema. Essa alternativa é muito usada em testes que envolvem crianças como sujeitos do teste.

Outra alternativa ao pensar em voz alta é fazer com que o usuário comente depois suas ações gravadas em vídeo. Isso auxilia quando se está também interessado em obter dados qualitativos de performance, mas deve ser levado em conta que todo o teste demora pelo menos o dobro do tempo.

MEDIDAS DE PERFORMANCE

Estudos de medidas quantitativas formam a base de muitas pesquisas tradicionais em fatores humanos, como visto no Capítulo 2. Também são importantes em usabilidade para avaliar se os objetivos de usabilidade foram efetivamente atingidos e também para comparar produtos competitivos.

Em usabilidade tem-se o critério de eficiência de uso como uma das *guidelines* de usabilidade. Dentro desse, são fundamentais algumas medidas de performance na forma de tomadas de tempo, por exemplo. Como e quando marcar tempos deve ser decidido *a priori* de acordo com os dados necessários na coleta. Por exemplo, se se quer saber quanto o usuário demora fazendo uma determinada tarefa é preciso primeiro definir quando começa e quando termina a tarefa e depois se o tempo será cronometrado pelo próprio usuário, pelo computador, pelo experimentados, etc.

Medidas típicas de usabilidade que são quantificáveis incluem:

- O tempo que o usuário gasta para fazer uma determinada tarefa
- O número de tarefas de diferentes tipos que são completadas em determinado limite de tempo
- A razão entre interações de sucesso e de erro
- O número de erros do usuário
- O número de ações errôneas imediatamente subsequentes
- O número de comandos (ou diferentes comandos) ou outras características que foram utilizados pelo usuário
- O número de comandos ou outras características nunca utilizados pelo usuário
- O número de características do sistema que o usuário consegue se lembrar na sessão subsequente ao teste
- A frequência de uso de manuais ou do sistema de *help* e o tempo gasto usando esses elementos do sistema
- Quão frequentemente o manual/sistema de *help* resolveu o problema do usuário
- A proporção entre comentários do usuário favoráveis e críticos com relação

ao sistema

- O número de vezes que o usuário expressou frustração (ou alegria)
- A proporção de usuários que disse preferir o sistema a outro sistema competidor
- A proporção de usuários utilizando estratégias eficientes e ineficientes
- A quantidade de “tempo morto” - quando o usuário não está interagindo com o sistema (ou esperando resposta ou pensando)
- O número de vezes que o usuário desviou do objetivo da tarefa

Certamente somente um pequeno subconjunto de medidas pode ser coletado em uma particular situação de teste.

A maioria dos testes de usabilidade são feitos em laboratórios onde os usuários são observados diretamente pelos avaliadores. Entretanto, a localização remota e distribuída dos usuários - freqüentemente na rede, atualmente o principal ambiente para distribuição e uso de aplicações de software, tais como os CSCW - dificulta, e até impede, a possibilidade da observação direta em testes de usabilidade. Também deve ser considerado que a rede em si e o trabalho remoto têm se tornado parte intrínseca de padrões de uso. Adicionalmente, desenvolvedores muitas vezes têm dificuldade em conseguir usuários representativos que possam participar de testes de usabilidade nos laboratórios; e o contexto de trabalho do usuário dificilmente consegue ser reproduzido em situação de laboratório. Todos esses fatores muitas vezes tornam o custo de um teste de usabilidade proibitivo.

Essas barreiras para o teste de usabilidade têm levado à uma extensão do teste para além dos limites dos laboratórios e começam a surgir métodos de teste de usabilidade remotos, tipicamente usando a rede como uma ponte de acesso aos usuários em seu ambiente natural de trabalho. Define-se portanto um teste de usabilidade remoto como aquele em que o observador que efetua a observação e análise e o usuário estão separados em tempo e espaço. Alguns resultados preliminares apontam para a efetividade desses métodos que usam tipicamente uma combinação dos mecanismos de comunicação eletrônicos, como tele-conferência por exemplo, e software especificamente construídos para coletar dados de uso (Hartson *et al*, 1996)

Temos também como modalidade de teste de usabilidade os denominados **testes de campo** que objetivam colocar novas interfaces em ambientes reais de uso por um determinado período de tempo. Testes de campo dão melhor resultado quando se pode dispor de software que geram arquivos *log* que capturam erros, comandos usados, freqüência de acesso a *helps* e mais algumas medidas de produtividade. Quando se conta somente com a resposta do usuário os resultados não são muito satisfatórios. Somente cerca de 20% dos usuários inscritos participam ativamente do teste; os demais estão mais interessados em obter uma primeira versão do produto. Um dos maiores testes de campo já realizados foi o efetuado pela Microsoft na avaliação da versão Beta do Windows 95, onde cerca de 400.000 usuários em todo o

mundo receberam versões preliminares do software e foram convidados a enviar comentários.

CONSIDERAÇÕES FINAIS

Designers e profissionais de IHC procuram por métodos rápidos e baratos de avaliação de interfaces em substituição aos testes de laboratório que geralmente são caros e muitas vezes sem possibilidade de serem realizados por falta de condições estruturais. Em virtude dessa situação, as técnicas de avaliação denominadas métodos de inspeção de usabilidade foram propostas com a promessa de oferecer informação de usabilidade de modo mais rápido e barato que os tradicionais testes de usabilidade. Os mais populares desses métodos incluem avaliação heurística e percurso cognitivo vistos anteriormente em detalhe neste capítulo.

Esses métodos, cada qual com seus procedimentos próprios, provêem dados que podem ser usados quando testes de usabilidade não são possíveis ou em conjunção com eles. Mas resultados de experimentos comparativos confirmam que até o momento eles não substituem os testes com usuário. Existe certamente um fator econômico a considerar na adoção de métodos de inspeção mas a relação custo-benefício precisa ser bem analisada.

Um amplo estudo (Desurvire, 1994) comparando testes de usabilidade, avaliação heurística e percurso cognitivo apresenta alguns resultados interessantes:

- Os resultados dos métodos de inspeção são melhores quando os avaliadores são especialistas em avaliação. Mesmo assim, não substituem o teste de usabilidade: nos experimentos relatados os melhores avaliadores, usando o método de melhor performance, não detetaram em média 56% dos problemas encontrados no teste de usabilidade.
- Avaliação heurística permite uma avaliação global da interface facilitando a identificação de melhorias na interface. Foi a mais eficaz na detecção de erros e principalmente na identificação da maioria de erros sérios. Além disso é a de menor custo.
- Teste de usabilidade é o mais eficaz em detetar erros, mas o mais caro. O custo de um teste de usabilidade é da ordem de 50 vezes o custo dos métodos de inspeção (isso está mudando com a introdução da metodologia de testes remotos). Todos os problemas sérios são encontrados mas perde na detecção de consistência
- Percurso cognitivo falha em identificar problemas gerais e recorrentes pois observa detalhes menores diretamente ligados à execução de uma tarefa. Foi

o que apresentou melhores resultados quando utilizado por não especialistas em usabilidade mas desenvolvedores de software.

Cada método tem pontos fortes e pontos fracos e ainda não se tem resultados substanciais de pesquisas comparando os métodos para que se possa efetivamente dizer qual é o melhor e em qual situação. Certamente cada situação de projeto irá requerer uma forma de avaliação. Acreditamos que o principal problema é a não disponibilidade de especialistas em usabilidade em situações de pequenas empresas desenvolvedoras de software (Chan e Rocha, 1996). Mas tendo como parâmetro que qualquer avaliação é melhor que nenhuma, ela deve ser feita de qualquer forma e a conseqüente geração de conhecimento em usabilidade será uma das melhores conseqüências.

Quanto à Web não existem ainda metodologias específicas para avaliação de Web sites. Conforme visto do Capítulo 1 *guidelines* começam a ser definidas (Nielsen, 1999) e irão possibilitar a definição de métodos de avaliação que levem em conta as especificidades desses design. Certamente os parâmetros de usabilidade permanecem e no decorrer do livro muitos exemplos extraídos de sistemas da Web foram mencionados, mas existe o componente da informação que certamente deve ser considerado em um processo de avaliação. Spool *et al.* (1999) relata alguns estudos de caso de avaliações de Web sites. Eles consideram que informação é o tema central e focalizam seus estudos em avaliar quanto um site é bom em prover informação às pessoas de forma a auxiliá-las a tomarem decisões. E concluem que quanto mais um site ajuda uma pessoa a encontrar a informação que está procurando mais usável ele é. De modo geral, são ainda resultados parciais quanto a usabilidade de sistemas baseados na Web e que ainda deverão ser mais aprofundados e formalizados.

REFERÊNCIAS:

Carrol, J. M., e Rosson, M. B. (1987) The paradox of the active user. Em J.M. Carrol (ed.) *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. Cambridge: Bradford Books/MIT Press

Desurvire, H. W. (1994) Faster, Cheaper!! Are usability Inspection Methods as Effective as Empirical Testing? Em J. Nielsen (ed.) *Usability Inspection Methods*. John Wiley, New York

Dix, A., Finlay, J., Abowd, G., Beale, R. (1998) *et al. . Human-Computer Interaction*. Prentice Hall Europe

Dumas, J. and Redish, J. (1993) *A Practical Guide to Usability Testing*. Ablex, Norwood, NJ

Egan, D. E. (1988) Individual differences in human-computer intercation. Em M. Helander (ed.). *Handbook of Human-Computer Interaction*. Amsterdam: Elsevier Science Publishers

Engelbeck, G. E. (1986) *Exceptions to generalizations: Implications for formal models of human-computer interaction*. Master Thesis. Department of Psychology, University of Colorado, Boulder

Fisher, G. (1991) Supporting learning on demand with design environments. *Proceedings of the International Conference on Learning Sciences* (Evanston, IL, August): 165-172

Gould, J. D., e Lewis, C. (1985) Designing for usability: Key principles and what designers think. *Communications of the ACM* 28, 3:300-311

Gould, J., Boies, S. J., e Lewis, C. (1991) Making usable, useful, productivity-enhancing computer applications. *Communications of the ACM* 34, 1: 74-85

Greenbaum, J., Kyng, M. eds. (1991) *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Lawrence Erlbaum Assoc.

Hartson, H. R., Castilho, J. C., Kelso, J. (1996) Remote Evaluation: The Network as an Extension of the Usability Laboratory. Disponível na Web em http://www.acm.org/sigchi/chi96/proceedings/papers/Hartson/hrh_texto.htm. Consulta 09/03/2000

Hix, D. and Hartson, H. R. (1993) *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley and Sons, New York

- Kahn, M.J., Prail, A. (1994) Formal usability Inspections. Em J. Nielsen (ed.) *Usability Inspection Method*. John Wiley, New York
- Karat, C. (1994) A Comparison of user Interface Evaluation Methods. Em J. Nielsen (ed.) *Usability Inspection Methods*. John Wiley, New York
- Lewis, C. (1982) Using the 'thinking-aloud' method in cognitive interface design. *IBM Research Report RC9265 (#40713)*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY
- Lewis, C., Polson, P., Wharton, C. and Rieman, J. (1990) Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. *Proceedings ACM CHI'90 Conference* (Seattle, WA, April 1-5):235-242
- Molich, R. and Nielsen, J. (1990) Improving a human-computer dialogue. *Communications of the ACM* 33,3: 338-348
- Monk, A., Wright, O., Haber, J., Davenport, L. (1993) *Improving your Human-Computer Interface: A Practical Technique*. New York: Prentice-Hall
- Nielsen J. (1989) Usability engineering at a discount. Em G. Salvendy *et al.* (eds.). *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*. Amsterdam:Elsevier Science Publishers
- Nielsen, J. (1992) Finding usability problems through heuristic evaluation *Proceedings ACM CHI'92 Conference* (Monterey, CA, May 3-7): 373-380
- Nielsen, J. (1993) *Usability Engineering*. Academic Press, Cambridge, MA
- Nielsen, J. (1994) Heuristic Evaluatin. Em J. Nielsen (ed.) *Usability Inspection Methods*. John Wiley, New York
- Nielsen, J. (1999) *Design Web Usability*. New Riders Publishing
- Polson, P. e Lewis, C. (1990) Theory-based design for easily learned interfaces. *Human Computer Interaction* 5, 2&3:191-220
- Polson, P., Lewis, C., Rieman, J. e Wharton, C. (1992a) Cognitive Walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36:741-773
- Preece, J., Sharp, H., Benyon, D., Holland, S., Carey, T. (1994) *Human-Computer Intercation*, Addison-Wesley

Spool, J., Scanlon, T. Schoeder, W. Snyder, C. e DeAngelo, T. (1999) *Web Site Usability: A Designers Guide*. Morgan Kaufmann Publishers

Romani, L. S. e Baranauskas, M.C.(1998) Avaliação heurística de um sistema altamente dependente do domínio. *Relatório Técnico* IC-98-26 , July. Disponível na Web em: <http://www.dcc.unicamp.br/ic-tr-ftp/ALL/Titles.html>

Chan, S. e Rocha, H.V. (1996) Estudo comparativo de métodos para avaliação de interfaces homem-computador. *Relatório Técnico* IC-96-05, September. Disponível na Web em: <http://www.dcc.unicamp.br/ic-tr-ftp/ALL/Titles.html>

Schneiderman, B. (1998) *Design the User Interface: Strategies for Human-Computer Interaction*. Addison Wesley Longman, Inc.

Wharton, C., Rieman, J., Lewis, C., Polson, P. (1994) The Cognitive Walkthrough: A Practitioner's Guide. Em J. Nielsen (ed.) *Usability Inspection Methods*. John Wiley, New York

Whitefield, A., Wilson, F., and Dowel, J. (1991) A framework for human factors evaluation. *Behaviour & Information Technology* 10, 1 (January-February), 65-79

Wright, P. C. and Monk, A. F. (1991) The use of think-aloud evaluation methods in design. *ACM SIGCHI Bulletin* 23, 1:55-71

