CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

RICARDO HISAO WATANABE

PROPOSTA DE UM MÉTODO FOCADO EM USABILIDADE PARA APLICAÇÕES WEB ADERENTE AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO 3º CENTRO DE TELEMÁTICA DE ÁREA

São Paulo

Setembro/2009

RICARDO HISAO WATANABE

PROPOSTA DE UM MÉTODO FOCADO EM USABILIDADE PARA APLICAÇÕES WEB ADERENTE AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO 3º CENTRO DE TELEMÁTICA DE ÁREA

Dissertação apresentada como exigência parcial para obtenção do Título de Mestre em Tecnologia no Centro Estadual de Educação Tecnológica Paula Souza, no Programa de Mestrado em Tecnologia: Gestão, Desenvolvimento e Formação, sob orientação do Prof. Dr. Marcelo Duduchi Feitosa.

São Paulo

Setembro/2009

Watanabe, Ricardo Hisao

W324p Proposta de um método focado em usabilidade para aplicações Web aderente ao processo de desenvolvimento de software do 3º Centro de telemática de área / Ricardo Hisao Watanabe. -- São Paulo: CEETEPS, 2009.
134 f.

Dissertação (Mestrado) - Centro Estadual de Educação Tecnológica Paula Souza, 2009.

1. Engenharia de usabilidade. 2. Software – desenvolvimento. 3. Web I. Título.

CDU 681.3.01

RICARDO HISAO WATANABE

PROPOSTA DE UM MÉTODO FOCADO EM USABILIDADE PARA APLICAÇÕES WEB ADERENTE AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO 3º CENTRO DE TELEMÁTICA DE ÁREA

PROF. DR. MARCELO DUDUCHI FEITOSA

PROF. DR. CARLOS HIDEO ARIMA

PROF. DR. ARISTIDES NOVELLI FILHO

São Paulo, 04 de setembro de 2009

Agradecimentos

Agradeço a Deus pela presença em todos os momentos da minha vida, especialmente aqueles momentos em que certamente Ele me carregou nos braços.

Muito obrigado a vocês, pessoas muito especiais na minha vida: Pierina, Daniel e Fernanda, saibam que eu não conseguiria sem o apoio, o carinho, o incentivo e principalmente a compreensão dos vários momentos de ausência.

Aos meus pais Wataru e Esther pelo eterno amor.

Agradeço ao meu professor orientador Marcelo Duduchi pela dedicação, motivação e confiança.

Um agradecimento muito especial à professora e amiga Cristiane pela motivação e dedicação com que me apoiou na construção deste trabalho.

Muito obrigado a todos vocês professores e colaboradores do programa de mestrado do Centro Paula Souza pelo exemplo, dedicação e atenção.

Aos meus colegas do curso de mestrado, principalmente o Carlos e o Vander pelo companheirismo e o mútuo incentivo que compartilhamos nesta caminhada.

Aos companheiros e amigos do 3º CTA em especial aos coronéis Rufino e Betat, ao major Roberto, aos tenentes Arthur, Glaucemara e Alfredo e aos sargentos De Paula, Nathaniel, Renato, Sidnei e Hermes.

Enfim agradeço a todas as pessoas que direta ou indiretamente acompanharam esta caminhada.

“*Agir, eis a inteligência verdadeira. Serei o que quiser. Mas tenho que querer o que for. O êxito está em ter êxito, e não em ter condições de êxito. Condições de palácio tem qualquer terra larga, mas onde estará o palácio se não o fizerem ali?*”

Fernando Pessoa

Resumo

WATANABE, R. H. Proposta de um Método Focado em Usabilidade para Aplicações Web Aderente ao Processo de Desenvolvimento de Software do 3º Centro de Telemática Área. 2009. 134f. Dissertação (Mestrado), Centro Estadual de Educação Tecnológica Paula Souza.

A Seção de Sistemas do 3º Centro de Telemática de Área (3º CTA), desde os anos 80 desenvolve sistemas corporativos para as Organizações Militares utilizando softwares proprietários. Porém, após o desenvolvimento e entrega do SASM - Sistema de Alistamento de Serviço Militar, para utilização das 2.185 Juntas de Serviço Militar distribuídas pelo Brasil, houve um grande número de dúvidas e problemas de uso do sistema pelos usuários. O processo de correção do SASM foi caro, consumiu muitas horas dos desenvolvedores e demora nas atualizações do sistema por parte dos usuários. Verificou-se que apesar do 3º CTA possuir seu Processo de Desenvolvimento de Software (PDS), este não disciplina as atividades de construção e avaliação de interfaces ou aborda as questões de facilidade de uso. Após a determinação, pelo Governo Federal, do uso de software livre e da priorização do formato Web no desenvolvimento de sistemas esta situação tornou-se mais complexa. Este estudo propõe um método para a concepção, desenvolvimento e avaliações de interfaces das aplicações Web aderente ao PDS do 3º CTA. O método proposto contempla a participação dos usuários no processo, especifica os artefatos a serem construídos, dirige as atividades da equipe de desenvolvimento por função e indica as técnicas e as avaliações de interface necessárias.

Palavras-Chave: Engenharia de Usabilidade, Aplicações Web, Processo de Desenvolvimento, Princípios e Diretrizes de Usabilidade.

Abstract

WATANABE, R. H. Proposta de um Método Focado em Usabilidade para Aplicações Web Aderente ao Processo de Desenvolvimento de Software do 3º Centro de Telemática Área. 2009. 134f. Dissertação (Mestrado), Centro Estadual de Educação Tecnológica Paula Souza.

The System Section of the 3º Centro de Telemática de Área (3º CTA), since the 80s develops corporative systems for the Military Organizations using proprietary software. However, after the development and delivery of the SASM - Sistema de Alistamento de Serviço Militar for the 2.185 Juntas de Serviço Militar, distributed by Brazil, there was a large number of doubts and problems about using the system by users. The process of correction of SASM was expensive, consumed many hours of the developers and delay in the updates of the system by users. Although the 3º CTA has one Processo de Desenvolvimento de Software (PDS), it does not discipline the construction activities and interfaces evaluation or it describes about the issues of ease of use facilities. Currently, this situation became more complex after the determination by the Federal Government, of the use of free software and the prioritization of the Web format in the systems development. This study proposes a method for the conception, development and interfaces evaluations for Web applications adhered to the PDS of the 3º CTA. The proposed method includes the user's participation in the process, it specifies the artifacts for being build, directs the activities of the development team by function and it shows the techniques and the necessary evaluations interface.

Key words: Usability Engineering, Web Applications, Development Process, Usability Principles and Guidelines.

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas e Siglas

3º CTA	- 3º Centro de Telemática de Área
ADV	- Abstract Data View
DER	- Diagrama de Entidade e Relacionamento
DECIDE	- Determine, Explore, Choose, Identify, Decide, Evaluate
DFD	- Diagrama de Fluxo de Dados
EB	- Exército Brasileiro
EVA	- Estudo de Viabilidade de Aplicativo
IEC	- International Electrotechnical Commission
INRIA	- Instituto Nacional de Pesquisa em Automação e Informática da França
HTML	- Hypertext Markup Language
ISO	- International Organization for Standardization
JSM	- Junta de Serviço Militar
MCT	- Ministério da Ciência e Tecnologia
NBR	- Norma Brasileira
OO	- Orientado a Objeto
OOHDM	- Object-Oriented Hypermedia Design Method
OM	- Organização Militar
PDS	- Processo de Desenvolvimento de Software
PMSLEB	- Plano de Migração para o Software Livre do Exército Brasileiro
SASM	- Sistema de Alistamento de Serviço Militar
SERMIL	- Sistema de Serviço Militar
TI	- Tecnologia da Informação
UML	- Unified Modeling Language
WebML	- Web Modeling Language

Sumário

1. Introdução

No passado os computadores eram caros, os sistemas eram específicos e exigiam um alto grau de conhecimento técnico dos usuários para operá-los. Apesar de a evolução tecnológica ter trazido para os computadores as interfaces gráficas, ricas em novos componentes que não existiam anteriormente, ainda há interfaces difíceis de usar, que levam os usuários a cometerem erros, alguns dos quais irreversíveis.

A baixa produtividade decorrente de projetos de interfaces mal elaboradas causa prejuízos financeiros, falta de competitividade, rotatividade de pessoal, menor retorno do investimento, problemas de auto-estima, estresses e irritações, pois o usuário se culpa e se sente diminuído quando não entende como usar o software (CYBIS, BETIOL e FAUST, 2007).

A existência de interfaces ruins, com muitos problemas, é resultante de fatores como a falta de conhecimento das necessidades dos usuários por parte dos projetistas, dos projetos de interface não centrados nos usuários, da preocupação apenas com a funcionalidade do sistema e o desconhecimento de fatores humanos, psicológicos, ergonômicos e formas de comunicação eficazes (ORTH, 2005).

Cybis, Betiol e Faust (2007) classificam os problemas de interfaces em dois grupos: Problemas de Usabilidade que se verificam por ocasião da interação com um sistema e os Problemas de Ergonomia que são identificados quando existe um desacordo entre um aspecto da interface e as características e maneiras pela qual os usuários realizam suas tarefas.

Existem, ainda, outros fatores que podem afetar a gravidade de um problema de interface: a frequência, que se refere a quantos usuários encontrarão o problema, quanto menos, menor a gravidade; o impacto, que se refere às dificuldades percebidas pelo usuários e podem variar de uma irritação imperceptível a perdas de horas de trabalho; e, a persistência, que é percebida quando o problema ocorre constantemente (NIELSEN e LORANGER, 2007).

No início da informatização, os usuários não tinham dificuldade de operar os

sistemas, pois eram seus próprios desenvolvedores. Com o tempo, os softwares passaram a ser utilizados por um pequeno público de não-desenvolvedores, para os quais, eram aplicados intensos treinamentos (CYBIS, BETIOL e FAUST, 2007).

Atualmente, principalmente com o advento da Internet, os sistemas visam um grande número de usuários sem nenhum treinamento, pois acessar a Internet tornou-se uma rotina, uma ferramenta pela qual os usuários realizam suas tarefas e atendem suas necessidades. Devido à grande disponibilidade de sistemas na Web, os usuários estão menos tolerantes e não utilizam sistemas complexos, difíceis de usar. “Portanto um projeto falho significa negócios perdidos. Nunca a usabilidade foi tão importante” (NIELSEN e LORANGER, 2007, p. XV).

A usabilidade está relacionada com a facilidade de uso de um produto, sendo definida pela norma brasileira NBR ISO/IEC 9126-1 (2003, p. 9) como a “capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas”.

No intuito de incluir a facilidade de uso de seus sistemas, as empresas de software vêm mudando a forma como elaboram seus projetos, pois perceberam que a inserção dos conceitos, metodologias e técnicas de concepção melhoram a qualidade da interface do usuário e aumentam as chances de sucesso do produto no mercado (ROCHA e BARANAUSKAS, 2003).

Iniciativas nesse sentido também são percebidas no setor público, onde o governo federal, com o objetivo de orientar os desenvolvedores de interfaces Web no âmbito público, disponibilizou a “Cartilha de Usabilidade” para Sítios e Portais do Governo Federal (Brasil, 2009) de forma a abranger as possíveis soluções dos problemas de interação das interfaces.

Ainda no âmbito público, encontra-se o 3º Centro de Telemática de Área (3º CTA), uma Organização Militar do Exército Brasileiro, que possui uma Seção de Sistemas, cuja preocupação com as questões de usabilidade, devido aos problemas detectados nas interfaces de seus sistemas e a recente priorização das plataformas livres com a adoção do ambiente Web no desenvolvimento de aplicações, têm ensejado várias mudanças e busca por soluções.

Este trabalho busca minimizar os problemas das interfaces das aplicações Web produzidas pelo 3º CTA, possibilitando a detecção e a solução dos problemas ainda durante o processo de desenvolvimento, a fim de produzir interfaces mais usáveis e que não impactem em alto custo ou acréscimo demasiado no tempo de desenvolvimento.

1.1 Objetivos do Trabalho

Diante deste contexto, o objetivo principal deste trabalho é propor um método para o desenvolvimento de aplicações Web que contemple a participação do usuário, as inspeções das interfaces durante o processo e que seja aderente ao Processo de Desenvolvimento de Software (PDS) do 3º CTA.

Tendo em vista o objetivo principal, a presente dissertação buscou os seguintes objetivos específicos:

- Identificar os conceitos, recomendações, métodos e técnicas que influenciam na concepção e avaliação de interfaces.
- Identificar as atividades de engenharia com foco em usabilidade a serem utilizadas.
- Identificar métodos e processos de desenvolvimento de aplicações Web.
- Identificar no PDS do 3º CTA os problemas de uso quanto às aplicações Web e as questões de construção de interfaces.

1.2 Justificativa e Contribuições do Estudo

O 3º Centro de Telemática de Área (3º CTA) é uma Organização Militar (OM) técnica do Exército Brasileiro (EB) com missão de desenvolvimento de software. Como em outras empresas e instituições voltadas para o desenvolvimento de sistemas, o 3º CTA, ao longo dos últimos anos, desenvolveu vários sistemas corporativos, como o SERMIL (Sistema de Serviço Militar), que é composto de um complexo sistema de âmbito nacional e vários sistemas periféricos, além de outros voltados para as demais OMs.

Em 2002 o 3º CTA desenvolveu o Sistema de Alistamento de Serviço Militar (SASM) e o distribuiu às Juntas de Serviço Militar (JSM), que são os órgãos de alistamento de serviço militar e estão distribuídos por mais de dois mil municípios do Brasil.

A disponibilização do manual de operação e o sistema de ajuda *on-line* do SASM não foram suficientes para evitar o grande número de acessos ao suporte por parte dos usuários. Muitas dúvidas e problemas relatados estavam relacionados ao entendimento das informações apresentadas nas interfaces que, da perspectiva dos desenvolvedores desse sistema, eram até mesmo considerados de simples solução, mas que da perspectiva dos usuários, não eram tão evidentes assim.

O grande volume de dúvidas e sugestões dos usuários fizeram com que o 3º CTA realizasse várias modificações e atualizações do produto SASM relacionadas ao apoio ao usuário durante a operação (apresentando mensagens de *feedback* e estado do sistema), ao uso de linguagem mais familiar ao usuário (nas opções de menus, nas caixas de diálogo, nos formulários) e à melhoria nos recursos de prevenção de erros e possibilidade de desfazer ações.

O processo de manutenção do SASM, além de caro e consumir muitas horas dos desenvolvedores, ocasionou outras complicações, tais como as dificuldades nas distribuições das novas versões e a demora das atualizações do sistema devido a problemas de comunicação com as JSM.

Para disciplinar o processo de desenvolvimento de software, em 2006 o 3º CTA desenvolveu o Processo de Desenvolvimento de Software (PDS), com o objetivo de estabelecer os critérios e os procedimentos metodológicos para o desenvolvimento de software na Seção de Sistemas.

Apesar do PDS definir vários pontos necessários no processo de desenvolvimento de software, tais como os agentes do processo, estudos de viabilidade, levantamento de requisitos e as fases do desenvolvimento, este deixa uma lacuna quanto aos processos de concepção e avaliação de interfaces.

Outra ocorrência a ser destacada no Exército Brasileiro, que impactou no

modelo de desenvolvimento de software do PDS, foi a promulgação do Plano de Migração para Software Livre (PMSLEB) por meio da Portaria nº 007-DCT (BRASIL, 2007), que regulou a estratégia para implantação do software livre em todos os escalões do EB e determinou a priorização da plataforma Web no desenvolvimento de sistemas.

O PDS foi concebido para disciplinar o processo de desenvolvimento de software convencional e não contempla as particularidades da construção de aplicações Web. Como conseqüência, o desenvolvimento das aplicações Web no 3º CTA está baseado na experiência da equipe de desenvolvimento.

Processos de desenvolvimento de aplicações Web e de software convencional diferem em muitos aspectos. Pode-se citar como mais importantes, os aspectos que englobam as pessoas envolvidas no desenvolvimento, as características específicas das aplicações Web e os usuários para os quais são desenvolvidos (MENDES, MOSLEY e COUNSELL, 2006).

A equipe de desenvolvimento de softwares convencionais é normalmente composta por profissionais com conhecimentos em programação, banco de dados e gerenciamento de projetos. As aplicações Web, por sua vez, englobam uma variedade maior de programadores como também de não programadores, como por exemplo, *designers*, especialistas no domínio da aplicação, especialistas em banco de dados e profissionais de Tecnologia da Informação.

Os softwares convencionais normalmente se destinam a uma plataforma computacional específica, comunicam-se com bancos de dados, podem estar conectados a outros sistemas e são desenvolvidos a partir de linguagens de programação, como por exemplo, o Delphi, o Visual Basic e o C++. As aplicações Web, por sua vez, utilizam tecnologias de comunicação, são multi-plataforma, utilizam recursos de navegação entre as páginas e englobam uma multiplicidade de tecnologias disponíveis para o desenvolvimento.

O grupo de usuários de softwares convencionais é normalmente conhecido pela equipe de desenvolvimento, tornando a tarefa de identificação das necessidades desses usuários mais fácil. No entanto, as aplicações Web

normalmente se destinam a um amplo grupo de usuários e muitas vezes desconhecidos.

A determinação pela utilização de software livre em formato Web, aliado ao volume de dúvidas dos usuários dos sistemas, relacionadas com questões de uso, motivaram o interesse pelo estudo e pesquisa dos aspectos relacionados à concepção e à avaliação de interfaces das aplicações Web.

Este estudo tem vínculo acadêmico com o contexto de Desenvolvimento de Tecnologia, haja vista que são estudados aspectos relacionados ao processo de desenvolvimento de aplicações Web, com proposta e foco em usabilidade. A área de concentração à qual este trabalho está relacionado é a de Inovação tecnológica e desenvolvimento sustentável, pois busca conhecer e entender os avanços tecnológicos, suas aplicações e seus impactos. E, dentro desta área de concentração, a linha de pesquisa à qual está relacionado é a de Gestão e desenvolvimento de tecnologias da informação aplicadas, com foco em desenvolvimento.

1.3 Metodologia

A construção do método proposto na presente dissertação tem como base uma pesquisa de natureza exploratória e bibliográfica.

As pesquisas exploratórias têm por objetivo buscar maior familiaridade com o problema, tornando-o mais explícito. Para seu desenvolvimento, na maioria dos casos, as pesquisas exploratórias envolvem levantamentos bibliográficos e análise de exemplos a fim de estimular a compreensão (GIL, 2007).

Para o processo de elaboração do conteúdo teórico do presente estudo, realizou-se levantamento bibliográfico na literatura, impressa e eletrônica, buscando-se fontes especializadas nas áreas de Sistemas de Informação e Tecnologia da Informação. As publicações pesquisadas, impressa e eletrônica, encontram-se distribuídas nos idiomas em português e inglês, no período de 1991 a 2009.

Considerada a metodologia adotada, foram analisados os elementos que

constituem o eixo da investigação:

- A conceituação e o estudo da usabilidade, sua estrutura, as diretrizes e as recomendações que se constituem no conhecimento necessário para a concepção de interfaces usáveis.
- A engenharia de usabilidade que compreende o estudo dos métodos estruturados para alcançar a usabilidade em projetos de interfaces, com a efetiva participação do usuário durante o desenvolvimento de um produto.

Com o objetivo de estimular a compreensão do problema e subsidiar a elaboração do método para aplicações Web, que é o objetivo geral desta dissertação, alguns métodos e processos de desenvolvimento de aplicações Web disponíveis na literatura foram analisados.

1.4 Organização da Dissertação

Além desta introdução, que contemplou a relevância, o interesse do trabalho, o problema, os objetivos, as justificativas e as contribuições do estudo, a presente dissertação apresenta mais cinco capítulos:

O capitulo dois, “Usabilidade”, define e descreve interfaces e os tipos de interações, apresenta a estrutura, os atributos da usabilidade e descreve um conjunto de princípios e diretrizes proposto pelos autores: Bastien e Scapin (1993) e por Nielsen e Loranger (2007). Seguida pela definição de usabilidade, a Engenharia de Usabilidade descreve as práticas para o desenvolvimento de aplicações Web com foco em usabilidade e no usuário, procurando sintetizar e descrever todas as atividades de forma seqüencial e progressiva.

O capítulo três, “Processos de Desenvolvimento de Aplicações Web”, define e diferencia os conceitos de *sites* e aplicações Web, com base nos estudos de Conallen (2000) e apresenta uma descrição geral de alguns métodos de desenvolvimento de aplicações Web que são encontrados na literatura. Esses métodos são considerados como base para a construção do método proposto pelo presente trabalho.

O capítulo quatro, “O Processo de Desenvolvimento de Software do 3º CTA”, descreve o PDS do 3º CTA, suas atividades, o pessoal envolvido e apresenta algumas considerações do Processo. Ao final do capítulo, conclui-se sobre qual metodologia de desenvolvimento de aplicações Web, estudadas no capítulo anterior, é mais adequada para servir de base para a construção do método proposto pelo estudo.

No capítulo cinco, “Desenvolvimento do Método de para Aplicações Web”, é apresentada a descrição detalhada do método proposto, principal objetivo desta dissertação. O método apresenta as fases, os artefatos, as atividades previstas na Engenharia de Usabilidade, os princípios e as diretrizes para concepção, as avaliações de interfaces e as considerações quanto ao método proposto.

O capítulo seis, “Conclusão”, apresenta as considerações gerais sobre esta dissertação e indica as perspectivas de trabalhos futuros proporcionadas por esta pesquisa.

2. Usabilidade

A usabilidade pode ser considerada como um sinônimo de facilidade de uso de um produto qualquer, como por exemplo: dirigir um automóvel, operar uma máquina, tirar uma foto ou ainda, mais especificamente para este trabalho, interagir com um sistema computacional. Substituta da vaga expressão “amigável”, muito utilizada pelos fabricantes de software para qualificar seus sistemas computacionais como “fáceis de usar”, a usabilidade se mostra mais abrangente, mensurável e com abordagens voltadas não somente para o produto, mas também para o ambiente, usuários e interação.

A abrangência da usabilidade foi especificada pela norma NBR 9241-11 (2002) e pode ser verificada pela identificação de seus componentes: os usuários, as tarefas, os equipamentos, o ambiente e os valores reais ou desejados de eficácia, eficiência e satisfação dos usuários por ocasião da interação. Além destes, citados pela norma, Nielsen (1994) acrescenta outros seis atributos que denotam a amplitude da usabilidade: a facilidade de aprender, a eficiência de uso, a facilidade de relembrar, poucos erros, a satisfação objetiva e a utilidade.

De forma geral, os componentes e os atributos de usabilidade constituem-se de pontos que devem ser levados em conta por ocasião da concepção dos sistemas computacionais. Além desses, ainda são necessárias especificações de usabilidade mais pontuais, que direcionem a concepção e possibilitem o projeto e o desenvolvimento de interfaces mais usáveis.

Os princípios e as diretrizes de usabilidade constituem-se então, nas especificações e aconselhamentos mais pontuais que têm por objetivo orientar o desenvolvedor Web na elaboração de interfaces que contemplem as questões usabilidade.

Para disciplinar o processo de desenvolvimento, a Engenharia de Usabilidade, também conceituada e descrita neste capítulo, descreve os métodos estruturados, a participação do usuário, o planejamento, as avaliações e as atividades que possibilitarão alcançar a usabilidade em projetos de interface de usuário durante o desenvolvimento de um produto.

A aplicabilidade da usabilidade dos sistemas computacionais está voltada para as interfaces e interações, pois sua preocupação é a de projetar interfaces fáceis de usar, que forneçam seqüências simples, consistentes e que mostrem claramente as alternativas disponíveis em cada passo da interação (OLIVEIRA NETTO, 2006).

Antes de se discutir efetivamente as questões relativas à usabilidade, torna-se necessário discorrer sobre os conceitos de interface e a interação.

2.1 Interface e Interação

Interface é o nome dado a toda a porção de um sistema com a qual um usuário mantém contato ao utilizá-lo e pressupõe a existência de um componente físico que o usuário manipula, um componente perceptivo que engloba as percepções do usuário durante a interação e o componente conceitual que é resultante dos processos de interpretação e raciocínio do usuário (PRATES e BARBOSA, 2003).

Como componentes físicos, uma interface compreende o hardware e o software. Os componentes de hardware compreendem os dispositivos com os quais o usuário interage para realizar suas atividades motoras e perceptivas (a tela, o teclado, o *mouse* e outros dispositivos físicos). O software da interface implementa os processos computacionais necessários para o controle dos dispositivos de hardware, geração de símbolos e mensagens, construção de dispositivos visuais e interpretação dos comandos de usuários (DE SOUZA *et al*. 1999).

Segundo Oliveira Netto (2006), houve uma evolução do conceito de interfaces ao se adicionar os aspectos relativos ao processamento perceptual, motor, viso-motor e cognitivo do usuário.

A interação (Figura 1) é um processo que compreende as ações do usuário sobre a interface do sistema e suas interpretações a partir das respostas reveladas pela interface (DE SOUZA *et al*., 1999).

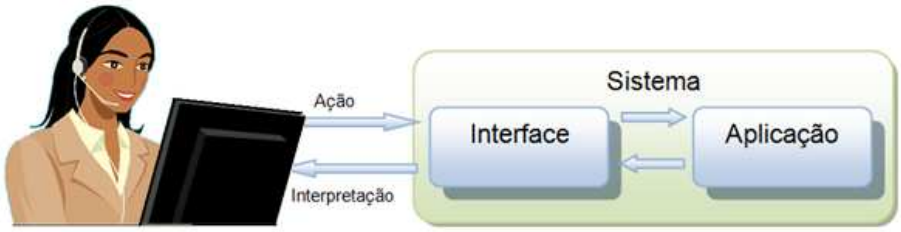


Figura 1– Processo de Interação Humano-Computador
Fonte: Adaptado de DE SOUZA *et al*. (1999, p. 4)

A interação pode ser classificada em quatro tipos de atividades que os usuários executam quando utilizam um sistema: a instrução; a conversação; a manipulação e navegação e a exploração e pesquisa. Cabe observar que os tipos de interação não excluem uns aos outros, pois um usuário pode realizar mais de um tipo de interação ao mesmo tempo (PREECE, ROGERS e SHARP, 2005).

A instrução é um tipo de atividade na qual o usuário instrui o sistema quanto à realização das tarefas, o que pode ser feito por meio de alguns estilos diferentes de interação, como por exemplo: digitar um comando, selecionar opções em um menu, dar comandos de voz, pressionar botões ou pressionar teclas.

A segunda atividade compreende a conversação do usuário com o sistema, que o usuário digita ou fala as perguntas e o sistema responde. Esta atividade difere da anterior ao pressupor que ocorre uma comunicação de duas vias e não apenas a execução de um comando do usuário.

A manipulação e navegação visam disponibilizar aos usuários um ambiente virtual que permita a navegação e a manipulação de objetos, compartilhando o ambiente virtual com as propriedades do mundo físico. Objetos virtuais podem, por exemplo, ser selecionados, movidos, abertos, fechados ou excluídos.

A exploração e a pesquisa consistem em disponibilizar as informações de forma estruturadas a fim de possibilitar que os usuários explorem, pesquisem e descubram o que procuram, apenas navegando pelo sistema, sem ter que elaborar uma questão específica.

As conceituações de interface e os tipos de atividades que ocorrem em uma interação constituem-se em conhecimentos necessários para a concepção de interfaces do usuário, porém, é preciso conceituar e identificar também a

usabilidade, seus componentes, princípios e diretrizes, pois estes têm por objetivo orientar o projetista Web na elaboração dessas interfaces com foco em usabilidade.

2.2 Definição de Usabilidade

Na literatura é possível encontrar várias definições para o termo “Usabilidade”. Segundo Dias (2007), o termo passou a ser usado no início dos anos 80, principalmente nas áreas de Psicologia e Ergonomia, como substituto da expressão *user-friendly* (traduzido para o português como “amigável”) por ser considerada uma expressão muito vaga no sentido de que as máquinas não precisam ser amigáveis, elas apenas não devem interferir nas tarefas dos usuários, que, por sua vez, têm necessidades diversas e características físicas e psicológicas diferentes.

A partir dos anos 80, vários autores tentaram definir o termo usabilidade utilizando abordagens diferentes (Quadro 1), mostrando que o foco do estudo está além do desenho da interface, pois engloba todos os aspectos relacionados entre usuários e computadores.

Quadro 1 - Abordagens relacionadas às definições de usabilidade

Abordagem	Descrição
Orientadas ao produto	Associadas às características ergonômicas do produto
Orientadas ao usuário	Relacionadas ao esforço mental ou atitude do usuário frente ao produto
Desempenho do usuário	Interação do usuário, com ênfase na facilidade de uso e no grau de aceitação do produto
Orientadas ao contexto de uso	Tarefas específicas realizadas por usuários específicos, em determinado ambiente de trabalho

Fonte: Adaptado de Dias (2007, p. 25)

A primeira norma a definir o termo usabilidade foi a ISO/IEC 9126 (1991), com uma abordagem orientada ao produto e ao usuário, por considerar a usabilidade como “um conjunto de atributos de software relacionado ao esforço necessário para seu uso e para o julgamento individual de tal uso por um determinado conjunto de usuários”. Segundo Dias (2007, p. 26), a partir dessa definição, houve uma expansão do uso do termo usabilidade:

> [...] o termo usabilidade ultrapassou os limites do ambiente acadêmico da Psicologia Aplicada e da Ergonomia, passando a fazer parte do vocabulário técnico de outras áreas do conhecimento, tais como a Tecnologia da Informação e Interação Homem-Computador, tendo sido traduzido literalmente para diversos idiomas.

A norma brasileira NBR 9241-11 (2002, p. 3), que trata de qualidade de pacotes de softwares bem como das características ergonômicas do produto, estrutura a usabilidade em componentes e a define como a “Medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso”.

A fim de especificar ou medir a usabilidade de um produto, a NBR 9241-11 (2002) atenta para a necessidade de se identificar os objetivos, os componentes do contexto de uso e os atributos mensuráveis e verificáveis, detalhando e relacionando-os em uma estrutura de usabilidade.

2.3 Componentes da Usabilidade

A identificação dos componentes de usabilidade constitui-se em: descrever os objetivos pretendidos, o contexto de uso existente ou pretendido de forma a incluir os usuários, tarefas, equipamentos e ambientes; e os valores reais ou desejados de eficácia, eficiência e satisfação para os contextos pretendidos (Figura 2).

O contexto de uso engloba os componentes de usabilidade relacionados com a interação do usuário com o produto e deve descrever as características relevantes de cada um deles: dos usuários, das tarefas executadas pelos usuários para alcançarem um objetivo, dos equipamentos (hardware, software e materiais relacionados) e, finalmente, do ambiente físico e social.

As características relevantes dos usuários, tais como o conhecimento, habilidade, experiência, educação, treinamento, atributos físicos e capacidades sensoriais precisam ser descritas, pois modificam o contexto de uso e influenciam no nível de usabilidade. Para Cybis, Betiol e Faust (2007), uma mesma interface pode proporcionar interações satisfatórias para usuários considerados experientes e ser um desastre para usuários novatos.

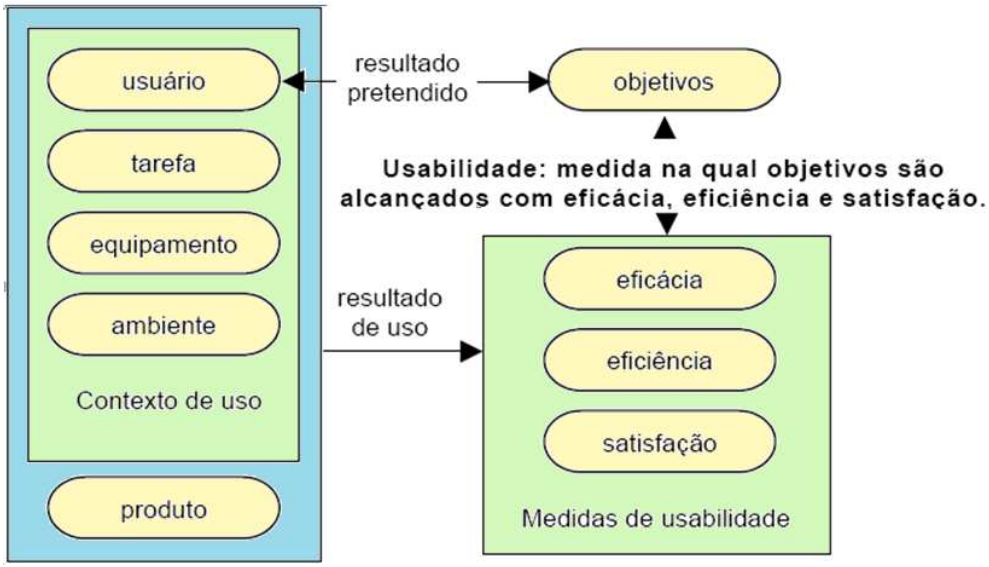


Figura 2 - Estrutura e componentes de usabilidade
Fonte: Adaptado da NBR 9241-11 (2002, p. 4)

As tarefas constituem-se de atividades que devem ser executadas para se atingir um determinado objetivo. Sendo assim, a especificação das tarefas deve ir além de descrever atividades, ela deve ter foco nos objetivos a serem alcançados. Cabe salientar que por ocasião das avaliações de usabilidade, um conjunto de tarefas-chave é selecionado para representar aspectos significantes da tarefa como um todo.

A descrição das características relevantes dos equipamentos tais como hardwares, softwares, materiais associados com computadores podem ter foco na avaliação de usabilidade.

Dentro do contexto de uso, a usabilidade ainda pode ser dependente das características do ambiente de uso, seja ele técnico (por exemplo, a rede de trabalho local), físico (por exemplo, o local de trabalho, mobiliário), atmosférico (por exemplo, a temperatura e a umidade), cultural ou social (por exemplo, as práticas de trabalho, a estrutura organizacional e as atitudes).

As medidas de usabilidade são obtidas pelo resultado de uso da combinação dos elementos de contexto de uso (usuários, tarefa, equipamentos e ambiente de uso) em função dos atributos de eficácia, eficiência e satisfação com que os usuários atingem os objetivos estabelecidos ou ainda, os resultados pretendidos.

A eficácia é percebida quando um sistema interativo possibilita que o usuário

atinja seus objetivos. Este atributo é considerado como a principal motivação de uso de um determinado sistema, ou seja, mesmo que um sistema seja fácil de usar, fácil de aprender e agradável de usar (eficiente e satisfatório), caso não atenda aos objetivos específicos do usuário (não eficaz), ele não será usado, mesmo que seja oferecido de forma gratuita (DIAS, 2007).

A eficiência é medida pela relação entre o nível de eficácia e o dispêndio de recursos utilizados. Os recursos podem incluir os esforços físicos e mentais, custos materiais ou financeiros. Para Dias (2007), a eficiência pode ser definida como a precisão e completeza com que os usuários atingem seus objetivos em relação aos recursos utilizados.

A satisfação do usuário refere-se às suas percepções quanto ao uso de um sistema. Percepções como desconforto, gosto pelo produto, satisfação ou aceitação da carga de trabalho podem ser especificados a partir de avaliações subjetivas ou objetivas aplicadas aos usuários. Tais medidas podem ser obtidas a partir do uso de questionários ou de observações de comportamento dos usuários.

Os componentes de eficiência, eficácia e satisfação do usuário aqui apresentados e constantes das medidas de usabilidade, propostos pela NBR 9241-11 (2002), não se constituem nos únicos atributos de usabilidade, como serão apresentados na seção seguinte.

2.4 Atributos de Usabilidade

Para Nielsen (1994, 2003), a usabilidade não é uma propriedade unidimensional de uma interface, pois ela é composta de múltiplos componentes, sendo tradicionalmente associada a outros atributos como: facilidade de aprender, eficiência de uso, facilidade de relembrar, poucos erros, satisfação objetiva e a utilidade.

O atributo facilidade de aprender é relativo ao quanto o sistema é fácil de aprender de modo que o usuário possa rapidamente usá-lo e realizar suas tarefas. O usuário deve ser capaz de aprender rapidamente a utilizar o sistema logo em sua primeira experiência, pois se não conseguir, ele tentará novas alternativas para

atingir seus objetivos.

Nielsen (1994) considera este atributo de usabilidade fundamental, pois muitos sistemas devem ser fáceis de usar desde a primeira experiência de uso, apesar de existirem sistemas mais especializados e complexos que exigem treinamento para aprender a usar; mesmo assim, na maioria dos casos, os sistemas devem ser fáceis de usar.

Preece, Rogers e Sharp (2005) afirmam que os usuários não gostam de passar muito tempo aprendendo como usar um sistema, eles preferem apenas utilizá-lo logo e tornarem-se competentes para usá-lo sem muito esforço. A questão-chave do conceito facilidade de aprender é determinar quanto tempo os usuários estão dispostos a gastar conhecendo um sistema. Conseqüentemente, por vezes, não há sentido em desenvolver uma série de funcionalidades se a maioria dos usuários não pode ou não está disposta a gastar muito tempo aprendendo.

Ao analisar o atributo da facilidade de aprender é preciso considerar que a aprendizagem ocorre durante o uso da interface, pois o usuário não estuda todo o sistema antes de usar. Sendo assim, a análise desse atributo deve ser feita em função do tempo que o usuário demora em atingir certo grau de proficiência no uso do sistema (ROCHA e BARANAUSKAS, 2003).

O atributo eficiência de uso prevê que o sistema deve permitir que, assim que se tenha aprendido a usá-lo, o usuário seja capaz de executar suas tarefas com altos níveis de produtividade. Portanto, este princípio está relacionado com usuários experientes, após algum tempo de uso do sistema.

A eficiência de uso está relacionada com a propriedade de tempo de resposta, também definido como taxa de comunicação entre o usuário e o sistema. A propriedade tempo de resposta refere-se ao tempo necessário para o sistema informar sua mudança de estado ao usuário. Normalmente, é desejável que os sistemas tenham tempos de resposta instantâneos ou de curta duração, caso contrário, o sistema deve informar ao usuário que está processando sua solicitação (DIAS, 2007).

Para avaliar a eficiência de uso de um sistema é necessário, inicialmente, definir um grupo de usuários experientes e avaliar o tempo gasto por esse grupo na realização de algumas tarefas típicas do sistema.

O atributo facilidade de relembrar refere-se a quando o usuário retorna a um sistema, após um período sem utilizá-lo, o usuário deve ser facilmente relembrado a fim de que não tenha de aprendê-lo novamente.

Para Nielsen (1994), os usuários casuais compreendem o terceiro maior grupo de usuários, ao lado dos novatos e dos experientes. São usuários que usam os sistemas de forma intermitente, porém, diferentemente dos usuários novatos, eles já usaram o sistema antes e não precisam aprender do zero, precisando apenas relembrar como usar.

Este atributo também se aplica para aqueles sistemas que são utilizados em períodos específicos após longos intervalos, como os sistemas de imposto de renda, sistemas para confecção de relatórios de atividades trienais etc. (ROCHA e BARANAUSKAS, 2003).

Para Dias (2007), interfaces fáceis de aprender geralmente são fáceis de relembrar, apesar de, em princípio, a usabilidade de quem já usou o sistema há algum tempo é diferente de quem está interagindo pela primeira vez com o sistema.

Há várias formas de se projetar uma interface que auxilie esses usuários a lembrar como realizar suas tarefas: disponibilizar ícones representativos em categorias relevantes de opções, nomes de comandos e opções de menus (PREECE, ROGERS e SHARP, 2005).

A facilidade de relembrar é um atributo raramente testado. Uma forma de fazê-lo é testar o sistema com um grupo de usuários que deixaram de usar o sistema por um determinado período, medindo o tempo que levam para executar tarefas específicas.

O atributo erros constitui-se em ações que não levam o usuário ao resultado esperado. O sistema deve evitar que os usuários cometam erros durante o uso. No caso de sua ocorrência, os sistemas devem possibilitar que os usuários corrijam e

não percam seu trabalho.

Erros, quando ocorrem, têm diferentes impactos no trabalho do usuário, dependendo de seu grau de complexidade. Alguns erros são imediatamente corrigidos pelo usuário sem danos à execução da tarefa. Outros afetam a eficiência de uso, que normalmente é medida em termos de tempo para realização de alguma tarefa. Existem ainda alguns erros considerados “catastróficos”, que destroem o trabalho do usuário, impossibilitando sua correção.

A avaliação da taxa de erros de um sistema pode ser contabilizada a partir da quantidade de erros ocorrida por ocasião da realização de uma determinada tarefa, considerando o grau de complexidade (NIELSEN, 1994).

O atributo de satisfação subjetiva é direcionado para que o sistema seja agradável, de forma que o usuário esteja satisfeito em utilizá-lo. Esta satisfação pode ser especialmente desejável para sistemas que não estão relacionados com o ambiente de trabalho, escolhidos pelo usuário como forma de lazer, no qual o entretenimento e o envolvimento são geralmente mais desejáveis do que a velocidade com que as tarefas são completadas.

A avaliação da satisfação subjetiva pode ser obtida a partir de questionários em que os usuários associam sua experiência interativa a adjetivos predeterminados, respondendo afirmativa ou negativamente frases prontas, identificando pontos positivos e negativos da interface, podendo ainda expor suas opiniões. Sob a perspectiva de um usuário, evidentemente as respostas são subjetivas, mas quando o questionário é aplicado a um número razoável de usuários, as respostas após consolidação é uma medida objetiva de satisfação de uso.

O atributo utilidade é considerado como chave, pois este se refere à capacidade do sistema de fazer o que o usuário deseja do sistema. Nielsen (2003) coloca a usabilidade e a utilidade em igualdade de importância, quando afirma que um sistema deve ser fácil de usar e permitir que o usuário faça o que precisa. Isto significa que não é desejável um sistema que apesar de fazer o que o usuário precisa, seja difícil de operar e também não é desejável um sistema ser fácil de usar, mas que não atenda as necessidades do usuário.

Os atributos de usabilidade têm por objetivo a concepção de produtos mais usáveis, porém, estes se constituem em colocações genéricas e não são específicas quanto à sua aplicação. Nesse sentido, os princípios e as diretrizes de usabilidade procuram atender aos atributos de usabilidade e são mais específicos quanto à sua aplicação na concepção de interfaces.

2.5 Princípios e Diretrizes de Usabilidade

Os princípios de usabilidade constituem-se de aconselhamentos sobre as características de usabilidade de uma determinada interface de usuário e dependendo do projeto, diferentes princípios podem ser utilizados: princípios gerais, para todas as interfaces de usuários e princípios específicos para um tipo específico de sistema, como por exemplo, os destinados às crianças e às pessoas com deficiências físicas (NIELSEN, 1994).

As diretrizes, que são versões mais específicas do que os princípios de usabilidade, proporcionam uma orientação mais detalhada e normalmente são acompanhadas de notas explicativas, exemplos e comentários (PREECE, ROGERS e SHARP, 2005).

Preece, Rogers e Sharp (2005) explicam que princípios de *design* são mais abstratos, exigem uma interpretação antes de sua aplicação. Por exemplo, o princípio para *sites* Web “Limite o número de estilos fontes e cores no seu *site* e aplique-os consistentemente” (NIELSEN e LORANGER, 2007, p. 235), apesar de relevante, é bastante genérico, devendo ser interpretado antes de ser aplicado ao contexto do sistema.

As diretrizes, também chamadas de regras de *design*, apresentam-se sempre de forma mais detalhada, mais específica e não exigem interpretação para sua aplicação. Como por exemplo, a seguinte diretriz: “Não ofereça uma opção de busca em toda rede a partir do seu próprio *Website*”.

Para o presente estudo, serão abordados os princípios dos pesquisadores Bastien e Scapin (1993), membros do INRIA (Instituto Nacional de Pesquisa em Automação e Informática da França) e como o presente trabalho é direcionado para

aplicações Web, serão apresentadas também, a seguir, algumas das principais diretrizes direcionadas para *sites* e aplicações Web propostas por Nielsen e Loranger (2007).

Os princípios propostos por Bastien e Scapin (1993) constituem-se de oito critérios principais subdivididos em sub-critérios e critérios elementares. Os critérios elementares são apresentados no estudo com sua definição, as razões e exemplos de diretrizes.

Os princípios estão estruturados de forma a minimizar a ambigüidade na identificação e classificação das qualidades e dos problemas da interface, aumentando a sistematização de resultados das avaliações, quando realizadas por diferentes especialistas (CYBIS, BETIOL e FAUST, 2007).

2.5.1 Condução do Usuário pelo Sistema

O critério de conduzir o usuário significa que o sistema deve: aconselhar, orientar, informar, instruir e guiar o usuário na interação com o sistema (podem ser utilizadas mensagens, alarmes ou rótulos). A condução deve possibilitar a localização do usuário, o conhecimento das ações permitidas, suas conseqüências e proporcionar aprendizado rápido, fácil utilização e poucos erros.

A análise da condução é feita a partir de outros quatro subcritérios: o convite, o agrupamento e distinção pela localização, agrupamento e distinção pelo formato, *feedback* imediato e a legibilidade.

O subcritério convite é um critério elementar que compreende os meios de levar o usuário a realizar determinadas ações a partir de informações que identificam os estados ou o contexto no qual se encontra o sistema, as ações alternativas, as ferramentas de ajuda e o modo de acesso.

Como exemplo, as interfaces convidativas apresentam aos usuários as seguintes características, tais como (Figura 3):

- O sistema deve permitir ao usuário identificar o estado ou contexto no qual se encontra;

- O sistema deve possibilitar ao usuário conhecer as alternativas de ações;
- O sistema deve apresentar títulos claros para as telas, janelas e caixas de diálogo;
- Para a entrada de dados, o sistema deve indicar o formato adequado e os valores aceitáveis (Por exemplo, as entradas para datas: Data (dd/mm/aaaa: __/__/____);
- O sistema deve indicar o tamanho do campo quando ele é limitado;
- O sistema deve prover ajuda *on-line* claramente indicada.

Figura 3 - Uma interface convidativa (com destaques do autor)
Fonte: http://www.submarinoviagens.com.br, acesso em: 16 de maio de 2009

O subcritério de agrupamento e distinção pela localização é uma qualidade que caracteriza um software organizado espacialmente e refere-se ao posicionamento relativo dos itens dentro de grupos de informações. Para Cybs, Betiol e Faust (2007), uma interface espacialmente organizada deve apresentar as seguintes características:

- Os grupos e opções de menus devem ser definidos logicamente em função dos objetos e ações a que se aplicam (organização das opções de menu apresentadas na Figura 4);

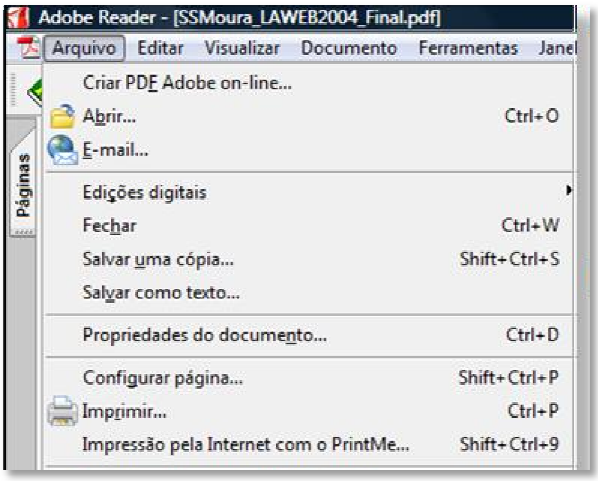


Figura 4 - Menu com opções organizadas em grupos
Fonte: Adobe Reader

- Os campos dos formulários devem ter uma seqüência lógica (Figura 5);
- As listas de dados ou informações devem ser coesas (conter informações de um mesmo tipo) e ordenadas logicamente;
- Deve haver separação ou aproximação de itens e grupos nas telas, conforme as relações lógicas estabelecidas entre elas.

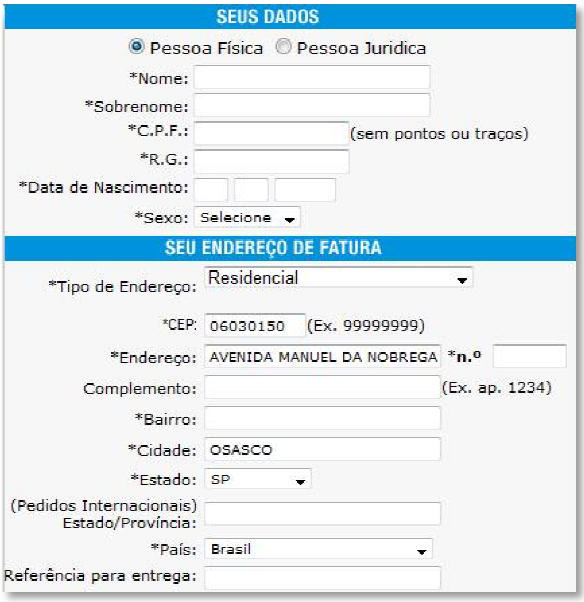


Figura 5 - Formulário com a seqüência lógica de campos
Fonte: Adaptado de http://www.submarinoviagens.com.br/Viagens/Cadastro.aspx, acesso em: 16 de maio de 2009

O subcritério de agrupamento e distinção pelo formato refere-se à organização dos itens a partir da forma gráfica dos componentes, tais como cores, tamanhos, estilos dos caracteres e outras formatações da interface, permitem que o

usuário perceba as similaridades e diferenças das informações.

Uma interface graficamente organizada deve estabelecer uma distinção visual para elementos de funções diferentes, tais como comandos, ferramentas, dados e informações (Figura 6); nos formulários, deve haver ainda, distinção entre rótulos e dados.

Figura 6 - Interface graficamente organizada com distinção visual de cores e ícones
Fonte: Adaptado de Cybis, Betiol e Faust (2007, p. 29)

O subcritério de *feedback* imediato consiste em garantir que o sistema apresente respostas às ações do usuário, desde um simples pressionamento de tecla a uma ação mais complexa. Em todos os casos o sistema deve oferecer respostas que devem ser rápidas, apropriadas e consistentes para os diversos tipos de transações.

A rapidez do *feedback* é um fator de qualidade necessário para o entendimento do diálogo e o estabelecimento da confiança e satisfação do usuário. As conseqüências da ausência ou demora de *feedback* pode confundir o usuário, indicar uma falsa falha do sistema e até induzir o usuário a tomar atitudes prejudiciais para o sistema.

Exemplos de situações aplicáveis:

- Todas as entradas e ações do usuário devem ser notificadas pelo sistema. Mesmo as entradas confidenciais devem ser relatadas de modo a não revelar seu conteúdo, como por exemplo, o uso de asteriscos (conforme a interface de identificação e autenticação do

sistema na Figura 7);

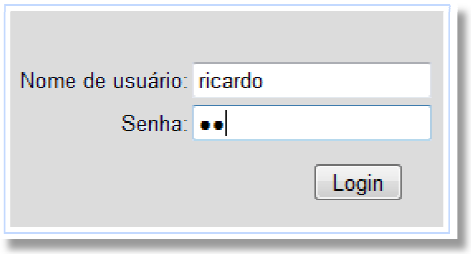


Figura 7 - Interface com recursos de confidencialidade
Fonte: Tela de login o autor

- Após o cancelamento de uma transação, o sistema deve apresentar uma mensagem informando ao usuário que o sistema retornou ao estado anterior;

O sistema deve informar ao usuário que um processamento demorado está em execução, bem como a sua conclusão (conforme a tela de *downloads* na Figura 8).

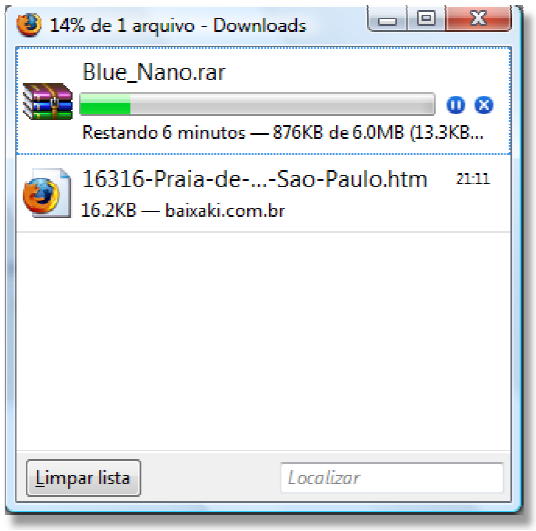


Figura 8 - Interface com identificação de processamento em execução
Fonte: Tela de *download* do navegador Mozzila Firefox

A legibilidade é um subcritério que se preocupa com as características da apresentação da informação na interface, que podem facilitar ou dificultar a leitura desta informação (brilho do caractere, contraste entre a letra e o fundo, tamanho da fonte, espaço entre letras, espaço entre linhas, espaço entre parágrafos e comprimento da linha).

Pode-se melhorar o desempenho da interação do usuário quando a apresentação da informação na tela leva em consideração as características

cognitivas e perceptivas dos usuários, tais como: atenção, percepção, memória, raciocínio, imaginação, pensamento e linguagem. Apesar do subcritério de legibilidade ser uma qualidade genérica, este deve ser particularmente observado quando a interface for destinada para pessoas idosas e com problemas de visão (CYBIS, BETIOL e FAUST, 2007).

Em uma interface legível:

- Os rótulos devem iniciar com letra maiúscula;
- Textos longos devem aparecer em letras maiúsculas e minúsculas naturalmente (maiúsculas no início de frases e em nomes);
- Para o caso de textos que devem ser lidos por idosos ou pessoas com problemas de visão, as letras devem ser claras sobre um fundo escuro, pois para essas pessoas, a disposição de letras escuras sobre fundo brilhante ofusca as letras. No exemplo da Figura 9, as telas da plataforma Lattes disponibilizam apresentações para pessoas com visão normal (fundo claro e letras escuras) e para pessoas com baixa visão (fundo escuro com letras claras).

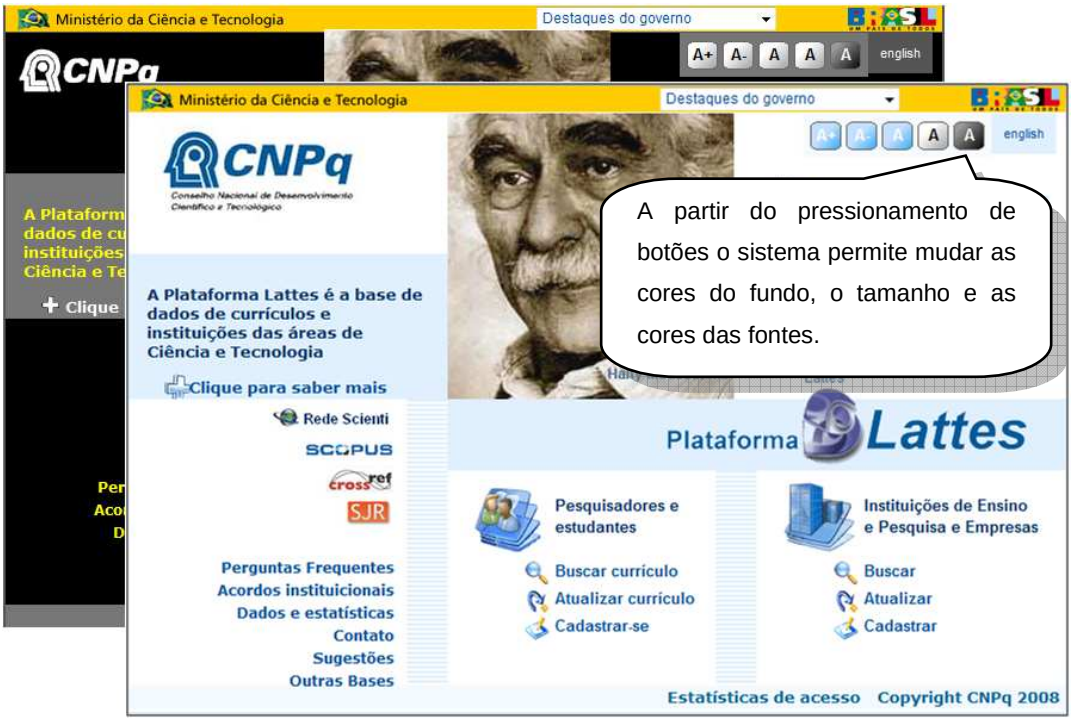


Figura 9- Interfaces com recursos de acessibilidade (com destaque do autor)
Fonte: http://lattes.cnpq.br/, acesso em: 16 de maio de 2009

Para Nielsen e Loranger (2007), independentemente da qualidade visual do *site*, se as pessoas não puderem ler o texto facilmente, ele estará destinado ao

fracasso. A preocupação com a tipografia e os esquemas corretos são componentes essenciais de um bom *design* visual. Para tanto, os autores sugerem quatro diretrizes fundamentais para apresentação de um texto:

- O sistema deve utilizar um tamanho de fonte comum com 10 ou mais pontos;
- Fundos visualmente poluídos devem ser evitados;
- O sistema deve utilizar texto preto sobre fundos brancos;
- Deve ser evitado o uso de textos gráficos, texto com todas as letras maiúsculas e texto em movimento.

2.5.2 Carga de Trabalho

A carga de trabalho é um critério necessário para contextos de trabalho intensos e repetitivos. Este critério tem por objetivo garantir a redução da carga cognitiva e do esforço físico do usuário, economizando leituras, memorizações desnecessárias, deslocamentos inúteis e repetições de entradas.

Em geral, observa-se que a alta carga de trabalho aumenta a probabilidade erros, pois quanto menor for a distração do usuário em virtude de informações desnecessárias, mais ele será capaz de cumprir suas tarefas de forma eficiente.

O critério de carga de trabalho é subdividido em dois subcritérios elementares: a brevidade (que inclui a concisão e ações mínimas), e a densidade informacional.

O subcritério de brevidade constitui-se em respeitar a capacidade de trabalho perceptivo, cognitivo e motor do usuário. O critério compreende preocupações como: limitação da carga de leitura; diminuição de entradas e ações necessárias para se alcançar uma meta no sistema; e a disponibilização de itens mais sucintos a fim de diminuição do tempo de leitura.

O critério de brevidade subdivide-se em dois outros subcritérios elementares: concisão e ações mínimas.

A concisão se preocupa com a minimização da carga perceptiva, conceitual

e motora do usuário no que se refere à realização de entradas e saída individuais.

Em uma interface concisa:

- O sistema deve apresentar denominações curtas para títulos de telas, janelas, caixas de diálogo, rótulos de campos, botões e comandos;
- Para campos com unidades de medida associadas, os usuários não devem ser obrigados a digitar tais unidades, devendo o sistema preenchê-los automaticamente (por exemplo, campos com valores de: larguras, distâncias, preços e temperaturas);
- Para dados numéricos, a entrada de zeros à esquerda não deve ser necessária.

Ações mínimas é um critério elementar que busca minimizar e simplificar as ações necessárias para que o usuário realize suas tarefas ou atinja seus objetivos. Quanto mais numerosas e complexas forem as ações necessárias para alcançar um objetivo, maior será a carga de trabalho e, conseqüentemente, maior será a probabilidade de erros.

Exemplos de aplicações do critério de ações mínimas:

- O sistema deve minimizar o número de passos requeridos para seleção de um menu;
- O sistema não deve solicitar aos usuários dados que podem ser deduzidos pelo próprio sistema (por exemplo: datas que podem ser calculadas pelo sistema e informações que podem ser buscadas internamente no sistema);
- O sistema deve, sempre que possível, fornecer valores padrões para campos de dados, listas, botões de opção e onde for capaz de acelerar as entradas individuais;
- Para os campos de dados numéricos, o sistema deve fornecer preenchimento automático de vírgulas, pontos decimais e zeros à direita;
- Quando várias páginas estiverem envolvidas, o sistema deve possibilitar que o usuário vá diretamente para uma página sem ter que passar pelas intermediárias.

O subcritério de densidade informacional se preocupa com a carga de trabalho do usuário do ponto de vista perceptivo e cognitivo, quanto ao conteúdo total de informações apresentadas, não se preocupando com cada elemento ou item de forma individual.

O desempenho dos usuários, em muitas tarefas, é prejudicado quando a densidade de informações é muito alta ou mesmo muito baixa, o que conseqüentemente, também levam os usuários a erros.

Este subcritério atende principalmente os usuários iniciantes, que têm dificuldade em filtrar as informações que realmente necessitam em uma interface carregada com muitos textos, imagens e outros conteúdos (CYBIS, BETIOL e FAUST, 2007).

Exemplos de aplicação do critério de densidade informacional:

- O sistema deve apresentar somente os itens que estão relacionados com a tarefa, removendo o restante (Figura 10);

Figura 10 - Interface com densidade informacional adequada (com destaques do autor)
Fonte: http://www.google.com.br/, acesso em: 16 de maio de 2009

- Os dados apresentados na interface não devem requerer traduções;
- O sistema não deve forçar os usuários a transportarem mentalmente dados de uma tela à outra;
- Nas tarefas ou formulações de questões, o sistema deve minimizar o número de campos a serem preenchidos, como por exemplo, a especificação de buscas avançadas “Pesquisa avançada” (Figura 10);

Sempre que possível, o sistema deve realizar o cálculo automático de dados

informados pelo usuário, evitando que este tenha de fazê-lo.

2.5.3 Controle Explícito

O critério de controle explícito preocupa-se em delegar ao usuário o controle das ações efetuadas pelo sistema, pois permite que o usuário possa definir explicitamente suas entradas no sistema, que mantenha controle das transações e, como conseqüência, os erros e as ambigüidades tendem a ser limitados e assim, a aceitabilidade do sistema tende a melhorar.

Este critério é aplicável em particular às tarefas longas e seqüenciais, nas quais os processamentos são normalmente demorados. Cybis, Betiol e Faust (2007) citam que em tais atividades, a falta de controle por parte dos usuários pode implicar em perda de tempo e dados.

O critério de controle explícito é dividido em dois critérios elementares: as Ações explícitas dos usuários e o Controle do usuário.

As ações explícitas do usuário referem-se ao relacionamento entre o processamento da transação e as ações do usuário. A relação pressupõe que o sistema deve processar apenas as ações requeridas pelo usuário e quando ele solicitar.

A possibilidade de controle dos passos de execução de um processamento pelo usuário favorece a aprendizagem, o entendimento do funcionamento do sistema e diminui a possibilidade de erros.

Exemplos de aplicação do critério de ações explícitas do usuário:

- Para entrada de dados ou parâmetros, o sistema deve sempre solicitar uma ação explícita de validação do usuário;
- Quando se referir a algum tratamento demorado, a ação de seleção da opção deve ser separada da ação de ativação;
- O usuário não deve ser colocado diante de comando de dupla repercussão (por exemplo, salvar e fechar).

O critério elementar de controle do usuário refere-se à necessidade de

permitir que os usuários sempre estejam no comando dos processamentos do sistema, sendo possível que estes possam: interromper, cancelar, pausar ou continuar. Cada possível ação do usuário deve ser antecipada e as opções apropriadas devem ser oferecidas.

Permitir o controle pelos usuários favorece a aprendizagem, torna o sistema mais previsível e diminui a probabilidade de erros.

Exemplos de aplicação do critério de controle do usuário:

- O cursor só deve se movimentar de um campo a outro no formulário após o pressionamento de alguma tecla de validação (como por exemplo, as teclas *enter* ou *tab*);
- Os usuários devem ter controle sobre todas as páginas da tela;
- O sistema deve permitir que os usuários interrompam ou cancelem um processamento;
- O sistema deve prover de uma opção de “Cancelar” que possibilite ao usuário desfazer as alterações feitas e garantir que o sistema volte ao estado anterior do processo;
- Em um diálogo seqüencial, o sistema deve oferecer ao usuário opções para avançar, recuar, interromper, retomar ou finalizar o processamento.

2.5.4 Adaptabilidade

A adaptabilidade de um sistema refere-se à sua capacidade de se comportar conforme o contexto e as necessidades e preferências do usuário. Este critério é particularmente desejado em sistemas cujo conjunto de usuários é vasto e variado.

A interface deve oferecer maneiras variadas de realizar uma tarefa, possibilitando que o usuário possa escolher e dominar uma delas no curso de seu aprendizado (CYBIS, BETIOL e FAUST, 2007).

O critério de adaptabilidade é dividido em dois critérios elementares: a Flexibilidade e a Consideração da experiência do usuário.

A flexibilidade é um critério elementar que se refere à possibilidade do usuário personalizar a interface para que esta atenda suas estratégias de trabalho, seus hábitos e requisitos de trabalho.

Cybis, Betiol e Faust (2007) subdividem a flexibilidade em dois outros critérios elementares, a flexibilidade estrutural e a personalização. A flexibilidade estrutural se refere às diferentes formas que o sistema pode ser configurado ou personalizado para que os usuários realizem suas tarefas.

Uma interface estruturalmente flexível deve fornecer diferentes formas para a realização de entrada de dados (por exemplo: digitação, seleção e manipulação direta) e diferentes caminhos para atingir uma funcionalidade de uso freqüente (por exemplo, atalhos, ícones na barra de ferramentas ou opções em menus).

A personalização é destinada aos usuários mais experientes e se refere à disponibilização de meios para configuração da interface, levando em conta as exigências das tarefas, as estratégias, ou ainda, os hábitos de trabalho (CYBIS, BETIOL e FAUST, 2007).

Características de uma interface personalizável:

- A interface deve permitir a personalização das telas, inserindo ou retirando ícones, dados ou comandos;
- A interface deve possibilitar seqüências de ações automáticas (macros);
- A interface deve permitir que os valores padrões oferecidos pelo sistema sejam alterados.

O subcritério de consideração da experiência do usuário é aplicável quando o sistema é destinado a usuários de diversos níveis de experiência. Nesse sentido, é desejável que o sistema disponibilize diálogos e ações do tipo “passo a passo” para usuários novatos e atalhos para os usuários experientes.

Vale salientar que os usuários experientes após um longo tempo sem utilizar o sistema podem retroceder a uma situação de iniciante, e necessitem utilizar dos recursos destinados aos usuários novatos.

Para considerar a experiência dos usuários, a interface deve:

- Fornecer atalhos aos usuários experientes atalhos para acesso rápido das funções do sistema;
- Fornecer aos usuários intermitentes (usuários que usam o sistema esporadicamente, após longo período sem utilizar) diálogos passo a passo;
- Oferecer diálogos que atendam as necessidades de diferentes níveis de usuários.

2.5.5 Gestão de Erros

O critério de gestão de erros se ocupa dos meios de prevenção, redução ou recuperação de erros quando ocorrem. Como exemplo, considera-se como erros, as entradas inválidas de dados, formatos inválidos de dados e sintaxe de comando errada.

As interrupções causadas pelos erros têm conseqüências negativas sobre as atividades dos usuários, provocam o aumento do número de interações com o sistema, além de distúrbios organizacionais, como por exemplo, atrasos em processos.

O critério de gestão de erros é dividido em três critérios elementares: a Proteção contra erros, a Qualidade das mensagens de erros e a Correção de erros.

A proteção contra erros refere-se à possibilidade do sistema detectar e se prevenir de erros na entrada de dados, comandos errados ou as ações destrutivas dos usuários.

Para proteção contra erros:

- O sistema deve informar ao usuário sobre transações ou documentos pendentes que poderão ser perdidos caso este abandone o sistema antes de gravar;
- O sistema não deve oferecer uma opção destrutiva como opção padrão (por exemplo: uma caixa de diálogo questionando sobre um processo destrutivo de dados, com o botão de escolha selecionado em

"Executar", e não em "Cancelar");

- Detectar possíveis erros durante a digitação de uma entrada, evitando fazê-lo apenas no momento da validação do formulário (Figura 11).

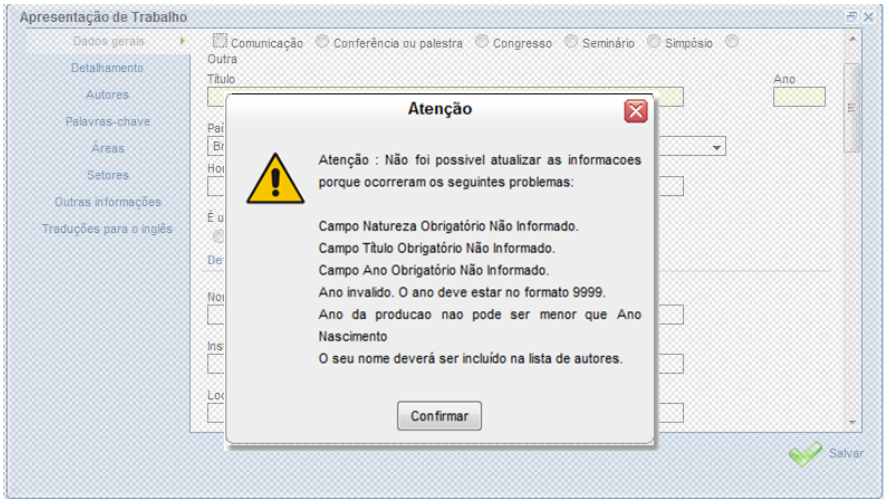


Figura 11 - Alerta de detecção de erros no preenchimento de formulário
Fonte: http://lattes.cnpq.br/, acesso em: 16 de maio de 2009

A qualidade das mensagens de erros é um critério elementar que se preocupa com a relevância, a legibilidade e a exatidão da mensagem de erro dada ao usuário.

Diretrizes para as mensagens de erros:

- As mensagens de erro devem indicar ao usuário a razão ou a natureza do erro, qual ação que originou o erro, o que deveria ter sido feito para evitar o erro e finalmente, informar o que deve ser feito para sair da situação de erro;
- A mensagem de erro deve ser orientada para a tarefa que o usuário está realizando, deve empregar termos específicos e ser breve;
- As mensagens de erro devem ter um tom neutro, não reprovador ou humorístico.

A correção dos erros é um critério que se refere à necessidade do sistema disponibilizar meios para permitir que o usuário corrija seus erros. Os erros são menos perturbadores quando podem ser facilmente corrigidos.

As seguintes recomendações facilitam a correção de erros por parte dos usuários:

- Após a detecção de erro, o sistema deve possibilitar que o usuário

refaça apenas a parte errada de uma entrada de dados, mostrando o erro e mantendo todos os outros dados intactos;

- O sistema deve oferecer funções que permitam ao usuário desfazer e refazer ações;
- O sistema deve permitir que o usuário corrija os erros, mesmo quando são detectados após o término de uma transação.

2.5.6 Homogeneidade e Coerência

O critério elementar de homogeneidade e coerência refere-se à forma como as escolhas de projeto da interface, tais como os códigos, denominações, formatos e procedimentos são mantidos iguais em contextos similares e modificados quando em contextos diferentes.

A manutenção padronizada da formatação, localização e sintaxe dos itens de uma interface para outra, favorece o reconhecimento, a localização; reduz o número de erros e contribui para que a interface seja mais previsível. A falta de homogeneidade/coerência aumenta consideravelmente o tempo de procura dos itens na interface e se constitui em um motivo para rejeição por parte do usuário.

Para Cybis, Faust e Betiol (2007) o critério de homogeneidade/coerência é de aplicação geral e é muito útil para usuários novatos ou intermitentes. Segundo os autores, estes usuários quando estiverem diante de uma interface desconhecida, empregarão as mesmas estratégias desenvolvidas em outras telas do mesmo sistema.

Algumas características de uma interface homogênea e coerente (Figura 12):

- Os títulos das janelas e diálogos estão sempre localizados no mesmo lugar;
- Os formatos de telas e procedimentos para acessar o menu de opções são similares;
- Os objetos de tela são distribuídos e denominados de forma padronizada.

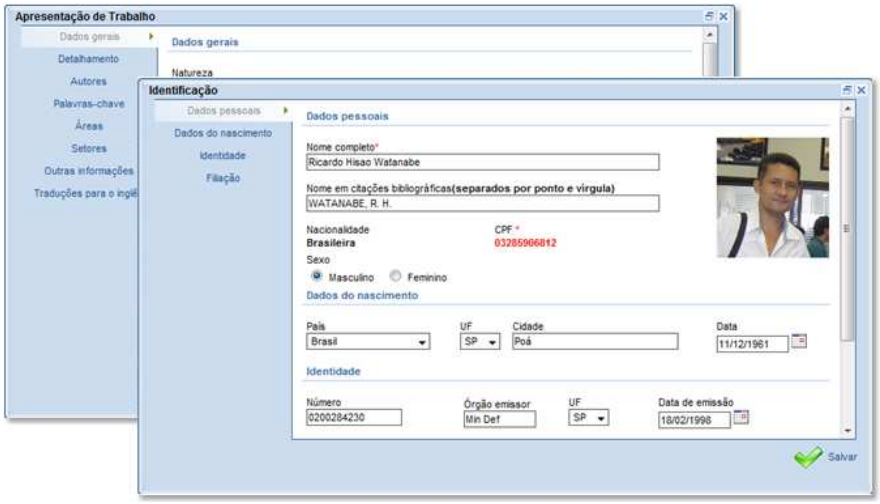


Figura 12 - Homogeneidade e coerência entre dois formulários
Fonte: http://lattes.cnpq.br/, acesso em: 16 de maio de 2009

2.5.7 Significado dos Códigos

O critério de significado dos códigos tem por objetivo adequar de forma significativa a relação entre o termo utilizado e o objeto que o representa. O estabelecimento de uma forte relação de significado entre os códigos e os itens ou ações a que se referem facilita a identificação e a relembrança por parte dos usuários.

No entanto, o uso de códigos e nomes não significativos pode levar os usuários a erros como escolher uma opção errada, deixar de informar um dado necessário.

Para Cybis, Betiol e Faust (2007) o critério de significado dos códigos é aplicável de forma geral, onde os usuários intermitentes e novatos são os mais beneficiados, visto que os usuários experientes já estarão acostumados com os possíveis problemas de linguagem existentes.

Para atendimento do critério elementar de significado dos códigos:

- Os títulos, nomes de funções e objetos de interação devem ser distintos e significativos;
- Os códigos devem ser distintos e representativos do conteúdo que representam (por exemplo: “M” para masculino e “F” para feminino e não “1” e “2” respectivamente);
- As abreviações devem ser de imediata interpretação.

2.5.8 Compatibilidade

O critério de compatibilidade refere-se à adequação entre: as características dos usuários, tais como a memória, as percepções, os padrões, habilidades e expectativas; as características das tarefas a serem executadas e a organização das entradas, saídas e diálogos do sistema.

Cybis, Betiol e Faust (2007) citam que o sistema deve ser compatível com as características do usuário em termos cognitivos (memória e percepção), demográficos (idade e sexo), culturais (hábitos), de competência (conhecimento e desempenho) e expectativas; deve haver um acordo entre as características do sistema e a organização das tarefas quanto às entradas, saídas e diálogos; e finalmente, o sistema deve ter coerência externa, ou seja, ser similar entre os diferentes sistemas executados em um mesmo ambiente.

Características de uma interface compatível:

- A apresentação das informações na interface deve estar conforme a organização lógica dos dados a serem digitados;
- Os procedimentos de diálogo devem ser compatíveis com a ordem que o usuário espera ou conforme o seu costume;
- Os formatos de datas devem respeitar o formato do país em que a aplicação será utilizada (por exemplo: no Brasil, o formato da data é dia/mês/ano e na Inglaterra é mês/dia/ano);
- Os termos empregados devem ser familiares aos usuários e relativos à tarefa a ser realizada;
- As expectativas e os costumes dos usuários devem respeitados quando da organização dos procedimentos e das tarefas no sistema;
- O sistema deve ser compatível com os documentos em papel, com as denominações e o vocabulário, minimizando as traduções, as transposições e as interpretações por parte do usuário.

2.6 Diretrizes para *sites* e aplicações Web

Nielsen e Loranger (2007) propuseram um conjunto de diretrizes de usabilidade baseadas em evidências empíricas, provenientes de testes de 716

Websites com 2.163 usuários espalhados pelo mundo e de uma fonte foi mais específica que contou com 69 usuários que testaram 25 *Websites* de vários gêneros (indústria, serviços, entretenimento, medicina e culturais).

Os autores afirmam que o segundo estudo foi mais específico por utilizar-se de estratégias observacionais, assistindo os usuários durante a realização dos testes. O método observacional possibilitou descobrir o que realmente os usuários "fazem" e não o quê eles "dizem fazer".

Para a experiência da observação direta, os usuários foram divididos em dois grupos: usuários pouco experientes e usuários muito experientes. A divisão foi feita com base em questões como: há quanto tempo está *on-line*; quanto tempo utiliza a Web por semana, sem contar com o tempo gasto em *e-mails*; hábitos avançados demonstrados; se conseguiam resolver seus próprios problemas nos computadores e até suas tendências atuais para a área de tecnologia.

A seguir são apresentadas as diretrizes propostas por Nielsen e Loranger (2007), que atualizam antigas diretrizes da década de 1990 à luz de suas pesquisas a partir de 2000. Estas se dividem em diretrizes relacionadas à navegação e a arquitetura da informação, textos, apresentação dos elementos das páginas e página principal.

2.6.1 Navegação e Arquitetura da Informação

A navegação e a arquitetura da informação são estruturas diretamente ligadas com a capacidade dos usuários localizarem as informações que desejam, pois, *designs* mal estruturados diminuem a velocidade de navegação e desencorajam os usuários a utilizá-los, o que, como conseqüência, fazem com que suponham que a funcionalidade não existe. Frustrados, os usuários abandonam o *site*.

Em geral, um grande equívoco de *design* é tentar estruturar um *site* Web de modo a refletir a estrutura hierárquica de uma empresa, pois tais estruturas não fazem sentido para os usuários. Exceto se o *site* for direcionado para o público interno de uma empresa, onde os funcionários estão acostumados com a estrutura interna.

Em uma estrutura ideal, o *site* deve organizar os elementos e as informações de maneira que façam sentido para o usuário e deve também, refletir as características, as informações e serviços disponíveis, considerando-se as seguintes recomendações: ser consistente, evitar o *design* rebuscado e a redundância.

A estrutura organizacional de qualquer *site* deve ser e se manter consistente de uma página para outra página, a fim de ajudar os usuários a localizarem as opções com o mínimo de suposições. A modificação da estrutura de navegação, drasticamente, de uma página para outra, desvia a atenção dos usuários quanto ao objetivo do *site*, pois são obrigados a aprender como utilizá-lo. Este problema é notoriamente observado em grandes *sites*, que normalmente são compostos por *sites* menores desenvolvidos por equipes diferentes.

Também deve ser evitada a navegação exótica e extravagante, pois sobrecarrega a página, dificulta a navegação, oculta os itens de navegação e é difícil de usar.

A duplicação de categorias de *links* ou a falta de distinção entre elas complica a página, obriga o usuário a pensar muito para entender a ordem e descobrir a diferença entre *links* com nomes semelhantes. Outra conseqüência da distribuição de *links* iguais ao longo da página é a diminuição da probabilidade dos usuários os verem, pois quanto mais objetos na página, maior também será a competição entre os elementos e nesse sentido, todos os itens perdem a importância.

Os *links* e nomes de rótulos devem ser curtos e sempre bem especificados, pois quando estão navegando, as pessoas ignoram grandes blocos de textos, indo diretamente para os *links* para terem uma idéia do significado do *site*.

Há também os chamados *links* profundos, que em *sites* organizados hierarquicamente, constituem-se de páginas mais internas e trazem os seguintes benefícios: aumentam a probabilidade dos usuários encontrarem o que procuram e de realizarem suas tarefas; aprimoram a usabilidade; e o conteúdo do *site* é substancialmente mais lido do que as páginas principais.

Para atendimento dos usuários que seguem *links* profundos, Nielsen e

Loranger (2007) propõem as seguintes diretrizes para as páginas internas:

- Informar ao usuário onde estão e como podem prosseguir para outras partes do *site*, incluindo os elementos abaixo em cada página:
    - O nome ou logotipo da empresa no canto superior esquerdo;
    - Um *link* de um único clique direto para a página principal;
    - Busca (preferivelmente no canto superior direito);
- Orientar o usuário quanto ao restante do *site*. Para *sites* com arquitetura hierárquica de informações, utilizar uma *breadcrumb trail* (“trilha de migalhas de pão”) a fim de indicar a localização atual do usuário no contexto hierárquico do *site*, permitindo que estes subam ou desçam pela hierarquia.
- Não pressupor que os usuários tenham seguido um caminho reto até à página atual, caso eles tenham optado por um caminho diferente do qual se esperava, significa que eles não viram as informações contidas nas páginas no nível mais alto.

Por exemplo, a Figura 13 apresenta a parte superior de uma página interna do *site* do Ministério da Ciência e Tecnologia (MCT) que atende às diretrizes de *links* profundos. No topo, à esquerda (item nº 1), a página apresenta o nome do MCT, o campo de busca na parte superior um pouco para a direita (item nº 3), um *link* direto para a página inicial (item nº 2) e um *breadcrumb-trail*, que no exemplo indica a seguinte hierarquia “Página Inicial->Legislação->Pesquisa Avançada” (item nº 2).

Figura 13 - Interface com itens de navegação e arquitetura da informação (com destaques do autor)
Fonte: http://www.mct.gov.br/index.php/content/view/8039.html?tema=Ant%E1rtica, acesso em: 16 de maio de 2009

Uma página deve indicar facilmente os itens que podem ser acessados (clicados). Quando os *links* não são bem indicados, os usuários são obrigados a se esforçarem mais e fazerem suposições sobre o que pode ser clicado. Em conseqüência, os usuários podem desistir prematuramente ou acreditarem que já exploraram tudo.

Em geral, os *links* são representados em azul. Quando por razões estéticas os *links* não puderem ser azuis, pode-se utilizar textos em negrito, destacá-los quando o ponteiro do *mouse* passa por cima, utilizar elementos gráficos salientes ou destacados. Deve-se evitar a cor azul para textos que não são *links*, pois o padrão para identificação de *links* é o sublinhado e a cor azul.

Outro recurso muito utilizado são os menus em cascata, que poupam o espaço limitado da página e se constituem em uma ferramenta de navegação muito utilizada com a qual os usuários aprendem a se adaptar a eles.

Apesar das vantagens oferecidas quanto ao espaço, os menus podem ser problemáticos principalmente se apresentarem longas listas de opções. Para esses casos, pode-se optar por listas no formato de hipertexto, em que há espaço para detalhamento das opções.

Também devem ser evitados os menus com mais do que dois níveis, pois encobrem a página, são difíceis de controlar, dificultam a navegação e a busca da opção desejada.

2.6.2 Textos

A tipografia e os esquemas de cores adequados desempenham um papel essencial em uma página Web, causam uma boa primeira impressão, dá aos usuários uma idéia do *site* e informa ao usuário sobre o que pode ser feito.

A escolha da tipografia e de esquemas de cores baseada unicamente na marca, nas preferências pessoais ou estéticas, podem ter como conseqüências negativas: textos muito pequenos, confusos e não redimensionáveis; cores de textos que não fornecem um bom contraste com o fundo; e textos obscurecidos por elementos de leiaute.

A tipografia e os esquemas de cores escolhidos devem comunicar algo, deve ser legível, refletir a personalidade e o tom do *site* aos usuários para que estes encontrem o que procurem e consigam realizar suas tarefas.

Nos parágrafos seguintes serão expostas algumas das características das fontes, as principais diretrizes para os estilos de fontes, tamanho, cores e contrastes adequados.

As fontes são normalmente divididas em duas famílias, as fontes com serifa e sem serifa (Figura 14). As fontes com serifas são caracterizadas por apresentarem arremates nas extremidades de cada letra ou outros enfeites e variações sutis. As fontes sem serifa são mais simples e não apresentam os detalhes decorativos.

Figura 14 - Fonte sem serifa (à esquerda) e com serifa (à direita)
Fonte: o autor

As fontes com serifas favorecem a leitura de material impresso como livros e revistas, porém, para a leitura diretamente nos computadores, os estudos mostram que as letras sem serifas são as mais indicadas, devido às telas não oferecerem a qualidade tipográfica para as serifas.

Para o corpo do texto, a fonte deve ter o tamanho adequado para não descartar nenhum dos possíveis usuários (Quadro 2). De qualquer forma, a fonte escolhida deve ter no mínimo dez pontos de tamanho. Textos menores reduzem a legibilidade por não serem distintos, mesmo para as pessoas com visão normal.

Quadro 2 - Tamanho de fontes recomendadas

Tipo de público	Tamanho em pontos
Público geral	10 - 12
Idosos e pessoas com deficiências visuais	12 - 14
Crianças e outros leitores iniciantes	12 - 14
Adolescentes/adultos	10 - 12

Fonte: Adaptado de Nielsen e Loranger (2007, p. 221)

Utilizar-se de fontes muito pequenas para acrescentar mais conteúdo em uma página não significa que os usuários vão ler mais, na realidade, eles provavelmente

lerão menos. A justificativa é que textos densos, com muitas palavras, afastam os usuários.

Ao escolher os tipos de fontes para as páginas, deve-se optar por fontes normalmente disponíveis na maioria dos navegadores, pois caso a fonte não esteja disponível, os sistemas dos usuários talvez utilizem uma fonte não adequada para a página e, conseqüentemente, esta não se apresentará como pretendido. No Quadro 3, tem-se a lista das fontes comuns, que normalmente vêm instaladas na maioria dos navegadores existentes, suas principais características, a legibilidade *on-line* e as indicações mais adequadas.

Quadro 3 - Características das fontes pré-instaladas nos navegadores

Fonte	Legibilidade *on-line*	Caractere/Tom
Arial	Legível em tamanhos razoáveis. Boa na fonte de 10pt ou acima	Moderna, limpa, básica sem firulas. Geralmente a preferida por pessoas de todas as idades
Comic Sans MS	Fonte decorativa, mas difícil de ler *on-line*, mesmo em tamanhos grandes	Amigável, juvenil, divertido e informal. Não apropriada para *sites* mais sérios ou profissionais
Georgia	A melhor fonte com serifa projetada para leitura *on-line*. Geralmente boa para tamanhos de 10pt ou acima	Aparência tradicional, mas com um visual mais moderno e legível.
Impact	Geralmente utilizada para impressão. Não recomendada para visualização *on-line*. Legibilidade ruim mesmo em tamanho grande	Escura. Não adequada a blocos de conteúdo. Pode ser utilizada algumas vezes para títulos curtos
Times New Roman	Boa para materiais impressos. A legibilidade na tela rapidamente diminui em tamanhos menores. Somente boa no corpo do texto. Tamanho de 12pt ou superior	Aparência tradicional. Não recomendada para *sites* profissionais. Não preferida pela audiência de qualquer idade.
Trebuchet MS	Legível em tamanhos razoáveis. Boa no tamanho 10pt ou acima	Moderna, simples e aguçada
Verdana	A fonte *on-line* mais legível, mesmo em texto pequeno	Profissional, simples e moderna para uso no corpo de texto, em que a legibilidade é fundamental. Altamente cotada na preferência dos usuários

Fonte: Adaptado de Nielsen e Loranger (2007, p. 233)

O número de tipos de fontes e cores deve ser limitado e aplicado de forma consistente, pois as variações ajudam as pessoas a diferenciar títulos, criam uma ordem e comunicam elementos hierárquicos no *site*.

Os textos devem ser escritos com letras maiúsculas e minúsculas normalmente, pois textos com todas as letras em maiúsculas reduzem a velocidade de leitura, tornam a aparência condensada, agressiva e não-profissional.

2.6.3 Apresentação dos Elementos da Página

A apresentação dos elementos da página deve enfatizar as informações de maior prioridade para os usuários a fim de atrair sua atenção. Deve-se estruturar a página em ordem de prioridade, agrupar as áreas relacionadas, alinhar os elementos adequadamente para criar ordem, posicionar os elementos no lugar em que os usuários esperam encontrar (por exemplo, menus à esquerda) e evitar a poluição com elementos desnecessários.

Deve-se atentar para as situações em que o tamanho da página ultrapassa uma tela, pois a maioria dos usuários ignora o conteúdo após a final da tela, conforme mostrou a pesquisa realizada por Nielsen e Loranger (2007), na qual, das páginas que apresentavam conteúdo após o fim da tela, foram acessadas (roladas) por 38% dos usuários novatos e 46% dos usuários considerados experientes.

Isto ocorre porque a tendência dos usuários é olhar exatamente para o meio da página, se necessário, olharão para a esquerda ou para a parte superior para encerrar a navegação. Os usuários raramente olham para a margem direita a fim de verificarem a barra de rolagem e pressupõem que os elementos abaixo da tela não têm importância.

Ao se oferecer páginas longas que exijam rolagem, deve-se atentar para que o conteúdo mais importante esteja na parte superior e que a estrutura da página forneça dicas visuais para que os usuários vejam e percebam o valor da rolagem.

Outro componente de uso muito comum quando se deseja coletar informações dos usuários são os formulários. Os formulários disponibilizam os elementos com os quais os usuários interagem, como por exemplo, as caixas de

textos, listas e botões de seleção.

O projetista deve atentar para a disposição dos elementos do formulário e seus respectivos rótulos, pois, a desorganização, a dispersão e a falta de uma indicação clara entre rótulos e elementos de formulário, induzem os usuários a cometerem erros de preenchimento ou de escolha.

No projeto de uma página deve-se valorizar os espaços em branco, pois tais espaços direcionam para os pontos mais importantes sem que o usuário tenha que se esforçar para encontrá-los. Páginas muito poluídas visualmente, com muitos elementos, sobrecarregam a visão do usuário, dificultando a identificação rápida do que é importante.

2.6.4 A Página Principal (*Homepage*)

Segundo Nielsen e Loranger (2007), a página principal recebe, de forma desproporcional, muito mais visitas do que qualquer outra página interna do *site*. O estudo mostrou que a página principal foi a primeira página do *site* a ser visitada por 40% dos usuários. Outro fator a considerar é que em geral os *sites* têm centenas ou milhares de páginas internas e que mesmo quando os usuários entram por essas páginas, acabam acessando a página principal a fim de ter uma idéia geral da finalidade do *site*.

Outra descoberta do estudo de Nielsen e Loranger é quanto ao tempo médio em que os usuários permanecem na página principal que é de 30 segundos. Os usuários de pouca experiência gastaram 35 segundos e usuários de muita experiência gastam 20 segundos.

Tendo em vista o curto espaço de tempo que os usuários permanecem na página principal, Nielsen e Loranger (2007) estabeleceram quatro pontos (diretrizes) que uma página principal precisa informar aos novos usuários, além de uma recomendação para que todas as mensagens sejam simples e diretas: informar o *site* a que chegaram, informar os benefícios que a empresa oferece, informar algo sobre a empresa e seus produtos mais recentes ou novos desenvolvimentos e informar as opções para os usuários e como chegar à seção mais relevante.

O exemplo da Figura 15 apresenta a página do projeto Fome Zero da Presidência da República. No topo da página encontra-se a informação do *site* em que se está no (item nº 1). À direita estão os benefícios que a empresa oferece (item nº 2). Informação sobre a empresa e seus produtos mais recentes ou novos desenvolvimentos encontra-se acima, na área de conteúdo (item nº 3). À esquerda encontra-se o menu de navegação para que os usuários acessem as opções mais relevantes (item nº 4). Na área de conteúdo encontram-se as notícias que são extremamente simples e diretas (item nº 5).

Figura 15 - *Home page* com diretrizes de elementos de página (com destaques do autor)
Fonte - http://www.fomezero.gov.br/, acesso em: 16 de maio de 2009

A colocação de *links* diretos na página principal para um número pequeno de tarefas de alta prioridade e de alto índice de acessos, independentemente da estrutura, constitui-se em uma das estratégias mais bem sucedidas de *design*. Ao encurtar e simplificar o acesso a essas áreas evita-se que os usuários fiquem perdidos ou impacientes por serem obrigados a navegar pela estrutura.

Infelizmente o número de *links* com esse objetivo na página principal não deve ser grande, entre três e cinco *links*, um número maior pode anular o propósito e confundir o usuário.

Por ocasião da concepção de uma página Web, o projetista deve atentar para os princípios e as diretrizes apresentadas neste capítulo, porém, a construção de aplicações Web também exige a adoção de métodos estruturados, participação do

usuário, planejamento e atividades que são disciplinadas pela Engenharia de Usabilidade.

2.7 Engenharia de Usabilidade

Historicamente, a engenharia de usabilidade originou-se de iniciativas de cientistas como Card, Moran e Newell (1983) e Norman (1989), que produziram os primeiros estudos de estruturas e processos cognitivos, com o objetivo de favorecer a concepção de Interfaces Humano-Computador mais adaptadas (CYBIS, BETIOL e FAUST, 2007).

Segundo Mayhew (1999), a engenharia de usabilidade é uma disciplina que fornece métodos estruturados para alcançar a usabilidade em projetos de interface de usuário durante o desenvolvimento de um produto. Trata-se de uma disciplina com raízes em outras disciplinas básicas, tais como: a psicologia, a etnografia, a ciência cognitiva e a engenharia de software.

A razão básica para a existência da engenharia de usabilidade é a impossibilidade de se conceber uma ótima interface de usuário a partir apenas, da melhor opinião de um designer. Os usuários têm um potencial infinito para entender de forma inesperada os elementos da interface e realizarem suas tarefas de forma diferente do que os designers imaginam (NIELSEN, 1994, tradução nossa).

Cybis, Betiol e Faust (2007) sugerem que, para o entendimento da proposta da engenharia de usabilidade, esta deve ser vista em relação à outra proposta de engenharia similar, como engenharia de software. Segundo os autores, a engenharia de software se ocupa do desenvolvimento núcleo funcional do sistema que é formado por estruturas de dados, algoritmos e recursos computacionais. As possibilidades de sucesso no desenvolvimento do núcleo funcional são maiores, pois engenheiro possui o conhecimento, a competência e o ferramental de engenharia de software que o auxiliam na elaboração de códigos eficazes.

A engenharia de usabilidade, por sua vez, ocupa-se da interface do usuário, que é o componente do sistema interativo, constituído de apresentações, estruturas de diálogo, painéis com informações, dados, controles, comandos e mensagens.

Preece, Rogers e Sharp (2005) ressaltam a importância da utilização de modelos de ciclos de vida na engenharia de usabilidade, haja vista que estes favorecem o entendimento de quais atividades de engenharia estão envolvidas e como elas se relacionam.

2.7.1 Ciclos de Vida de Engenharia de Usabilidade

Os ciclos de vida são utilizados para representar um conjunto de atividades e a forma como elas se relacionam. Para Preece, Rogers e Sharp (2005), os ciclos de vida se tornaram muito populares por permitir que os desenvolvedores, e particularmente os gerentes, tenham uma visão geral do esforço global de desenvolvimento; do progresso alcançado; das metas estabelecidas; dos recursos alocados e dos resultados especificados.

A utilização de modelos prescritivos de processo que definam um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalho, não garante a perfeição, mas fornece um roteiro útil para o trabalho de engenharia (PRESSMANN, 2006).

Os modelos de ciclo de vida existentes apresentam diferentes graus de sofisticação e complexidade. Para projetos menores com poucos desenvolvedores, a utilização de um modelo mais simples provavelmente seria o mais adequado. Porém, para grandes projetos, com centenas de desenvolvedores e milhares de usuários, a utilização de um modelo simples não forneceria a estrutura e a disciplina de gerência necessária para a construção de um sistema usável (PREECE, ROGERS e SHARP, 2005).

Preece, Rogers e Sharp (2005) citam ainda, a existência de três características-chave nos processos de engenharia de usabilidade: o foco no usuário, que é a base central do processo; a identificação e documentação dos objetivos específicos dos usuários, por auxiliarem os desenvolvedores a escolherem entre as diferentes opções de *design* e verificarem seu progresso durante o desenvolvimento; e a iteração, por permitir refinar o *design* com base no *feedback*.

Para Cybis, Betiol e Faust (2007), o ciclo de engenharia de usabilidade

descreve as atividades, as relações e tem foco no usuário. Deve permitir a realização de sucessivos ciclos de análise, concepção e testes, com o necessário *feedback* dos resultados dos testes, de um ciclo a outro, identificando e refinando continuamente o conhecimento sobre o contexto de uso e as exigência em termos de usabilidade (Figura 16).

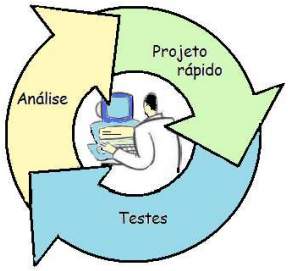


Figura 16 - Ciclo de desenvolvimento centrado no usuário
Fonte: Adaptado de Cybis, Betiol e Faust (2007, p. 104)

A utilização de ciclos contínuos possibilita a construção de versões intermediárias da interface do sistema, que são submetidas a testes de uso pelos representantes dos usuários. Nesse processo, tem-se inicialmente, versões “grosseiras” de interfaces, mas com o avanço do desenvolvimento, concebem-se protótipos e versões mais acabadas do sistema, em simulações mais fidedignas a cada ciclo (CYBIS, BETIOL e FAUST, 2007).

O desenvolvimento centrado no usuário tem como propósito conceber sistemas, pensando em quem vai utilizá-los: os usuários. Para esse fim, a norma ISO 13407 (1999), que trata do desenvolvimento de sistemas interativos centrado no usuário, sugere que representantes de todos os tipos de usuários finais do sistema façam parte da equipe de desenvolvimento de software.

2.7.2 Desenvolvimento Centrado nos Usuários

A abordagem tendo como foco o usuário é especialmente indicada no desenvolvimento de sistemas com transações onde usuários tenham expectativas de eficácia e de eficiência, além de proporcionar os seguintes benefícios: desenvolvimento de sistemas mais intuitivos; fáceis de aprender e de utilizar; causam menos fadiga, proporcionam mais conforto ao usuário; e garantem maior qualidade para o resultado final (CYBIS, BETIOL e FAUST, 2007).

Para Preece, Rogers e Sharp (2005), o envolvimento dos usuários durante o

desenvolvimento do sistema é a melhor maneira de assegurar que se esteja levando em conta as atividades dos usuários, pois dessa forma os desenvolvedores têm um melhor entendimento das necessidades e dos objetivos desses usuários. Outros benefícios são: o gerenciamento da expectativa e o sentimento de apropriação.

O gerenciamento da expectativa consiste em garantir que as expectativas do usuário quanto ao sistema sejam realistas, evitando surpresas quando da entrega do produto. Com a participação do usuário no desenvolvimento, o gerenciamento da expectativa fica mais fácil, pois eles estarão aptos a verificar desde os estágios iniciais do projeto quais são as capacidades do produto.

O sentimento de apropriação fica evidenciado quando o usuário participa do desenvolvimento, pois este se sente mais envolvido ao perceber que contribuiu para o desenvolvimento do software (sente-se como “o dono do sistema”), mostrando-se mais receptivo quando da entrega do software pronto.

Em síntese, o objetivo principal da participação do usuário no desenvolvimento é fazer com que o futuro sistema ou produto, tenha maior qualidade e seja facilmente aceito por este usuário. Porém, é preciso investir em técnicas de planejamento e execução adequadas, pois a falta de gerenciamento dessa atividade pode se constituir em um grande risco, com perda de tempo e de recursos em função da variabilidade e da subjetividade que caracterizam os resultados das atividades com usuários (CYBIS, BETIOL e FAUST, 2007).

O gerenciamento da participação do usuário no desenvolvimento constitui-se em estabelecer diferentes graus de envolvimento do usuário, onde a cooperação em meio turno ou em tempo integral, durante uma parte ou por todo o projeto, pode trazer vantagens e desvantagens (PREECE, ROGERS e SHARP, 2005).

Cybis, Betiol e Faust (2007) descrevem três formas de envolvimento do usuário no desenvolvimento de um sistema ou produto: envolvimento informativo, envolvimento consultivo e envolvimento participativo.

No envolvimento informativo o usuário é visto como uma fonte de informações, que são coletadas por meio de entrevistas, questionários ou de

observações do seu trabalho.

O envolvimento consultivo é aquele em que o usuário é chamado para opinar sobre soluções de projeto, elaboradas a partir de informações coletadas do próprio usuário ou não. Da mesma forma que no envolvimento informativo, este envolvimento pode ser feito por meio de entrevistas ou questionários.

O envolvimento participativo constitui-se no nível mais elevado de envolvimento, onde o usuário tem poder decisório sobre o projeto, necessitando um esforço maior de planejamento, organização e execução do que os outros tipos de envolvimento.

É desejável que o envolvimento dos usuários deva se dar como uma combinação dos três níveis anteriormente citados, e que para isso é necessário conscientização, mudanças organizacionais e culturais, na empresa e na equipe de desenvolvimento: “Deve-se buscar informações junto ao usuário, consultá-lo sobre decisões de projeto e lhe passar o poder de tomar determinadas decisões.” (CYBIS, BETIOL e FAUST, 2007, p. 109)

O desenvolvimento de sistemas, tendo como foco o envolvimento dos usuários, será demonstrado nas atividades da engenharia de usabilidade.

2.7.3 Atividades da Engenharia de Usabilidade

O processo de *design* de interfaces de usuário compreende atividades iterativas que implicam em passar por várias fases em níveis diferentes de detalhes. O processo consiste em: entender as necessidades dos usuários a partir da coleta de requisitos, elaborar os modelos conceituais com base nos requisitos coletados, construir protótipos, avaliá-los quanto às questões de usabilidade e objetivos da experiência do usuário, decidir sobre as implicações observadas a partir das avaliações, realizar as alterações nos protótipos e avaliar os protótipos novamente (PREECE, ROGERS e SHARP, 2005).

Para o processo de *design*, Preece, Rogers e Sharp (2005) propõem quatro atividades básicas, que por serem genéricas, podem ser encontradas em outras áreas de *design* como, por exemplo, no *design* arquitetônico. As atividades

compreendem: a identificação e estabelecimento dos requisitos, que busca conhecer quem são os usuários-alvo e o tipo de suporte que o novo sistema pode oferecer; o estudo das opções alternativas de *design*; construção de versões interativas com a elaboração de protótipos a fim de fornecer aos usuários uma melhor indicação do *design* que está sendo construído; e a execução de avaliações, a fim de determinar a usabilidade e a aceitação do produto.

2.7.3.1 Identificação e Estabelecimento dos Requisitos

O objetivo desta atividade é entender ao máximo possível os usuários, seu trabalho e o contexto deste trabalho, de forma que o sistema a ser construído ofereça o suporte necessário para atingir seus objetivos.

A partir da identificação das necessidades, produz-se um conjunto de requisitos estáveis a fim de formar uma base para elaboração dos *designs* (PREECE, ROGERS e SHARP, 2005).

A busca por requisitos estáveis deve-se ao fato que, segundo Pressman (2006), os requisitos para sistemas computacionais mudam e que o desejo de mudá-los persiste ao longo da vida do sistema.

Preece, Rogers e Sharp (2005, p. 224) definem que: “Um requisito consiste em uma declaração sobre um produto pretendido que especifica o que ele deveria fazer ou como deveria operar”. As autoras acrescentam que a atividade de estabelecimento de requisitos tem como objetivo torná-los mais específicos, não-ambíguos e mais claros.

Em síntese, as atividades iniciais de um processo de engenharia de usabilidade consistem em apoiar os projetistas de interfaces na busca de informações sobre o contexto de uso e sobre a usabilidade do sistema a ser construído (CYBIS, BETIOL e FAUST, 2007).

A identificação das necessidades e o estabelecimento de requisitos têm importância fundamental no processo, pois caso não seja executada corretamente e os requisitos estejam errados, o produto poderá ser ignorado ou pior, desprezado pelos usuários, ocasionando, tanto para o desenvolvedor como para o cliente,

frustações, retorno de investimento perdido, perda da confiança do cliente e assim por diante. Preece, Rogers e Sharp (2005, p. 420) acrescentam o seguinte dilema a respeito do que os usuários dizem e o que realmente fazem:

> O que os usuários dizem nem sempre é o que fazem. As pessoas algumas vezes dão respostas que consideram mostrar-lhes em sua melhor forma, ou apenas podem esquecer o que aconteceu ou quanto tempo passaram realizando uma atividade em particular.

Em uma pesquisa realizada no ano de 2000, que envolveu entrevistas com 38 profissionais de Tecnologia da Informação (TI) a fim de se investigar as causas da falha de projetos de TI, verificou-se que as questões sobre requisitos figuraram com alto índice nas respostas. Segundo a pesquisa, o item “definição de requisitos” foi apontado como o estágio de projeto que mais causou falhas; os “objetivos e requisitos pouco claros” foram apontados como a maior causa de falhas em geral e a “clareza e o detalhamento dos requisitos” foram indicados como um fator crítico de sucesso (TAYLOR, 2000).

Existem muitas técnicas para auxiliar os usuários e os desenvolvedores na busca de um entendimento melhor sobre os requisitos, mas pouco se tem estudado com o objetivo de se identificar as causas dos requisitos mal-entendidos. Requisitos mal-entendidos induzem a produção de soluções erradas de sistemas, contribuindo para o aumento do custo do projeto, aumento do tempo de desenvolvimento e diminuição do sucesso do software (MCALLISTER, 2006).

Buscando elencar quais os fatores que mais contribuem para os requisitos mal-entendidos, McAllister (2006) realizou uma pesquisa com três grupos de usuários e três grupos de desenvolvedores, com seis a oito participantes por grupo, representantes dos setores de indústria, tecnologia, governo e empresas sem fins lucrativos.

Como resultado da pesquisa e consolidação (Tabela 1), foram elencados 14 fatores que, na opinião dos usuários e dos desenvolvedores, constituem-se nos mais relevantes e diretamente relacionados com requisitos mal-entendidos. O conhecimento desses fatores favorece a adoção de estratégias pelas equipes, a fim

de mitigar os requisitos mal-entendidos.

Tabela 1 - Fatores que contribuem para os de requisitos mal-entendidos

Fatores	Usuários	Desenvolv.
Perspectivas diferentes entre usuários e desenvolvedores	18%	11%
Experiências passadas dos usuários podem direcionar os requisitos para soluções individuais	15%	-
Falta de envolvimento dos usuários-chave	13%	17%
Incertezas dos usuários quanto às suas reais necessidades	13%	10%
Mudanças de requisitos	10%	-
Prazos curtos para coleta de requisitos	10%	8%
Pouco conhecimento de negócios por parte dos desenvolvedores	10%	9%
Incertezas dos usuários quanto ao seu papel na determinação dos requisitos	8%	-
Falta de conhecimento dos usuários das possibilidades da TI	5%	-
Falta de clareza de como se processará a atividade de coleta de requisitos	-	10%
Falta de comunicação efetiva e constante entre todos os participantes	-	9%
Documentação ineficiente ou mal redigida	-	9%
Dificuldade na revisão de requisitos por esta ter sido redigida de forma muito técnica	-	9%
Desconhecimento dos objetivos organizacionais pelos usuários	-	8%

Fonte: Adaptado de McAllister (2006, p. 148)

Conforme mostrado na Figura 17, os fatores percebidos pelos usuários e desenvolvedores, relacionados com requisitos mal-entendidos, afetam o domínio do problema, que é constituído pelos objetivos da empresa, dos usuários, dos desenvolvedores e as exigências do negócio, que por sua vez, dão origem às necessidades em sistemas de informação.

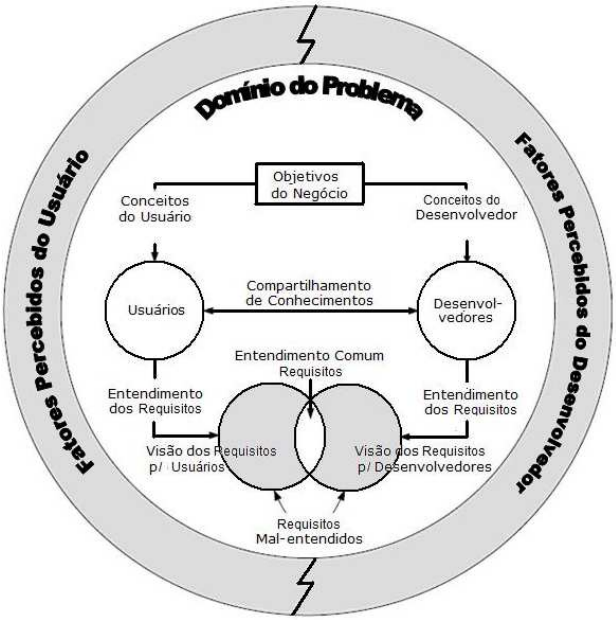


Figura 17 - Influência de usuários e desenvolvedores na determinação de requisitos mal-entendidos
Fonte: Adaptado de McAllister (2006, p. 149)

Para atendimento das necessidades em sistemas de informação, usuários e desenvolvedores trabalham juntos, compartilham conhecimentos e formulam seus conceitos a respeito dos objetivos do negócio. A partir dos conceitos sobre os objetivos do negócio, usuários e desenvolvedores passam a entender, dentro de suas perspectivas, os requisitos.

Ao final do processo, pode ocorrer que o entendimento dos usuários e desenvolvedores sobre o conjunto de requisitos coletados seja desigual, pois apenas uma parte dos requisitos é entendida de forma correta, os demais, que não são comuns por falta de entendimento entre os dois grupos, constituem-se os requisitos mal-entendidos.

Usuários compartilham a idéia de que não foram treinados ou qualificados para determinação de requisitos e os desenvolvedores alegam que não entendem, de forma adequada os negócios dos usuários. Diante dessa constatação, McAllister (2006) afirma que a solução não está apenas na determinação da técnica de coleta de requisitos; é preciso que usuários e desenvolvedores estejam motivados para participar e que a técnica de prospecção de requisitos favoreça a comunicação.

A coleta de requisitos constitui-se em uma das atividades de maior causa em falhas de projetos, por esta razão, a equipe de desenvolvimento deve atentar para os fatores de risco estudados e para a escolha e uso das técnicas de coleta de requisitos que serão apresentadas nas seções seguintes.

a. Entrevistas

Para Cybis, Betiol e Faust (2007), a entrevista é uma técnica de coleta de requisitos, utilizada para descobrir fatos e opiniões dos potenciais usuários do sistema a ser concebido e é, geralmente, conduzida por um entrevistador (integrante da equipe de desenvolvimento) falando com um usuário ou um grupo de usuários. Os resultados das entrevistas geram relatórios que serão cuidadosamente analisados pela equipe de desenvolvimento para assegurar seu impacto na concepção das interfaces do sistema.

Para que o entrevistador seja capaz de conduzir adequadamente a entrevista,

ele deve ser treinado, capaz de registrar as reações do entrevistado às perguntas e habilidoso para registrar as respostas de maneira completa e suficiente. Algumas características incovenientes para um entrevistador seriam problemas de dicção, timidez, apresentação deficiente ou opinião apaixonada sobre o assunto (GIL, 2007).

Durante a execução de uma entrevista, o entrevistador deve sempre assumir uma postura neutra e analítica e esta deve ocorrer em quatro fases distintas: aquecimento, introdução, entrevista e encerramento (CYBIS, BETIOL e FAUST, 2007).

Na fase de aquecimento, o entrevistado e o entrevistador se apresentam e conversam um pouco para que se conheçam e possam falar rapidamente sobre temas neutros ao assunto da entrevista.

A fase de introdução é destinada a informar ao entrevistado sobre o contexto da entrevista, para assegurá-lo das questões éticas evolvidas e para perguntar-lhe se a conversa pode ser gravada, se for o caso.

A fase de corpo da entrevista é a fase mais importante da atividade, quando o entrevistador apresenta as questões de forma lógica.

Na fase de encerramento, o entrevistador deve se certificar de que todos os assuntos foram abordados, resumir todos os tópicos tratados e solicitar ao entrevistado para que valide o relato.

Para Preece, Rogers e Sharp (2005), dependendo de quão rigorosamente o entrevistador se atém a um conjunto predeterminado de questões, as entrevistas podem ser classificadas como estruturadas, não-estruturadas ou semi-estruturadas.

As entrevistas não estruturadas assemelham-se mais a uma conversação, focando um tópico em particular, onde as perguntas do entrevistador são abertas e o formato e o conteúdo das respostas não são predeterminados. Nessa modalidade, tanto o entrevistador como o entrevistado podem direcionar a entrevista. Para que as respostas às questões relevantes sejam obtidas é necessário que o entrevistador seja organizado e que tenha um plano dos principais pontos a serem tratados.

Nas entrevistas estruturadas, as perguntas são predeterminadas e exigem uma resposta fechada. Esta modalidade é útil quando se tem um claro entendimento das metas do estudo e quando as perguntas específicas podem ser identificadas. Para que a atividade tenha sucesso, as perguntas devem ser curtas e claramente escritas.

As entrevistas semi-estruturadas misturam as características de entrevistas estruturadas e não-estruturadas. O entrevistador alterna questões predeterminadas e questões abertas, onde a consistência da entrevista é mantida a partir de um roteiro básico que guia o entrevistador durante a execução da sessão. A entrevista inicia-se pelas questões predeterminadas e após a resposta, o avaliador faz com que o entrevistado estenda sua resposta até que nenhuma nova e relevante informação seja dada.

b. Entrevistas em Grupos de Foco

A técnica de grupos de foco consiste em uma reunião com uma amostra representativa de usuários que manifestam suas opiniões sobre determinado assunto. A reunião é conduzida por um moderador que, a partir de um roteiro previamente preparado, desenvolve os assuntos a serem tratados. Os usuários, cujo número normalmente varia de seis a doze participantes, devem ser convidados individualmente e informados sobre o que é uma reunião desse tipo, como se processa e qual os objetivos (CYBIS, BETIOL e FAUST, 2007).

Preece, Rogers e Sharp (2005) citam que a técnica de grupos de foco apresenta como vantagens a possibilidade do levantamento de questões diferentes e sensíveis, as quais de outra forma poderiam ter sido esquecidas e é atrativa por apresentar baixo custo, fornecer resultados rápidos e poder ser facilmente trabalhada a fim de se coletar mais dados.

Como desvantagem, a técnica exige um moderador habilidoso na condução da reunião, que seja capaz de evitar a perda de tempo com questões irrelevantes, além da dificuldade em reunir as pessoas em um local apropriado.

c. Questionários

Os questionários se constituem em uma série de perguntas projetadas com o objetivo de se obter informações específicas dos usuários. Sua elaboração pode exigir diferentes formas de respostas, tais como: “sim ou não”, escolha entre opções pré-estabelecidas, justificativas, comentários ou uma resposta mais longa (PREECE, ROGERS e SHARP, 2005).

Cybis, Betiol e Faust (2007) citam que a elaboração dos questionários deve sempre apresentar questões objetivas, amigáveis, fáceis de responder e tratar. O foco do questionário deve estar direcionado para a identificação de quais são as principais decisões e dúvidas da equipe de projeto.

Pelo fato dos questionários poderem ser enviados aos usuários no formato eletrônico, disponibilizados em um *site* da Web, ou ainda, distribuídos em papel. A administração desta técnica é feita à distância, pois o usuário não é ajudado a responder às perguntas (PREECE, ROGERS e SHARP, 2005).

d. Observação do Usuário

A observação do usuário constitui-se em uma técnica de coleta de requisitos onde um membro da equipe de *design* observa o usuário em suas tarefas diárias, procurando entender como o trabalho realmente acontece em seu ambiente natural (CYBIS, BETIOL e FAUST, 2007).

A técnica de observação é muito útil devido à dificuldade que as pessoas têm em explicar o que fazem ou mesmo descrever precisamente como realizam suas atividades. Isto significa que é pouco provável que se obtenha uma história completa e verdadeira dos usuários utilizando-se apenas de entrevistas ou questionários (PREECE, ROGERS e SHARP, 2005).

O planejamento da atividade deve incluir o estabelecimento de seus objetivos, como por exemplo, situações de normalidade, situações críticas ou de aprendizado e definir a forma como os acontecimentos serão registrados: notas manuscritas, fotos, filmagens, gravações ou a combinação destes meios (CYBIS, BETIOL e FAUST, 2007).

A principal recomendação para o sucesso da atividade é certificar-se de que o usuário observado está ciente de que a atividade busca o conhecimento da situação e não a avaliação de seu desempenho. Nesse contexto, a execução da técnica exige que o observador seja habilidoso no trato com o usuário, de forma a não obstruí-lo durante a realização de suas tarefas (CYBIS, BETIOL e FAUST, 2007).

e. Estudo de Documentação

Para Preece, Rogers e Sharp (2005), o estudo da documentação existente é uma boa fonte de dados sobre os passos envolvidos em uma atividade e sobre as regulamentações que governam determinadas tarefas. A documentação é normalmente composta por manuais com procedimentos e regras, documentos do trato diário de tarefas, formulários, fichas entre outros.

A vantagem desta técnica é a de não comprometer o tempo do usuário, como as entrevistas e os questionários, e é muito útil para conhecer as legislações ou as normas internas a fim de se obter informações sobre o embasamento do trabalho.

2.7.3.2 Escolha da Técnica e Análise de Dados

Para Preece, Rogers e Sharp (2005), a escolha de uma técnica para a coleta de dados em uma atividade de identificação de requisitos, deverá considerar a natureza da técnica, o conhecimento de quem analisará os dados, a natureza da tarefa a ser estudada, a disponibilidade dos usuários e de outras fontes e o tipo de informação que se precisa.

Outros fatores que pesam nessa escolha estão relacionados com: as vantagens e desvantagens de cada técnica; do tipo de informação desejada, que depende do ponto em que se está no ciclo de interações do projeto; dos recursos disponíveis; da localização e a acessibilidade aos usuários e o tempo disponível para execução da coleta.

O Quadro 4 apresenta as técnicas de coleta de dados estudadas com algumas informações adicionais, quanto a finalidade, o tipo de dados coletados as vantagens e desvantagens.

Quadro 4 - Técnicas de coleta de dados

Técnica	Boa para	Tipo de dados	Vantagens	Desvantagens
Questionários	Responde as questões específicas	Qualitativos e quantitativos	Pode atingir várias pessoas com poucos recursos.	A formatação das questões é crucial. Baixo índice de respostas. Não obter o que se deseja.
Entrevistas	Explorar questões	Alguns dados quantitativos, mas mais qualitativos	O entrevistador pode guiar o entrevistado. Encoraja o contato entre desenvolvedores e usuários.	Requer tempo. Ambientes artificiais podem intimidar o entrevistado.
Grupos de foco	Coletar vários pontos de vista	Alguns dados quantitativos, mas mais qualitativos	Ressalta áreas de consenso e conflito. Encoraja o contato entre desenvolvedores e usuários.	Possibilidade de dominarem certos tipos de personalidade.
Observação natural	Entender o contexto da atividade do usuário	Qualitativo	A observação do trabalho real oferece percepções que outras técnicas não podem oferecer	Requer muito tempo. Grande quantidade de dados.
Estudo de documentação	Aprender sobre procedimentos, regulamentações e padrões	Quantitativo	Não compromete o tempo dos usuários	O trabalho diário será diferente do documentado

Fonte: Adaptado de Preece, Rogers e Sharp (2005, p. 235)

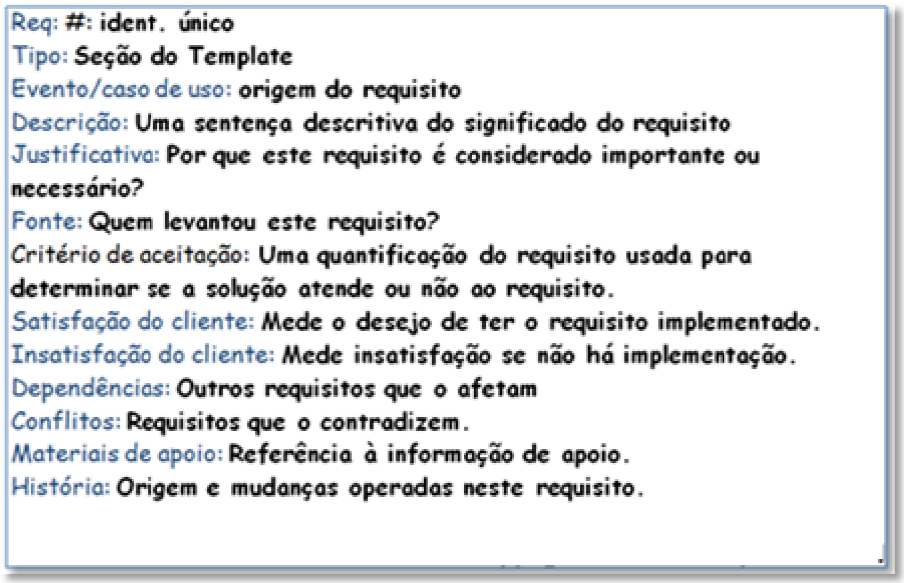
A escolha das técnicas de coleta de requisitos não é exclusiva, podendo ser adotada uma ou um conjunto delas. A organização da atividade de coleta e a utilização das técnicas mais adequadas favorecerem a interpretação e a análise dos dados, que é a etapa seguinte do processo.

A interpretação e a análise de dados constituem-se em um método que busca a integração e a organização dos resultados obtidos pelas técnicas de entrevistas, questionários, grupos de foco, observação e estudo de documentação (CYBIS, BETIOL e FAUST, 2007).

Preece, Rogers e Sharp (2005) sugerem que a atividade de interpretação e análise deve iniciar-se assim que a primeira coleta de dados for conduzida, pois dessa forma, a experiência da coleta estará bem presente na mente dos participantes e isso ajudará na estruturação e no registro dos requisitos.

Para o registro dos requisitos, é sugerido o uso de fichas (Quadro 5), pois elas auxiliam na condução da interpretação e análise dos dados, informam quem levantou o requisito e em que local pode-se encontrar mais informações sobre ele (ROBERTSON e ROBERTSON, 1999).

Quadro 5 - Ficha para registro de requisitos



Fonte: http://www.systemsguild.com/GuildSite/Robs/Template.html, acesso em 16 de maio de 2009.

Cybis, Betiol e Faust (2007) ressaltam que os requisitos devem ser analisados segundo as perspectivas que distinguem a lógica de funcionamento da lógica de operação do sistema.

A lógica de funcionamento compreende os requisitos relacionados com os componentes internos do sistema e suas inter-relações. Como exemplos de componentes internos poderiam ser citados os algoritmos, as bases de dados e os protocolos.

A lógica de operação compreende o conjunto de operações que o usuário realizará com o sistema, a partir da interpretação de elementos na interface, como os botões, as caixas de diálogo, os menus e os seletores. Para a compreensão da lógica de operação, que envolve o usuário e sua interação com o sistema, o analista deve se basear nas metas e objetivos dos usuários, nos objetos que manipulam, nas regras ou restrições que são impostas e nas tarefas que executam.

A Engenharia de Software normalmente classifica os requisitos em duas

categorias mais amplas: os requisitos funcionais, que dizem respeito ao que o sistema deve fazer e os requisitos não-funcionais, que indicam as limitações no sistema e no seu desenvolvimento, como por exemplo, “o sistema deve ser desenvolvido para operar em ambiente Web”, “o sistema deve estar pronto em oito meses” (PREECE, ROGERS e SHARP, 2005).

A atividade descrita ao longo desta etapa teve como objetivo conhecer as necessidades dos usuários e o tipo de suporte o sistema poderia oferecer de maneira útil. Para possibilitar este conhecimento foram estudadas as técnicas utilizadas para a coleta, registro e análise dos requisitos que vão sustentar o *design* e o desenvolvimento subseqüentes. Para uma investigação mais pontual do suporte desejado ao produto a ser desenvolvido, a próxima seção apresenta a etapa de observação de *designs* existentes.

2.7.3.3 Busca de *Designs* Alternativos

Esta etapa compreende a busca de alternativas para o processo de concepção de interfaces do usuário, a partir da observação de outros *designs* semelhantes. Para Preece, Rogers e Sharp (2005), a consideração dos *designs* alternativos é uma atividade valiosa dentro de qualquer processo de *design* e as fontes alternativas podem estar muito próximas do *design* desejado, tais como os concorrentes, versões anteriores de sistemas similares ou ainda algo completamente diferente.

O processo de desenvolver a partir de outras aplicações parte do pressuposto de que muito pouco neste mundo é completamente novo e que normalmente, as inovações surgem a partir de idéias de diferentes aplicações, da evolução de um produto a partir da observação de outros ou da simples cópia de produtos semelhantes (PREECE, ROGERS e SHARP, 2005).

Nielsen (1994) sugere nesta etapa a análise competitiva, que consiste em buscar e utilizar produtos de terceiros que tenham funcionalidades semelhantes às do software em desenvolvimento para realizar testes empíricos. Dessa forma pode-se analisar como o usuário age ao interagir com um sistema real na realização de tarefas com as quais ele se deparará durante a utilização do sistema.

Nielsen ressalta, ainda, que a análise competitiva não implica em apropriar-se de interfaces cujo direito autoral pertence a outros desenvolvedores, mas sim de projetar sistemas melhores do que os analisados, como um resultado da análise de seus pontos fortes e fracos (NIELSEN, 1994).

A escolha entre as alternativas de *design* depende, basicamente, de dois fatores: o primeiro, da observação da interação e da experiência dos usuários com as alternativas, as preferências e sugestões para melhoria e o segundo fator de decisão é apoiado por questões de qualidade, levantadas a partir de avaliações de usabilidade (PREECE, ROGERS e SHARP, 2005).

As fases de identificação das necessidades, identificação dos requisitos e de análise de *designs* alternativos, compreendem atividades investigativas e fundamentais para a construção de *designs* conceituais e versões interativas dos *designs*.

2.7.3.4 Construção de Versões Interativas dos *Designs*

Segundo Preece, Rogers e Sharp (2005), existem dois tipos de *designs*: o conceitual, que se preocupa em transformar os requisitos e as necessidades do usuário em um modelo conceitual e o físico, cuja preocupação está em detalhes do *design*, tais como a tela, as estruturas dos menus, os ícones e os gráficos.

A preocupação do *design* conceitual é transformar os requisitos e as necessidades dos usuários em um modelo conceitual. Preece, Rogers e Sharp (2005, p. 268) definem o modelo conceitual como:

> "...uma descrição do sistema proposto - no que diz respeito a um conjunto de idéias integradas e de conceitos sobre o que ele deveria fazer, como se comportar e com o que se parecer - que seria compreensível pelos usuários da maneira pretendida".

Para o desenvolvimento de um modelo conceitual é preciso visualizar o sistema proposto, baseando-se nas necessidades dos usuários e nos requisitos identificados. A verificação se o modelo será entendido da forma pretendida é realizada a partir de testes interativos ainda na fase de desenvolvimento do produto (PREECE, ROGERS e SHARP, 2005).

Segundo Mayhew (1999), o *design* do modelo conceitual define um processo coerente que possibilita a unificação da identificação e análise dos dados coletados com todas as decisões de detalhes do *design*, sendo o primeiro passo real na concepção da interface do usuário.

Preece, Rogers e Sharp (2005) alertam para a dificuldade da atividade de transformar um conjunto de requisitos em um modelo conceitual “melhor” ou mesmo um modelo “bom o suficiente” e sugerem que uma das melhores maneiras de se proceder é mergulhar nos dados e tentar criar uma empatia com os usuários.

Após a coleta e o estabelecimento do conjunto de requisitos, iniciam-se as atividades de *design*, que devem evoluir de forma iterativa, em ciclos de *design*-avaliação-*redesign* (Figura 16, p. 61) envolvendo os usuários. Nos primeiros estágios do desenvolvimento, os protótipos são as primeiras versões interativas do sistema e podem ser feitas de papel e cartolina; com o progresso dos *designs* e o detalhamento das idéias, os protótipos vão se tornando partes do software, pois passam a se parecer com o produto final (PREECE, ROGERS e SHARP, 2005).

Os protótipos são artefatos que possibilitam visualizações do futuro sistema e podem se constituir em um esboço de uma tela ou um conjunto de telas desenhado em uma folha de papel, num conjunto de imagens de telas em um vídeo, enfim, qualquer representação que possibilite aos usuários, desenvolvedores e interessados interagirem com o produto desejado (PREECE, ROGERS e SHARP, 2005).

A utilização de protótipos é muito útil nas discussões entre os interessados no sistema, facilita a comunicação pela demonstração de idéias e são eficazes para o teste de soluções. Preece Rogers e Sharp (2005) citam que os protótipos esclarecem requisitos vagos, possibilitam a realização de testes com usuários e verificam se o *design* é compatível com o restante do sistema.

Além da utilidade e dos benefícios, Nielsen (1994) complementa que o uso de protótipos pode economizar tempo e dinheiro no desenvolvimento de algo, pois a longa experiência em engenharia de software indica que é muito mais barato mudar alguma coisa no início do projeto do que no final.

A prototipação pode ser de baixa ou de alta fidelidade. Os protótipos de baixa fidelidade não se assemelham muito ao produto final, são simples, baratos e de rápida produção. Por serem facilmente modificáveis, oferecem excelente suporte à exploração de *designs* e idéias alternativas e são particularmente indicados nos primeiros estágios do desenvolvimento (PREECE, ROGERS e SHARP, 2005).

Os *storyboards* (Figura 18) ou narrativas gráficas são protótipos de baixa fidelidade e são utilizados para representar as interações entre o usuário e o sistema. A representação é feita por uma seqüência de desenhos de esboços de tela e elementos do contexto de uso (CYBIS, BETIOL e FAUST, 2007).

A seqüência de desenhos pode ser feita em folhas grandes, coladas em uma parede para serem validadas pelos usuários e especialistas com base nos requisitos de usabilidade coletados.

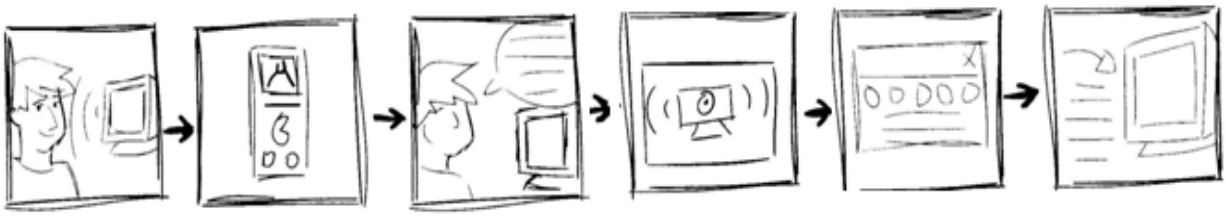


Figura 18 - Exemplo de *Storyboard*
Fonte: Adaptado de Cybis, Betiol e Faust (2007, p. 152)

As maquetes (protótipos de papel) são esboços do sistema, utilizadas para o esclarecimento e desenvolvimento de requisitos específicos da interface. Favorecem a simulação e o teste da interação de maneira bastante rápida, necessitam de mínimos recursos e proporcionam a identificação precoce de problemas de usabilidade (CYBIS, BETIOL e FAUST, 2007).

Preece, Rogers e Sharp (2005) atentam para que as pessoas não se sintam inibidas em razão da qualidade de seus desenhos, e que podem usar bonequinhos de palito, quadradinhos para ícones, procurando desenvolver seus próprios símbolos.

Segundo Cybis, Betiol e Faust (2007), a construção de maquetes é organizada em quatro etapas:

- Definição do conceito: o objetivo dessa etapa inicial é transformar

requisitos ou especificações do sistema em modelos conceituais de interface. A partir de uma reunião, buscando a geração de idéias, são definidas as telas com os componentes essenciais e o mapa de navegação com o fluxo principal do sistema.

- Projeto da interação: em reunião com usuários e projetistas, definem-se os nomes de cada tela sugerida, criam-se cartões com os nomes de cada tela. Os cartões são dispostos em uma parede, o grupo de usuários e projetistas verifica a seqüência em que as telas são acessadas durante a realização da tarefa. A seqüência pode ser reorganizada, cartões de telas podem ser suprimidos ou adicionados.
- Projeto e teste das telas: nesta etapa, as telas identificadas na etapa anterior são criadas e para o teste, o projetista deve organizar uma reunião com os usuários. Para o teste, as maquetes deverão ser coladas na parede, dispostas na mesma seqüência que foi verificada na etapa anterior, os usuários deverão interagir com as telas de modo a simular a tarefa. A cada interação do usuário, o projetista explicará a reação da interface e indicará a próxima tela, se for o caso.

Os protótipos de alta fidelidade utilizam materiais que são esperados no produto final e por essa razão, estes se parecem muito com o produto já acabado (PREECE, ROGERS e SHARP, 2005).

Protótipos de alta fidelidade oferecem componentes de interface, aparência e comportamento bem parecidos com o futuro sistema, pois são desenvolvidos por meio da utilização de ferramentas de software. O conteúdo de informação é normalmente mais elaborado e possibilita a obtenção de medidas de usabilidade (CYBIS, BETIOL e FAUST, 2007).

Cybis, Betiol e Faust (2007) alertam para não gastar muito tempo de desenvolvimento na construção desse tipo de protótipo, pois prejudicaria o tempo previsto para desenvolvimento. Para os autores, uma alternativa aos protótipos de alta fidelidade é a construção de versões evolutivas do sistema, construídas na própria plataforma definitiva do futuro sistema.

As versões evolutivas oferecem uma variante positiva que é a distribuição de

versões intermediárias aos usuários que, ao utilizá-las vão se adaptando ao sistema e podem realizar uma validação mais fidedigna.

Ao construir protótipos, o projetista deve sempre ter em mente que há um limite para o número de questões que este artefato pode responder. A intenção é a produção rápida de algo que possa testar algum aspecto do futuro sistema (PREECE, ROGERS e SHARP, 2005).

O Quadro 6 informa as vantagens e desvantagens da utilização dos protótipos de alta fidelidade e dos protótipos de baixa fidelidade.

Quadro 6 - Vantagens e desvantagens dos protótipos de alta e baixa fidelidade

Tipo	Vantagens	Desvantagens
Baixa fidelidade	• Custo mais baixo de desenvolvimento • Avalia múltiplos conceitos de *design* • Instrumento de comunicação útil • Útil para identificação de requisitos de mercado • Demonstra que o conceito funciona	• Verificação limitada de erros • Especificação pobre em detalhes para codificação • "Uso" conduzido pelo facilitador • Utilidade limitada após estabelecimento dos requisitos • Utilidade limitada para testes de usabilidade • Limitações de fluxo e navegação
Alta fidelidade	• Funcionalidade completa • Totalmente interativo • Uso conduzido pelo usuário • Define claramente o esquema de navegação • Uso para exploração e teste • Mesma aparência do sistema final • Serve como uma especificação viva • Ferramenta de venda e marketing	• Desenvolvimento mais caro • Sua criação demanda tempo • Não serve para coleta de requisito

Fonte: Adaptado de Preece, Rogers e Sharp (2005, p. 266)

A atividade de prototipação compreende ciclos de interação com o usuário ou mesmo com especialistas em usabilidade. As avaliações, que são abordadas na próxima seção, podem ser realizadas sobre os protótipos, sejam eles funcionais ou não, possibilitando o gradual refinamento.

2.7.3.5 Execução de Avaliações de Usabilidade

A conscientização quanto às questões de usabilidade ampliou-se muito nos últimos anos devido à presença da Web, porém, existe ainda muita resistência por parte dos projetistas, que insistem em achar que suas próprias impressões sobre a atratividade de uma interface são suficientes (PREECE, ROGERS e SHARP, 2005).

Algumas empresas têm a crença errônea quanto ao estudo e aplicação da usabilidade, por acreditarem que as atividades de avaliações de usabilidade vão retardar seus projetos (NIELSEN e LORANGER, 2007).

As avaliações de interfaces são as atividades pelas quais os projetistas podem se certificar de que o sistema é usável e que está de acordo com o que os usuários desejam (PREECE, ROGERS e SHARP, 2005).

A produção de interfaces de baixa qualidade, sem preocupação com as questões de usabilidade e de avaliações, requer treinamento excessivo dos usuários, desmotiva a exploração dos recursos disponibilizados, confunde e induz os usuários ao erro, gera insatisfações devido às dificuldades de uso, diminui a produtividade e não traz o retorno de investimento esperado (PREECE, ROGERS e SHARP, 2005).

No processo de concepção de interfaces, as avaliações têm um papel fundamental e devem ser executadas durante todo o ciclo de desenvolvimento, a fim de que seus resultados sejam utilizados para a melhoria gradual da interface. Isso significa que as avaliações não constituem uma fase única no desenvolvimento e muito menos como uma atividade a ser executada apenas no final do processo (ROCHA e BARANAUSKAS, 2003).

Considerando que as avaliações podem ser realizadas em qualquer momento do processo de desenvolvimento, ou mesmo ao final do projeto, estas podem ser classificadas em Avaliações Formativas ou Avaliações Somativas (PREECE, ROGERS e SHARP, 2005).

As Avaliações Formativas (ou Construtivas) são realizadas ao longo de todo o processo de *design*, sempre que os projetistas precisarem compreender melhor o

que os usuários desejam e precisam, ou quando precisam verificar se suas idéias atendem as necessidades conhecidas dos usuários. Para este tipo de avaliação são utilizados artefatos como: *storyboards* e protótipos de sistema.

As Avaliações Somativas (ou Conclusivas) são realizadas nas etapas finais de cada ciclo do desenvolvimento ou quando o produto está pronto. Nesta fase são avaliados os protótipos intermediários ou finais da aplicação.

Avaliações formativas e avaliações somativas são classificações baseadas em etapas do projeto, porém, as avaliações também podem ser agrupadas em padrões de avaliação, que podem ser distinguidas conforme a forma em que são realizadas ou quanto às pessoas que realizam a avaliação.

Para Preece, Rogers e Sharp (2005), as avaliações compreendem juízos de valor emitidos por usuários, interessados, especialistas ou desenvolvedores, baseados em crenças e expectativas originadas da teoria e/ou de experiências empíricas. As crenças e as expectativas, associadas aos métodos ou técnicas de avaliação, aplicam-se conforme os seguintes padrões de avaliação: Avaliação Rápida, Testes de Usabilidade, Estudos de Campo e Avaliação Preditiva.

a. Avaliações Rápidas

Avaliações rápidas constituem-se em uma prática muito comum durante a concepção de interfaces. São realizadas por meio de reuniões informais entre usuários e desenvolvedores, com o objetivo de se obter apreciações sobre as interfaces e confirmar se as idéias dos desenvolvedores vão ao encontro das necessidades dos usuários.

As avaliações rápidas são do tipo formativas, literalmente rápidas, realizadas a qualquer momento em que se deseje um *feedback* sobre um *design*, um novo ícone ou um novo recurso implementado na interface, e deve ser repetido em vários pontos do ciclo de desenvolvimento.

Os dados coletados das avaliações rápidas são geralmente descrições informais, que são canalizadas para o processo de *design* no formato de desenhos, relatos ou bilhetes.

b. Testes de Usabilidade

Os testes de usabilidade pressupõem o envolvimento de usuários finais ou representativos do público-alvo, realizando tarefas específicas em um contexto, real ou simulado, pelas quais, busca-se constatar a existência de problemas, os impactos negativos e identificar suas causas na interface (CYBIS, BETIOL e FAUST, 2007).

A realização de testes com usuários gera um amplo conjunto de dados que podem ser capturados por meio de recursos de monitoração, como por exemplo: vídeo do usuário realizando a tarefa, arquivos de *log* da interação na interface (por exemplo, movimentos de *mouse*, cursores ou teclas apertadas), áudio de comentários ou interjeições do usuário, e até mesmo certos sinais sensório-motores, como por exemplo, a direção do olhar ou a tensão muscular.

Os dados coletados, após análise, podem ser utilizados para prever e explicar certas ocorrências de desempenho, bem como para ajudar a corrigir e prevenir erros de interação.

Outras técnicas de avaliação englobadas por este padrão referem-se às entrevistas e questionários, pelas quais, pode-se descobrir o grau de satisfação dos participantes, bem como outros aspectos psicológicos.

c. Estudos de Campo

O estudo de campo caracteriza-se por englobar técnicas de avaliação que são realizadas em ambiente “natural” dos usuários, a fim de se buscar um maior entendimento do que os usuários realmente executam.

Observar como as pessoas utilizam e se relacionam com artefatos tecnológicos, sem interferir ou propor tarefas, possibilita ampliar o conhecimento sobre suas atividades e a influência da tecnologia sobre elas.

Estudos de campo são indicados para: prospectar a introdução de novas tecnologias; determinar requisitos para o *design*; decidir sobre estratégias de promoção e adoção de tecnologias; e, para descobrir como uma tecnologia é de fato utilizada.

d. Avaliação Preditiva

As avaliações preditivas são realizadas por especialistas que aplicam seus conhecimentos acerca dos usuários e de situações típicas de uso para prever problemas de usabilidade.

Para realização da avaliação, os especialistas são geralmente guiados por listas de heurísticas e não é necessária a presença de usuários, razão pela qual o método é considerado relativamente barato, rápido e, conseqüentemente, atrativo para as empresas, apesar de suas limitações.

As avaliações preditivas devem ocorrer a qualquer momento do projeto e podem ser avaliados protótipos, versões finais de interfaces ou modelos de aspectos específicos de uma interface.

Os dados gerados são consolidados em uma lista com os problemas observados, que podem ser quantificados quanto ao grau de severidade do problema, adicionada de sugestões para aplicação no redesenho da interface.

A filosofia das avaliações preditivas baseia-se na utilização das heurísticas e na experiência dos avaliadores e dos desenvolvedores que sustentam as revisões sugeridas pelos avaliadores.

O estudo dos padrões de avaliação contribui para informar aos desenvolvedores as características gerais das avaliações, os tipos de dados que podem ser coletados e principalmente, o padrão de avaliação mais adequado para uma determinada fase do desenvolvimento. Porém, a execução de avaliações não é atividade trivial e é necessário algum tipo de planejamento para que esta não se torne uma perda de tempo e dinheiro. Com o intuito de orientar a equipe de desenvolvimento nas atividades de avaliações, na seção de Planejamento e Execução de Avaliações é apresentado o *framework* DECIDE, que oferece uma lista de itens para guiar e auxiliar os avaliadores na realização de avaliações.

2.7.4 Planejamento e Execução de Avaliações

Existem muitos fatores a serem considerados quando da realização de

avaliações. O planejamento das atividades de avaliação exige dos avaliadores a determinação dos objetivos, das técnicas de avaliação, das questões a serem respondidas, das questões éticas, da avaliação e da interpretação do que foi avaliado.

Com o objetivo de auxiliar os avaliadores, principalmente os menos experientes, no planejamento e na realização das atividades de avaliação é apresentado o *framework* DECIDE, desenvolvido por Preece, Rogers e Sharp (2005) que oferece uma lista de verificação com seis itens, que são descritos nas seções seguintes: Determinar as metas; Explorar as questões; Escolher o paradigma de avaliação e as técnicas; Identificar questões de ordem prática, Decidir como lidar com as questões éticas; e, Avaliar, interpretar e apresentar os dados (PREECE, ROGERS e SHARP, 2005).

2.7.4.1 Determinar as metas

A determinação das metas deve ser o primeiro passo no planejamento de execução de uma avaliação e pode ser obtida respondendo-se as seguintes questões: “Quais são as metas de alto nível da avaliação?” e “Quem as quer e por que?”.

As metas guiam e influenciam na escolha do padrão de avaliação a ser utilizado, como por exemplo: “os usuários podem completar uma determinada tarefa dentro de certo período de tempo”, “encontre um determinado item” ou “encontre a resposta para uma determinada questão”, indicam a utilização do padrão de testes de usabilidade (testes com usuários).

Outros exemplos de metas de avaliação poderiam ser: “Assegurar-se de que a interface final é consistente”, “Identificar como a interface de um produto existente poderia ter sua interface melhorada” e “Avaliar a usabilidade de um sistema de carrinho de compras”.

2.7.4.2 Explorar as questões

As metas definidas no primeiro passo do planejamento podem ter um caráter muito amplo e neste caso, devem ser operacionalizadas a fim de se identificar

questões cujas respostas satisfaçam a meta global.

A meta “Avaliar a usabilidade de um sistema de carrinho de compras” é muito ampla e, neste caso, é preciso identificar questões mais específicas e exclusivas, que satisfaçam a meta global, como por exemplo: “Os usuários conseguem encontrar os produtos que desejam comprar com facilidade e rapidez?”, “Os usuários conseguem se cadastrar com facilidade e rapidez?” e “As etapas do processo de compra são intuitivas e fáceis de serem executadas?”.

O resultado positivo da atividade de exploração das questões é realização de avaliações eficazes, pois são guiadas por objetivos bem definidos e questões claras.

2.7.4.3 Escolher o padrão de avaliação e as técnicas

Após a definição das metas e das perguntas a serem respondidas pela avaliação o passo seguinte é a escolha do padrão e da técnica de avaliação. O padrão de avaliação determina os tipos de técnica que podem ser utilizada.

Para o padrão de avaliações “Rápidas”, por serem rápidas, informais, e para atendimento de questões pontuais, podem ser aplicadas diretamente entre o projetista e o usuário, sem a necessidade de um plano de avaliação.

Questões exploradas no passo anterior, como: “Os usuários conseguem encontrar os produtos que desejam comprar com facilidade e rapidez?” talvez possam indicar o padrão de testes de usabilidade e a utilização de técnica de testes com usuários, questionários ou entrevistas.

2.7.4.4 Identificar questões práticas

As questões relativas aos usuários, avaliadores, equipamentos, cronogramas e orçamentos devem ser consideradas antes do início das atividades de avaliação. A disponibilidade desses recursos define os ajustes necessários e as técnicas de avaliação que serão utilizadas.

Como exemplo, os testes de usabilidade devem conter informações referentes à identificação dos usuários, a faixa etária, o sexo, o grau de experiência e se realmente representam a população de usuários pretendidos.

Para utilização da técnica de questionários é necessário um grande número de participantes, o que conseqüentemente, exige uma forma de identificar os usuários-alvo da pesquisa em quantidade suficiente para uma amostra significativa.

Quantos aos avaliadores é preciso verificar se a equipe de avaliação dispõe de conhecimento técnico para a realização da atividade. A análise de resultados de avaliações com usuários, realizadas a partir de filmagens (vídeos) ou de coletas de informações de operações sobre o sistema, exige conhecimento especializado e, no caso de vídeos, podem ser necessárias muitas horas de avaliação.

Os equipamentos necessários para a avaliação compreendem outra questão que deve ser abordada antes da escolha da técnica de avaliação; decisões sobre utilização e disponibilidade de vídeos, laboratórios, computadores e gravadores também devem ser consideradas.

As questões de cronograma e orçamento são fundamentais e devem sempre estar presente nas decisões de padrões e técnicas de avaliação. Preece, Rogers e Sharp (2005) citam que em ambientes comerciais ou corporativos, em geral não há tempo suficiente para a realização de avaliações da forma que se desejaria fazer, e, nesse caso, a equipe deve planejar e procurar fazer um bom trabalho com o tempo e os recursos disponíveis.

2.7.4.5 Decidir como lidar com as questões éticas

As questões éticas são principalmente indicadas para as avaliações que envolvem pessoas e se constituem em ações que visam garantir sua privacidade, como garantir que os nomes das pessoas não estarão associados aos dados coletados sobre elas ou divulgados em relatórios sem sua permissão.

Para formalização das preocupações éticas na execução de avaliações, muitas instituições e gerentes de projeto utilizam um termo de consentimento, que é um recurso destinado a explicar os objetivos dos testes ou das pesquisas e garantir a confidencialidade dos dados pessoais dos usuários.

Em um Termo de Consentimento devem constar os seguintes itens (Figura 19):

- O objetivo do estudo;
- A confidencialidade das informações e a garantia do anonimato no caso de uso de vídeo ou áudio;
- Aviso ao participante de que ele é livre para abandonar a avaliação no momento em que não se sentir confortável com o procedimento.

Termo de Consentimento

Afirmo que sou maior de 18 anos e desejo participar do programa de pesquisa que está sendo conduzida pela equipe de Desenvolvimento de Sistemas do 3º CTA.

O propósito da pesquisa é avaliar a usabilidade do “Sistema x”, desenvolvido no 3º CTA para disponibilizar informações ao público em geral. Também responderei a questões abertas sobre o “Sistema x” e sobre a experiência de uso.

Todas as informações coletadas neste estudo são confidenciais e meu nome não será identificado em momento algum.

Estou ciente de que posso fazer perguntas ou desistir da colaboração em qualquer momento, sem qualquer tipo de penalidade.

Assinatura do participante e data

Figura 19 - Termo de Consentimento
Fonte: Adaptado de Preece, Rogers e Sharp (2005, p. 372)

2.7.4.6 Avaliar, interpretar e apresentar dados.

A última etapa do *framework* DECIDE compreende atividades para verificar quais dados serão coletados, como serão analisados e de que forma essas descobertas serão passadas para a equipe de desenvolvimento.

Normalmente, os padrões de avaliação e as técnicas determinam o tipo de dado que será coletado, mas ainda é necessário definir como estes dados serão tratados pelos avaliadores. Nos testes com usuários, por exemplo, os dados devem ser tratados estatisticamente, pois os resultados de medidas de desempenho (por exemplo, tempo de realização de tarefas, erros etc.), podem ser contabilizados e apresentados em medidas de máximo, mínimo e médio do grupo estudado.

A coleta dos dados ainda deve ser verificada quanto à confiabilidade, se realmente quantifica o que se deseja medir, aos possíveis desvios decorrentes de distorções e quanto à validade do ambiente em que foi realizado.

Diferentes técnicas de avaliação de dados possibilitam diferentes graus de confiabilidade, como exemplo, uma entrevista informal, não-estruturada, com um usuário é de baixa confiabilidade, pois é muito difícil a repetição da mesma discussão, por outro lado, uma avaliação realizada em laboratório controlado possui um grau maior de confiabilidade.

O *framework* DECIDE é uma ferramenta elaborada para guiar os avaliadores na estruturação e no planejamento de qualquer tipo de avaliação. Uma das atividades do DECIDE é a escolha do padrão de avaliação e da técnica de avaliação a ser utilizada.

Existem muitas técnicas de avaliação e para o presente trabalho é apresentada Avaliação Heurística proposta por Nielsen (1994), que é realizada por avaliadores com base em princípios de usabilidade. A técnica, já utilizada pelo 3º CTA com o objetivo de avaliar as interfaces de um sistema desenvolvido pela Organização Militar, mostrou-se eficiente ao detectar problemas de interface, possibilitando as correções necessárias antes da efetiva distribuição do sistema (WATANABE e DUDUCHI, 2008).

2.7.5 Avaliação Heurística

Segundo Nielsen (1994), a avaliação heurística é um método de inspeção sistemático, realizada por um grupo de avaliadores, com base em princípios de usabilidade, também conhecidos como heurísticas, com o objetivo de encontrar problemas de usabilidade na interface a fim de serem analisados e corrigidos ao longo de um processo de *design*.

Apesar de, em princípio, poder ser realizada por apenas um avaliador, a experiência em muitos projetos mostra que uma única pessoa não é capaz de encontrar todos os problemas de usabilidade da interface. Dessa forma, a utilização de múltiplos avaliadores melhora significativamente o resultado da avaliação, pois diferentes pessoas encontram problemas diferentes (NIELSEN, 1994).

Para Nielsen (1994), as equipes devem ser constituídas de um mínimo de três até cinco avaliadores, pois como mostrado na Figura 20, a utilização de três

avaliadores em vez de apenas um, representa um ganho considerável no percentual de problemas encontrados e cinco avaliadores, segundo o autor, parece ser um número razoável em vista do custo-benefício.

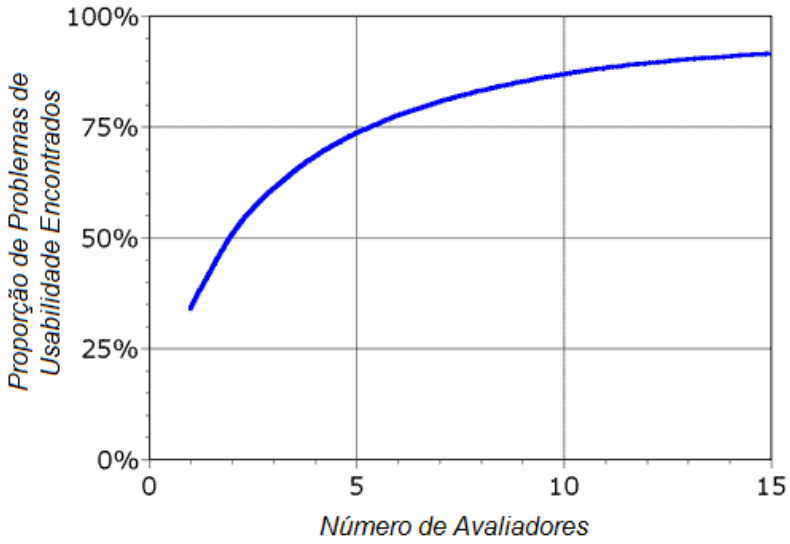


Figura 20 - Problemas de usabilidade em função do número de avaliadores.
Fonte: Adaptado de Nielsen (1994, p. 156)

Nielsen (1994) acrescenta que apesar da análise de custo-benefício, a definição do número de avaliadores pode ser maior que cinco, caso o sistema seja crítico, extensivo ou relacionado a tarefas críticas.

Apesar da avaliação heurística poder ser realizada por pessoas com pouca ou nenhuma experiência em usabilidade, é recomendável o uso de especialistas, pois se obtém um desempenho melhor e com um número menor de avaliadores. O resultado da avaliação poderá ser melhor ainda, caso os avaliadores sejam especialistas e, além disso, tenham domínio da área de atuação do sistema (DIAS, 2007).

Segundo Cybis, Betil e Faust (2007, p. 184), além da competência e do domínio dos avaliadores, a rapidez da avaliação, a quantidade e a importância dos problemas de usabilidade também dependem das estratégias empregadas pelos avaliadores. Para os autores:

> Em função da natureza subjetiva do conhecimento dos avaliadores e da variabilidade de estratégias adotadas nas avaliações, os resultados das avaliações heurísticas apresentam grandes diferenças de avaliador para avaliador.

Pollier (1993) *apud* Cybis, Beriol e Faust (2007) registrou as estratégias que os especialistas em ergonomia de software utilizam durante a realização de avaliações e as descreveu em cinco abordagens diferentes:

- **Objetivos dos usuários** - avaliação das principais tarefas e subtarefas para os usuários e para as empresas.
- **Estrutura de interface** - avaliação das estruturas de menus em níveis hierárquicos.
- **Níveis de abstração** - avaliação segundo um modelo lingüístico estruturado em níveis de abstração: semântica (significado das ações realizadas com a interface), sintática (seqüências de ações de comandos e das apresentações nas telas), lexical (significado das unidades de apresentação) e físico (características perceptíveis das apresentações e dos dispositivos de entrada).
- **Objetos da interface** - avaliação a partir da estrutura dos objetos de interação (janelas, formulários, botões, menus etc.).
- **Qualidades esperadas da interface** - avaliação com base em um conjunto de qualidades, princípios, heurística ou critérios de usabilidade esperados.

Dias (2007) sugere que além dos especialistas e das equipes de desenvolvimento, as avaliações heurísticas também sejam realizadas por representantes dos usuários.

Em geral, um avaliador demora em média de uma a duas horas para realizar a avaliação. Sessões de avaliação muito longas devem ser divididas em sessões menores, com foco em determinadas partes do sistema (NIELSEN, 1994).

Para obtenção de resultados independentes e sem influências, os avaliadores devem inspecionar todos os elementos das interfaces do sistema individualmente, comparando-os com os princípios ou as heurísticas escolhidas (Quadro 7).

Quadro 7 - Lista de Heurísticas de Usabilidade

Heurísticas
1. Visibilidade do *status* do sistema - O sistema deve 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 - O sistema precisa falar a linguagem do usuário, com palavras, frases e conceitos familiares, 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 - Os usuários freqüentemente escolhem por engano funções do sistema e precisam ter como sair do estado indesejado sem ter que percorrer um extenso diálogo. O usuário deve ser capaz de desfazer, interromper ou cancelar uma ação quando desejar.
4. Consistência e padrões - Os usuários não precisam adivinhar que diferentes palavras, situações ou ações significam a mesma coisa. O sistema deve seguir as convenções de plataforma computacional.
5. Prevenção de erros - O sistema deve evitar que o erro aconteça, informando o usuário sobre as conseqüências de suas ações ou, se possível, impedindo as ações que levariam a uma situação de erro.
6. Reconhecimento em vez de relembrança - O sistema deve tornar objetos, ações e opções visíveis. O usuário não deve ter que lembrar de informações de uma para outra parte do diálogo. As instruções para uso do sistema devem estar visíveis e facilmente recuperáveis quando necessárias.
7. Flexibilidade e eficiência de uso - Os usuários novatos se tornam peritos com o uso. O sistema deve prover aceleradores de forma a aumentar a velocidade da interação, permitindo que usuários experientes possam "cortar caminho" em ações freqüentes.
8. Estética e *design* minimalista - Os diálogos não devem conter informações irrelevantes ou raramente necessárias. 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 - As 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.

Fonte: Adaptado de Nielsen (1994)

Outra característica adicional à avaliação heurística é a possibilidade de se estimar o grau de severidade de cada problema detectado (Quadro 8). Essa estimativa é muito útil no momento da alocação dos recursos, pois dessa forma, podem ser priorizados os problemas mais graves e os demais, podem ser deixados para uma nova versão (NIELSEN, 1994).

Quadro 8 - Graus de severidade dos problemas encontrados

Severidade	Descrição
1	Não 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 - corrigi-lo deve ter prioridade baixa.
4	Problema de usabilidade grave - importante corrigi-lo, deve ser dada alta prioridade.
5	Catástrofe de usabilidade - a sua correção é imperativa antes do produto ser liberado.

Fonte: Adaptado de Nielsen (1994, p. 103)

Após a realização das avaliações e de posse do relatório com os resultados das inspeções individuais, o grupo de avaliadores e a equipe de desenvolvimento, podem discutir os graus de severidade e as sugestões de *redesign* (NIELSEN, 1994).

Quanto às questões de tempo, complexidade, custo e eficácia de aplicação, Nielsen (1994, p. 160, tradução nossa) apresenta que: “A avaliação heurística foi explicitamente desenvolvida como um método de *engenharia de usabilidade com desconto* [..]”. A sua aplicação não garante resultados perfeitos ou a identificação de todos ou qualquer problema de interface, mas é considerado um método rápido, fácil e barato em comparação com os tradicionais testes com usuários.

Neste capítulo foram apresentados os conceitos da usabilidade, as interfaces e interações, os princípios, diretrizes e avaliações de usabilidade e, por fim, as principais atividades da engenharia de usabilidade necessárias à concepção de aplicações Web. Para a análise e construção do método proposto pelo presente trabalho, nos próximos capítulos serão conceituados e apresentados de forma breve alguns modelos de processos de desenvolvimento de aplicações Web.

3. Processos de Desenvolvimento de Aplicações Web

Um processo de desenvolvimento de software constitui-se de um conjunto de documentos que definem o fluxo de trabalho, as atividades, os artefatos e as funções dos envolvidos no processo. Como principais funções, um processo de desenvolvimento de software deve (CONALLEN, 2000):

- Guiar o time de desenvolvimento quanto à ordem das atividades;
- Especificar quais os artefatos devem ser desenvolvidos;
- Dirigir as tarefas dos desenvolvedores de forma individual e o time de desenvolvimento como um todo; e,
- Oferecer critérios de monitoração do projeto e das atividades.

No caso das aplicações Web, o processo de desenvolvimento pode aproveitar os princípios, os conceitos e os métodos da engenharia de software, porém, as características específicas desse tipo de software exigem abordagens diferentes, novas metodologias e ferramentas para seu desenvolvimento, implantação e avaliação (PRESSMAN, 2006).

Gonçalves *et al.* (2007) corroboram com Pressman ao afirmar que os trabalhos publicados sobre o desenvolvimento de aplicações Web encontram-se fortemente baseados nos paradigmas da Engenharia de Software, mas, que apesar dessa forte ligação, a engenharia de aplicações Web apresenta novas preocupações, tais como a multidisciplinaridade, a abordagem de aspectos estéticos, funcionais e de usabilidade.

A diferente abordagem das aplicações Web quanto ao aproveitamento dos processos de desenvolvimento da Engenharia de Software está diretamente relacionada com as características específica desses tipos de aplicações, tais como seus componentes, arquitetura, aspectos de segurança, disponibilização e concorrência de acessos que serão apresentadas na seção seguinte.

3.1 Aplicações Web

As aplicações Web podem ser definidas como sistemas de software que

utilizam a Web como ambiente de execução (WINCKLER e PIMENTA, 2001).

Para Conallen (2000), aplicações Web englobam *sites* Web e sistemas Web. Os *sites* Web compreendem a forma original de sistemas hipermídia distribuídos, que são compostos por documentos, imagens, sons, vídeos, com o propósito de permitir a pesquisa e o acesso a esses elementos e informações, publicados nos vários outros computadores que formam a Internet.

A arquitetura dos *sites* Web é composta de alguns componentes, tais como o servidor Web, a conexão de rede e um *browser* (navegador de internet) do cliente (Figura 21).

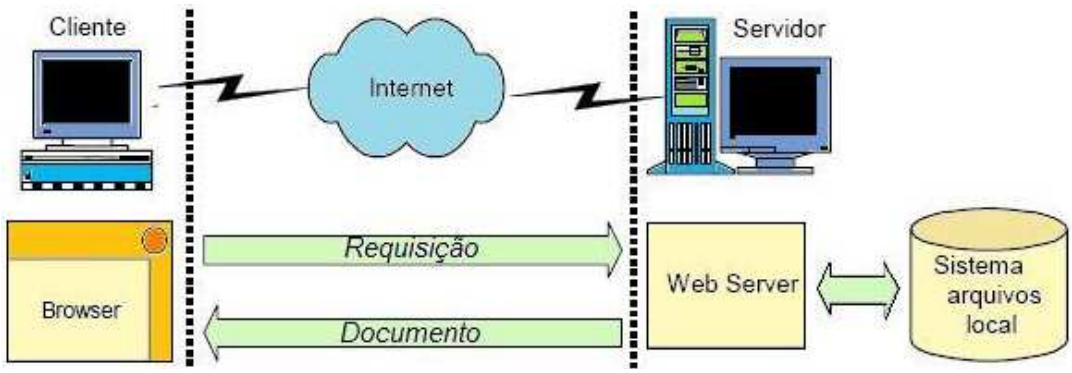


Figura 21 - Arquitetura de um *Site* Web
Fonte: Adaptado de Conallen (2000, p. 10)

Por outro lado, as aplicações Web ampliam o conceito de *sites* Web no momento em que são adicionadas funcionalidades que permitem aos usuários executarem lógicas de negócio a partir de um *browser*. Diferentemente dos *sites* Web, nos quais a busca constitui-se de documentos pré-formatados, nas aplicações Web, o conteúdo é construído dinamicamente, em função da interação dos usuários com as páginas Web (CONALLEN, 2000).

A arquitetura das aplicações Web inclui um servidor de aplicações, que é responsável pela execução da lógica de negócios, além dos componentes básicos de um *site* Web. Nas aplicações Web, é possível, ainda, adicionar um repositório de dados (banco de dados) ao servidor de aplicações (Figura 22).

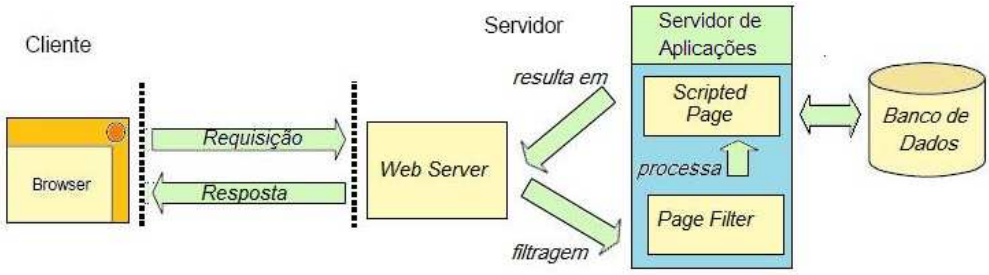


Figura 22 - Arquitetura de uma Aplicação Web
Fonte: Adaptado de Martins (2003, p. 26)

As aplicações Web são intensamente voltadas para redes, guiadas por conteúdo e evoluem continuamente. Podem atender uma comunidade diversificada de usuários, onde a concorrência, que se constitui nos acessos simultâneos, pode ser imprevisível. Independentemente da concorrência, o desempenho no atendimento das requisições do usuário e a disponibilidade da aplicação, devem ser mantidos, pois os usuários podem abandonar a aplicação e irem para o concorrente (PRESSMAN, 2006).

Quanto ao processo de desenvolvimento, as aplicações Web freqüentemente exigem um curto prazo para o projeto, construção e colocação no mercado. Após a disponibilização da aplicação, o processo de evolução é rápido e constante, diferentemente dos demais softwares de aplicação convencional que evoluem ao longo de uma série de versões programadas.

A forma de disponibilização das aplicações Web torna difícil, senão impossível a limitação da população final que pode ter acesso ao sistema. A proteção do seu conteúdo reservado e da transmissão de dados exigem a implementação de fortes medidas de segurança na aplicação e em toda a infra-estrutura que a apóia.

No contexto de ambiente Web (sejam *sites* ou aplicações Web) a facilidade de como o usuário navega e interage com esses sistemas é possibilitada, entre outras atividades, pela adoção e atendimento das diretrizes e dos critérios de usabilidade, que atualmente, tornaram-se uma preocupação evidente para as empresas, para os desenvolvedores e também para os usuários, que passaram a ser mais exigentes e a não tolerar sistemas difíceis de usar.

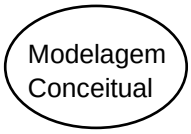
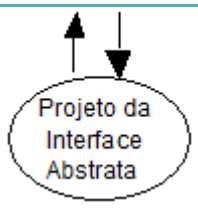
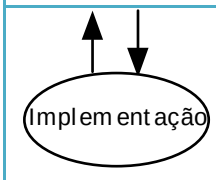
A identificação das características dos processos de desenvolvimento de

software e da abordagem das aplicações Web que diferem das aplicações convencionais (não Web), constituem-se na fundamentação necessária para o estudo de métodos e processos de desenvolvimento de aplicações Web que serão apresentados nas seções seguintes.

3.2 Método de Projeto de Hipermídia Orientado a Objetos (OOHDM)

O método de projeto de hipermídia orientado a objetos (*Object-Oriented Hypermedia Design Method* - OOHDM) foi proposto por Schwabe e Rossi (1998), e é composto de quatro diferentes etapas, por meio das quais o modelo é construído ou enriquecido: o projeto conceitual, o projeto navegacional, o projeto de interfaces abstratas e a implementação (Quadro 9).

Quadro 9 - Resumo do método OOHDM

Atividades	Produtos	Mecanismos	Interesses do Projeto
	Classes, subsistemas, relacionamentos, perspectivas de atributos	Classificação, composição, generalização e especialização	Modelagem da semântica do domínio de aplicação
	Nós, elos, estruturas de acesso, contextos de navegação, transformações navegacionais	Mapeamento entre os objetos conceituais e de navegação. Padrões de navegação para a descrição da estrutura geral da aplicação.	Leva em conta o perfil do usuário e a tarefa; ênfase em aspetos cognitivos e arquiteturais.
	Objetos de interface abstrata, reações a eventos externos, transformações de interface	Mapeamento entre objetos de navegação e objetos de interface.	Modelagem de objetos perceptíveis, implementa metáforas escolhidas. Descrição de interface para objetos navegacionais
	Aplicação em execução	Aqueles fornecidos pelo ambiente alvo	Desempenho, completitude

Fonte: Adaptado de Schwabe e Rossi (1998, p. 2)

A modelagem conceitual do OOHDM gera uma representação a partir das classes, relacionamentos e subsistemas que definem o domínio da aplicação,

utilizando-se dos princípios de modelos da orientação a objetos. Nesse processo, são utilizados mecanismos de agregação, generalização e especialização para aumentar o poder de abstração (Figura 23).

Durante este passo, o principal objetivo é capturar a semântica do domínio sem se preocupar com os usuários e as tarefas envolvidas.

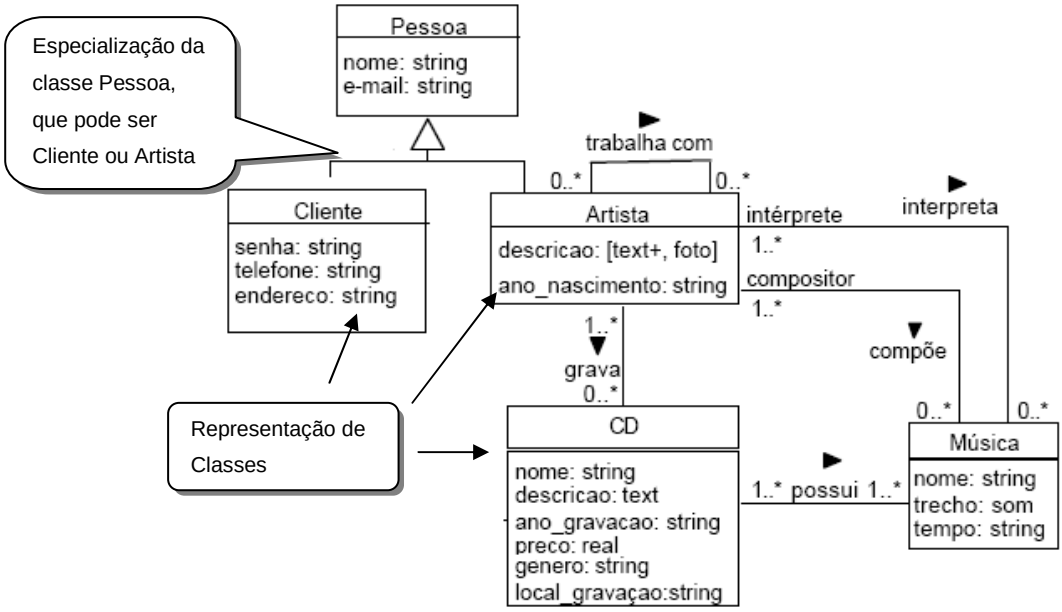


Figura 23 - Modelagem conceitual do método OOHDM
Fonte: o autor

O projeto de navegação do OOHDM é concebido a partir dos objetos e relacionamentos abstraídos no modelo conceitual (passo anterior). Neste momento são definidos quais objetos serão navegados, quais os relacionamentos e quais as estruturas de elo existentes.

Também são definidas as visões navegacionais do esquema conceitual, onde cada visão define um conjunto de contextos e classes de navegação. Os contextos de navegação expressam a estrutura geral da aplicação, enquanto as classes de navegação, como os nós e os elos, especificam os objetos que serão vistos pelo usuário.

O projeto de interfaces abstratas tem como objetivo definir os objetos da interface do usuário. Para representação das características comportamentais da interface e do relacionamento entre os objetos de interface e os objetos de navegação, utiliza-se um modelo formal chamado de visão abstrata de dados

(*Abstract Data View* - ADV).

Um modelo ADV (Figura 24) representa uma metáfora de interface, inclui representação de objetos de navegação da interface (por exemplo: botões, menus e ícones) e a definição do leiaute estático da interface.

Figura 24 - Exemplo de modelo ADV e sua respectiva interface
Fonte: Adaptado de http://vagalume.uol.com.br/la-bouche/, acesso em: 16 de maio de 2009

A implementação, que é a quarta fase do OOHDM, contempla a elaboração das interfaces, que pode ser feita com base na especificação fornecida pelos ADV, desenvolvida na fase anterior. A estrutura dos ADVs oferece uma indicação sobre quais os objetos de interface precisam ser definidos.

3.3 Linguagem de Modelagem *Web* (WebML)

A Linguagem de Modelagem Web (*Web Modeling Language* - *WebML*) é um processo de modelagem para aplicações *Web* proposto por Ceri *et al.* (2003), que permite que os desenvolvedores modelem as funcionalidades de um *site* em um alto nível de abstração, sem se comprometerem com detalhes de alguma arquitetura específica.

A WebML é atualmente suportada por uma ferramenta de software denominada WebRatio, disponível na Internet (http://www.webratio.com), com licença livre para uso não comercial. Segundo as especificações do próprio fabricante, a ferramenta gera aplicações completas a partir dos diagramas especificados na WebML.

A especificação de um *site* em WebML consiste de quatro perspectivas:

O Modelo Estrutural (*Structural Model*) expressa a organização conceitual dos dados do *site*, ou seja, suas entidades e relacionamentos, compatível com notações clássicas como diagramas de Entidades e Relacionamentos ou diagrama de classes da UML (*Unified Modeling Language*);

O Modelo de Hipertexto descreve os documentos hipertexto que podem ser publicados no *site*. Cada hipertexto define uma visão do *site*, que é dividida em dois submodelos: o de composição (que define as páginas e sua organização interna em termos de elementos) e de navegação (que especifica os *links* entre as páginas);

O Modelo de Derivação é um processo de adição de informações à estrutura do esquema, visando aumentar o detalhamento da informação, oferecendo diferentes formas de visualização dos mesmos dados. Como exemplo de derivação, pode-se citar a importação de atributos de uma entidade para outra, como no caso de um álbum de músicas que importa o nome dos cantores, dos estilos das músicas e da produtora.

O Modelo de Apresentação descreve o leiaute e a aparência gráfica das páginas, independentemente da linguagem final que representará as páginas.

A Figura 25 mostra um modelo de hipertexto representado pelos sub-modelos de composição (elementos internos às caixas) e de navegação de páginas (setas de uma caixa a outra). Ainda na mesma imagem, a representação das interfaces relacionadas com os sub-modelos.

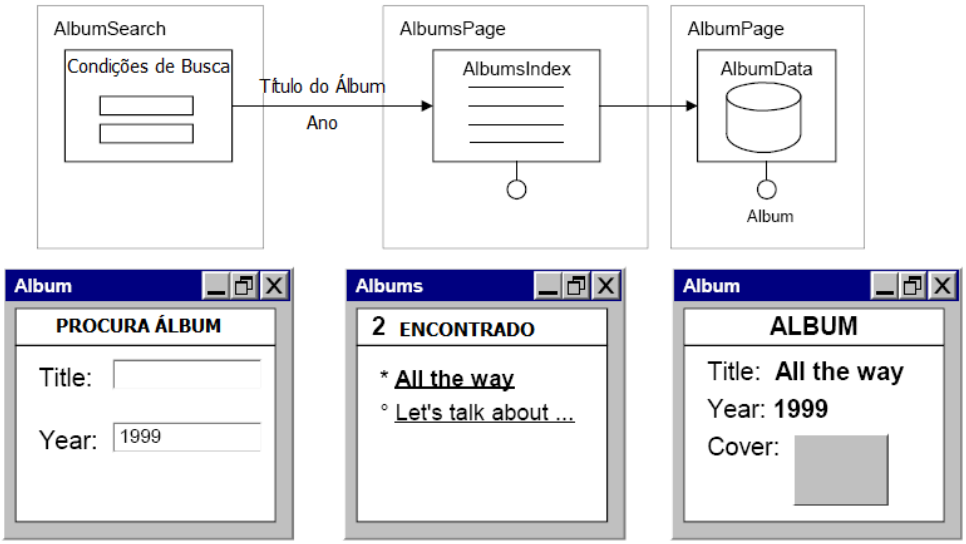


Figura 25 - Modelo de navegação WebML e as interfaces resultantes
Fonte: Adaptado de http://www.webml.org/webml, acesso em: 16 de maio de 2009

3.4 Processo de Desenvolvimento Simultâneo

O processo de desenvolvimento proposto Gonçalves *et al.* (2005) foi elaborado com base em um estudo de caso, pelo qual os autores tinham como objetivo entender como era feito o desenvolvimento multidisciplinar de aplicações Web, com a participação do usuário e funcionalidade complexa.

Na proposta é adotado um processo que separa as atividades relacionadas com aspectos de autoria (processo responsável pelo trabalho criativo de produção e organização do conteúdo estético e informativo) dos aspectos de infra-estrutura (processo responsável pelas atividades tipicamente encontradas nos processos de desenvolvimento de software).

O estudo de caso foi elaborado a partir de cinco aplicações Web, desenvolvidas em uma instituição de pesquisa em tecnologia de grande porte, onde foram adotados os seguintes papéis e qualificações: o projetistas Web, o Web designer e o analista de banco de dados.

O Projetista Web é responsável pelo planejamento da aplicação como um todo e gerenciamento do trabalho da equipe, e faz ainda, a ponte entre os aspectos funcionais e os estéticos da aplicação. O projetista também é responsável pela programação das páginas dinâmicas bem como a integração com o banco de dados.

O Web designer é responsável pela concepção visual da aplicação, planejamento e criação das mídias, definição de cores, tipografia e aplicação de logomarcas.

O analista de banco de dados é responsável pela criação lógica e física da estrutura de dados do sistema, implementação do banco de dados, definição e implementação dos procedimentos armazenados em banco de dados.

O processo de desenvolvimento simultâneo (Figura 26) é composto de quatro fases: levantamento de requisitos preliminar, protótipo não-funcional, implementação e refinamento.

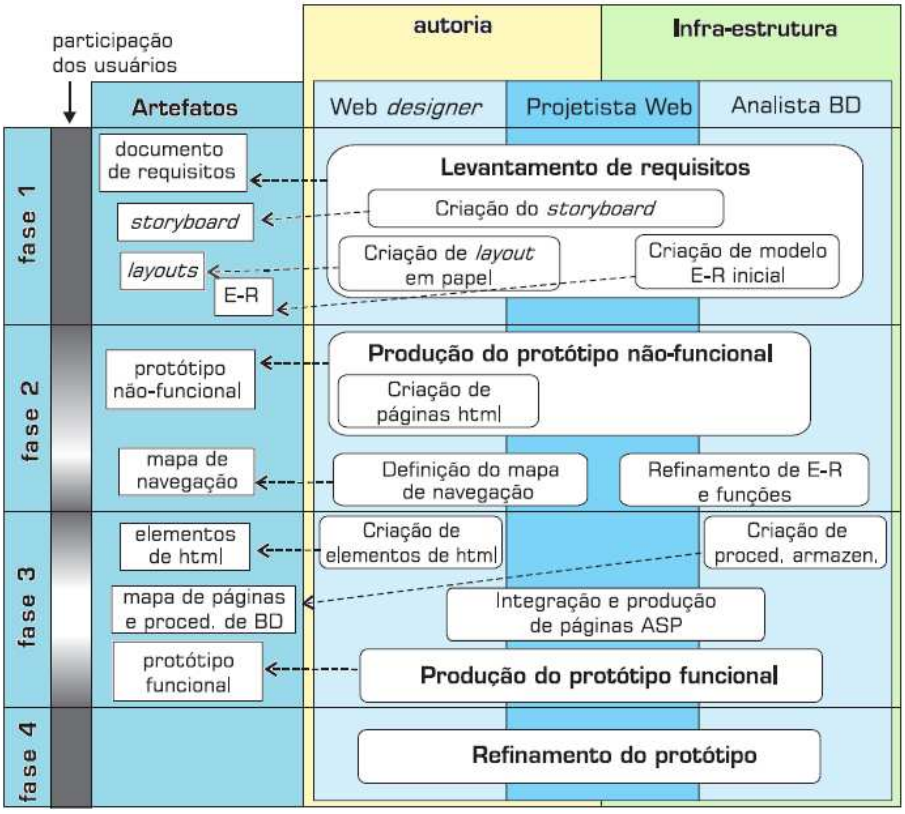


Figura 26 - Processo de Desenvolvimento Simultâneo
Fonte: Adaptado de Gonçalves *et al.* (2005, p. 387)

A fase 1 compreende o levantamento de requisitos com o usuário e com a participação ativa dos interessados e usuários-chave, documentação das entrevistas e montagem por etapas de *storyboards* e protótipos em papel do protótipo não-funcional, contendo as interfaces e a estrutura inicial de navegação. Nessa fase também é elaborada uma visão inicial do modelo de dados da aplicação.

A fase 2 compreende as iterações/interações da equipe de desenvolvimento com os usuários e interessados, para refinamento do protótipo não-funcional desenvolvido na fase anterior. A adequação da estrutura visual com a estrutura funcional é feita pelo Web designer em conjunto com o projetista Web, que se baseiam no protótipo de interface pretendida, para guiar o trabalho de interação com os participantes das sessões.

Ao final desta fase, o Web designer terá criado as páginas HTML, sem códigos de programação, mas validadas pela equipe de desenvolvimento e pelos usuários, usuários-chave e interessados pelo sistema.

A fase 3 compreende a geração dos procedimentos de armazenamento pelo analista de banco de dados. O projetista Web gera o protótipo funcional a partir das páginas HTML elaboradas pelo Web designer e da integração com a base de dados elaborada pelo analista de banco de dados.

Na fase 4 realiza-se o refinamento final do protótipo funcional, que é feito com a participação ativa de todos os usuários e da equipe de desenvolvimento, com a operação efetiva do sistema no ambiente de trabalho. A fase é considerada como um ajuste fino da aplicação, pois, considera-se que as principais funcionalidades já foram implementadas nas fases anteriores.

4. O Processo de Desenvolvimento de Software do 3º CTA

Em 2006, por determinação do chefe do 3º CTA, foi elaborado o Processo de Desenvolvimento de Software (PDS), com o objetivo estabelecer critérios e procedimentos metodológicos para o desenvolvimento de software na Seção de Sistemas.

O PDS é um documento que define de maneira geral, vários pontos do processo de desenvolvimento de software, tais como os agentes do processo, estudo de viabilidade, levantamento de requisitos, as fases do desenvolvimento e disponibiliza outros documentos e modelos que serão detalhados a seguir.

Como agentes do processo, o PDS descreve cinco funções: o Coordenador-geral, o Gerente de projeto, o Administrador de dados e banco de dados, o Programador visual e o Analista programador.

As atividades de engenharia de software estão agrupadas em três fases básicas: os estudos preliminares, a análise e a implantação, que serão estudados a seguir.

4.1 Estudos preliminares

Os estudos preliminares, a primeira fase do processo, iniciam-se logo após a formalização do pedido e aprovação do desenvolvimento de software. A meta desta fase é a elaboração de um anteprojeto, chamado de Estudo de Viabilidade de Aplicativo (EVA), que será o documento de entrada para a fase seguinte (fase de análise).

A elaboração do EVA compreende atividades de coleta de informações, para as quais o PDS sugere a técnica de entrevista, que pode ser realizada em três níveis de estratégias: as iniciais, para coleta de dados e seqüenciamento do trabalho a ser desenvolvido; as de revisão, para que dúvidas e problemas detectados sejam debatidos e resolvidos; e finalmente, a de encerramento, para apresentação, discussão e aprovação do EVA.

O EVA é um documento formal e padronizado, que contém todo esforço de coleta de informações da fase preliminar do processo. O estudo se divide em dois grandes grupos: situação atual e a situação proposta.

A situação atual busca coletar informações e compreender o problema, o ambiente, as tarefas do usuário, o processo, os recursos em softwares e hardwares existentes, as pessoas envolvidas e as necessidades percebidas.

A situação proposta compreende as ações necessárias para a solução do problema, sendo composta dos seguintes itens: tipo da necessidade, o acordo de serviço, a estimativa inicial de custos e o plano de recursos humanos.

O tipo de necessidade descreve de maneira geral, um sistema computacional que soluciona o problema proposto. O item é descrito em termos de linguagens, ambientes, bancos de dados, plataformas (sistema operacional), hardware necessário e a rede de comunicações.

O acordo de serviço ressalta pontos importantes do software a ser desenvolvido: lista os benefícios que o sistema dever proporcionar, as prioridades que se constituem nos requisitos mais importantes e os riscos.

As estimativas de custos compreendem os gastos estimados com equipamentos e softwares. No caso dos softwares, cabe ressaltar que estes devem sempre ser livres, tendo em vista o Plano de Migração para Software Livre no Exército Brasileiro (BRASIL, 2007).

O plano de recursos humanos lista as pessoas, em termos de função, necessárias ao desenvolvimento, instalação e suporte. A lista informa o quantitativo de funções, as horas previstas, o custo individual e total desses recursos.

O EVA finaliza com um resumo que descreve novamente o problema encontrado com a contrapartida dos benefícios da solução apresentada, indicando a viabilidade ou não de seu desenvolvimento.

4.2 Análise

A fase de análise compreende as atividades de modelagem, codificação, banco de dados e documentação técnica do sistema. O PDS possibilita duas abordagens para a análise e desenvolvimento: a análise estruturada ou a análise orientada a objetos. As duas abordagens são descritas de forma geral e são oferecidos exemplos de gráficos, diagramas, notações e documentação.

O Estudo de Viabilidade de Aplicativo se constitui no documento de entrada para esta fase e apesar de conter requisitos do usuário, a análise deve produzir um grupo exato de requisitos do sistema a partir do refinamento e de outras coletas que serão executadas.

Para a análise são ressaltados alguns pontos considerados importantes: o envolvimento efetivo do usuário, a identificação dos processos-chave e as suposições para o projeto e a implantação do sistema.

Nesta fase, o PDS cita de forma geral, o projeto de telas do sistema: “Primeiramente realiza-se um Projeto de Telas, com o objetivo de esboçar toda a parte gráfica no sistema. Faz-se necessário especificar o tipo de plataforma, campos existentes, imagens, dimensões, estudo de cores, animações e tipos de fontes” (BRASIL, 2006, p. 9) e é exatamente neste ponto que se insere a proposta do presente estudo, ou seja, apresentar um método com os processos de concepção e avaliação de interfaces.

A finalização da fase de análise compreende o EVA, que foi elaborado na fase preliminar, a documentação técnica elaborada nesta fase e o sistema codificado, depurado e em condições de ser implantado.

4.3 Implantação

A implantação compreende as atividades de confecção do material de treinamento, de testes de operação, homologação, completamento da documentação do sistema e questionário de satisfação.

Apesar de mencionar, o PDS não detalha a composição do material de

treinamento e as estratégias dos testes de operação do sistema.

Pelo PDS, a documentação do sistema com o intuito de orientar a operação pelo usuário, deve ser desenvolvida durante os testes de operação, em razão de mudanças que podem ocorrer nesta fase.

A homologação do sistema constitui no recebimento formal do software pelo solicitante, com a lista dos requisitos cumpridos.

A finalização da fase de implantação compreende a entrega e aceite do sistema pelo solicitante, a documentação do sistema, a definição dos responsáveis pela manutenção e controle de qualidade e no questionário de satisfação.

4.4 Considerações sobre o PDS do 3º CTA

Um processo de software se constitui em uma série de passos previsíveis, mais precisamente um roteiro, que ajuda a criar a tempo um resultado de alta qualidade (PRESSMAN, 2006).

O Processo de Desenvolvimento de Software do 3º CTA apresenta-se como um roteiro de atividades, divididas em três fases globais: estudos preliminares, análise e implantação.

Pressman (2006) menciona a necessidade de cinco atividades genéricas para o processo de desenvolvimento de software: a comunicação, que envolve a colaboração com o cliente e o levantamento de requisitos; o planejamento, que estabelece o plano de trabalho de engenharia, as tarefas técnicas, os riscos, os recursos, os produtos de trabalho a ser produzidos e o cronograma; a modelagem, que inclui a criação de modelos que permitam ao desenvolvedor e cliente o melhor entendimento dos requisitos, bem como o projeto que satisfará esses requisitos; a construção, que compreende a geração de código e os testes necessários para revelar erros nestes; e a implantação, que compreende a entrega do software completo ao cliente, que avalia e fornece *feedback*s com base na avaliação.

Ao se fazer uma analogia entre as fases globais definidas pelo PDS e o modelo proposto por Pressman (2006), verifica-se que a fase de estudos

preliminares do PDS engloba as atividades de comunicação e planejamento; a fase de análise engloba as atividades de modelagem e construção; e por fim, a fase de implantação é tal como apresentada no modelo de Pressman.

Apesar de não citar claramente qual o modelo prescritivo adotado, a abordagem do PDS é sistemática e seqüencial, sugerindo que o modelo utilizado é o "em cascata", pois as atividades de engenharia de software descritas estão englobadas nas três fases globais estudadas (Figura 27).

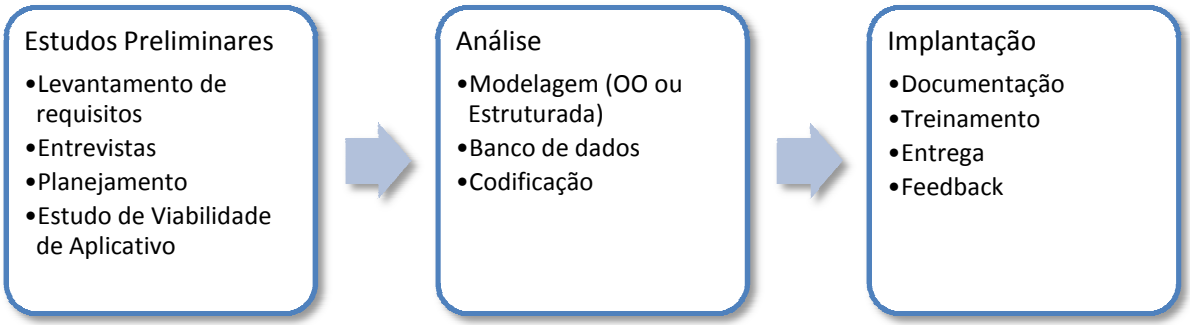


Figura 27 - Modelo prescritivo do PDS.
Fonte: o autor.

Modelos prescritivos de processo, segundo Pressman (2006. p. 37), "[..] definem um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalho que são necessários para fazer engenharia de software com qualidade".

Originalmente elaborado para disciplinar o processo de desenvolvimento de software convencional, o PDS não contempla as particularidades das aplicações Web, que se tornaram a plataforma básica de desenvolvimento, conforme o Plano de Migração para o Software Livre no âmbito do Exército Brasileiro (PMSLEB).

Quanto aos processos de concepção e avaliação de interfaces, o PDS cita a realização de um Projeto de Telas, porém não este não orienta quanto às questões de avaliações, princípios e diretrizes de usabilidade. Desta forma, a atividade de projeto e construção de interfaces fica exclusivamente condicionada ao conhecimento e a habilidade do Programador Visual.

Após estudo dos processos de desenvolvimento Web, apresentados no capítulo anterior e da descrição e considerações gerais do PDS, faz-se necessário verificar, dentre os modelos apresentados, qual o mais adequado para servir de

base para construção do método de desenvolvimento de aplicações Web proposto por esta dissertação.

4.5 Definição do processo base para a construção do método

O método de projeto de hipermídia orientado a objetos (OOHDM) proposto por Schwabe e Rossi (1998) é composto de etapas pelas quais os modelos são construídos ou enriquecidos. Os modelos são concebidos a partir de classes e objetos, fazendo uma alusão à orientação a objetos, apesar de não exigir que a implementação seja orientada a objetos.

Nesse sentido, o método difere parcialmente do PDS do 3º CTA, que em sua construção permite que o desenvolvimento tenha dois caminhos de codificação, a modelagem orientada a objetos e a modelagem estruturada.

A Linguagem de Modelagem Web (WebML), proposta por Ceri *et al.* (2003) permite a modelagem de aplicações em alto nível de abstração, porém, como ocorre também como o método OOHDM, ela não trata de protótipos, sejam funcionais ou não, nem dos papéis dos envolvidos no desenvolvimento.

O Processo de Desenvolvimento Simultâneo proposto por Gonçalves *et al.* (2005), sugere que o desenvolvimento das aplicações Web possa ser realizado, em processos de autoria e de infra-estrutura.

Observam-se algumas similaridades do processo proposto com por Gonçalves *et al.* (2005) em relação ao PDS do 3º CTA:

- As atividades de infra-estrutura coincidem com as atividades de infra-estrutura já previstas no PDS (Figura 27).
- Apesar da definição explícita dos papéis dos desenvolvedores, o Processo de Desenvolvimento Simultâneo permite que estes variem, havendo espaço para inclusão e exclusão de funções. O PDS, por sua vez, prevê a seguinte equipe: Coordenador-geral, o Gerente de projeto, o Administrador de dados e banco de dados, o Programador visual e o Analista programador. Analisando-se as funções previstas entre os

> dois processos, tem-se o Administrador de banco de dados nos dois processos, o Analista programador e o Programador visual previstos no PDS têm função semelhante ao do projetista Web e o Web designer, respectivamente, no Processo de Desenvolvimento Simultâneo. O coordenador geral constitui-se em uma função mais administrativa, não técnica e o Gerente de Projeto tem conhecimento técnico e pode exercer o papel de projetista Web.

Em relação aos métodos OOHDM e WebML estudados anteriormente, verifica-se que a estrutura de atividades são bem específicas e exigiriam mudanças significativas no PDS do 3º CTA e nos processos de trabalho do desenvolvedores. O Processo de Desenvolvimento Simultâneo, por sua vez, encontra maior aderência por apresentar similaridades com as atividades do PDS do 3º CTA, sendo por esta razão, escolhido como base para o método de desenvolvimento de aplicações Web a ser construído por este estudo.

5. Desenvolvimento do Método para Aplicações Web

Um método, segundo Pfleeger (2004) e Sommerville (2007), constitui-se de um procedimento formal, ou ainda, de uma abordagem estruturada com o objetivo de se atingir um determinado resultado. Para Pressman (2006), os métodos fornecem a técnica de “como fazer” e mais especificamente para a engenharia de software, estes abrangem um conjunto de tarefas tais como a análise de requisitos, modelagem de projeto e construção de programas.

O Método para Aplicações Web (Figura 28), proposto pela presente dissertação é composto de três fases: os Estudos Preliminares, cuja atividade principal é a coleta de requisitos para elaboração dos primeiros protótipos, do mapa de navegação e do modelo de análise inicial (Orientado a Objetos ou Estruturado); a Análise e Implementação, que compreende atividades baseadas nos requisitos e artefatos produzidos na fase anterior, deve possibilitar ciclos de refinamento dos protótipos, elaboração da codificação lógica do sistema, integração com as páginas em HTML e, finalmente, a concepção da versão final do sistema; e, a Implantação, que compreende as atividades de homologação da versão final, aceite formal do sistema pelo cliente, preparação do material de treinamento e treinamento dos usuários.

A participação dos usuários é contemplada pelo Método, onde o grau de participação varia de acordo com a fase do projeto. A representação da participação mais ativa ou não, é representada pela coluna “Participação dos usuários”, na qual, as cores mais escuras representam grau maior de participação.

As atividades das fases do projeto geram os artefatos (documentos, arquivos, *scripts*, códigos ou páginas) que são apresentados na sua respectiva coluna “Artefatos”. Os artefatos representados por linhas tracejadas representam itens cuja construção é opcional.

As atividades de desenvolvimento são divididas em duas linhas de produção: a Autoria, que é o processo responsável pelo trabalho criativo de produção das interfaces e pela organização do seu conteúdo estético e informativo; e, a Infra-estrutura, que é o processo responsável pela de criação do modelo lógico e do

banco de dados da aplicação, atividades tipicamente encontradas nos processos da engenharia de software.

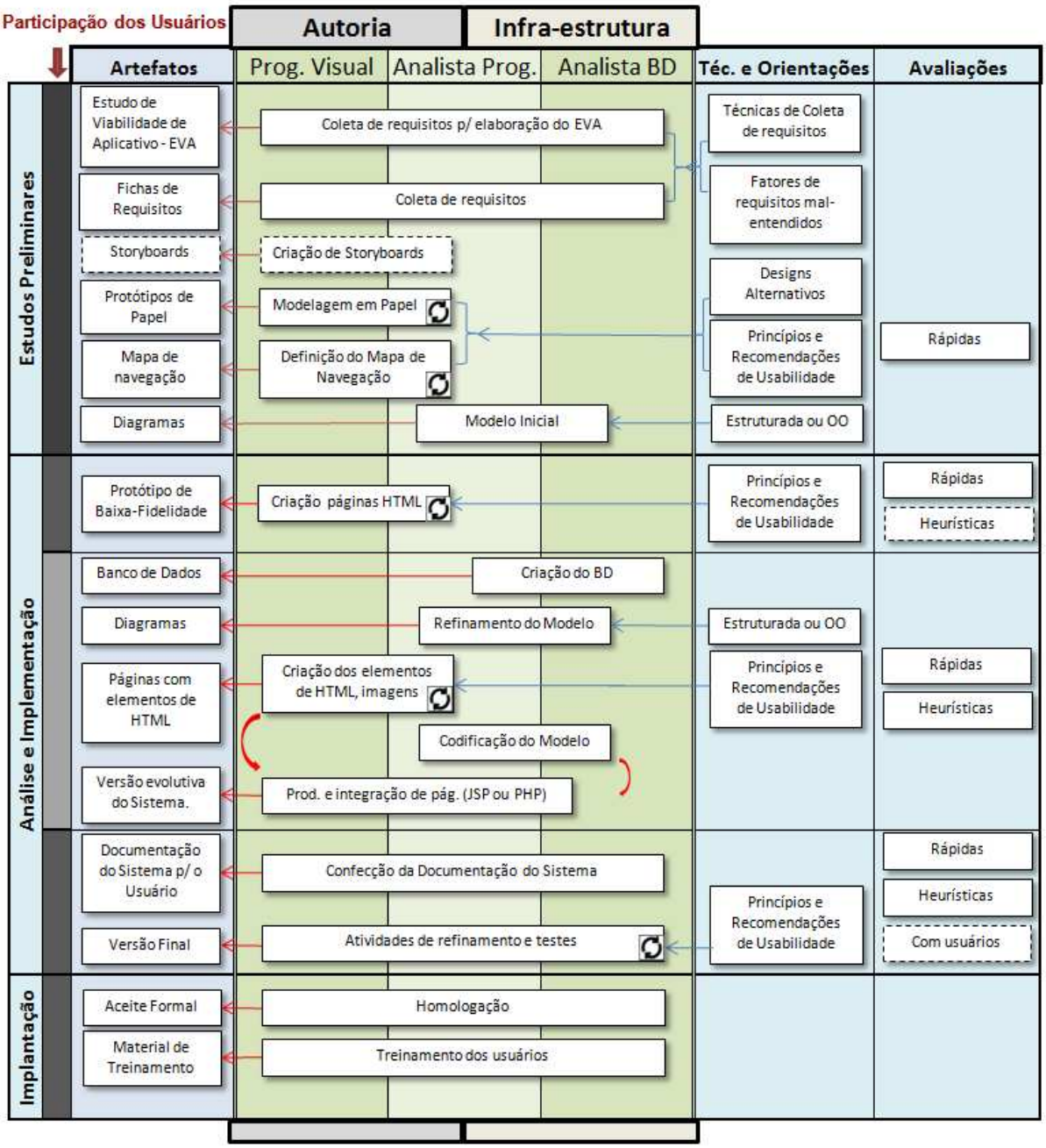


Figura 28 - Método de Desenvolvimento de Aplicações Web
Fonte: o autor.

O método prevê que os desenvolvedores possam desempenhar três papéis distintos dentro das linhas de produção (Autoria e Infra-estrutura): o Programador Visual, que atua especificamente na linha de Autoria; o Analista Programador, que atua e deve ter conhecimento técnico nas duas linhas de produção; e, o Analista de Banco de Dados, que atua somente na linha de Infra-estrutura.

As atividades são distribuídas ao longo das colunas dos papéis dos desenvolvedores. A sobreposição das atividades sobre as colunas indica quem é ou, quem são os responsáveis pela execução. As atividades também sobrepõem as linhas de produção, indicando se a atividade é de Autoria, Infra-estrutura ou de ambas.

Algumas atividades são dinâmicas, envolvem ciclos sucessivos de análise, concepção e teste, por essa razão são representadas no método com a adição de um ícone (🔄).

A coluna “Técnicas e Orientações” relaciona as técnicas e orientações necessárias à execução das atividades durante as fases de desenvolvimento. As setas indicam especificamente a atividade relacionada.

As avaliações são recomendadas (itens com linha contínua) ou sugeridas (itens com linha pontilhada) dependendo da fase e constam da última coluna do método. As avaliações não se constituem em atividade única em cada fase e podem ser realizadas quantas vezes forem necessárias.

5.1 Estudos Preliminares

A primeira fase do Método compreende as atividades de levantamento de requisitos com o objetivo de entender ao máximo possível os usuários, suas tarefas, o contexto do trabalho e iniciar as primeiras diagramações e modelagens de telas, para possibilitar que o sistema a ser construído ofereça o suporte necessário para atingir seus objetivos.

Por ser uma fase decisiva para a execução do projeto, os Estudos Preliminares envolvem intensa participação dos usuários e de toda equipe de desenvolvimento. A fase prevê a geração de seis artefatos: o Estudo de Viabilidade de Aplicativo, as Fichas de Requisitos, os *Storyboards*, os Protótipos de Papel, o Mapa de Navegação e os Diagramas (os diagramas dependem da modelagem escolhida: orientado a objetos ou estruturado). A fase descreve também, as atividades necessárias para a construção dos artefatos, os itens que suportam as atividades e as avaliações.

5.1.1 O Estudo de Viabilidade de Aplicativo (EVA)

O EVA é o primeiro artefato elaborado pela equipe de desenvolvimento, com os requisitos coletados a partir dos primeiros contatos com os usuários. O EVA constitui-se em um anteprojeto e também um acordo de serviço com o cliente (normalmente comandantes, chefes ou diretores de OMs, Comandos Militares, Diretorias ou Departamentos), e deve descrever sobre a situação atual e a proposta de solução.

A situação atual deve descrever sobre: o problema, as tarefas dos usuários, o processo, os recursos em softwares e hardwares existentes, as pessoas envolvidas e as necessidades percebidas.

A proposta de solução deve descrever, de maneira geral, as ações e sistemas computacionais (*software* e *hardware*) que solucionem o problema apresentado. Deve apresentar um acordo de serviço que descreva sobre: os benefícios com a adoção do sistema, as prioridades, os riscos, as estimativas de custos, os recursos humanos necessários e, ao final do estudo, discorrer sobre a viabilidade ou não do desenvolvimento do sistema.

Todos os desenvolvedores devem participar da coleta de dados para a elaboração do EVA, pois é um artefato composto de requisítos das duas linhas de produção (Autoria e Infra-estrutura).

Para a coleta de requisitos poderão ser utilizadas técnicas de entrevista, questionário, grupos de foco ou observação. A equipe deve ater-se às preocupações com os fatores de requisitos mal-entendidos.

5.1.2 Fichas de Registro de Requisitos

As fichas de registro devem relacionar todos os requisitos coletados. Para organização e controle, as fichas podem ser agrupadas por tipo de requisitos funcionais e não funcionais:

- Os requisitos funcionais dizem respeito ao que o sistema deve fazer e neste caso, podem ser subdivididas em fichas com requisitos

direcionados à lógica de operação e requisitos direcionados para lógica de funcionamento do sistema;

- Os requisitos não-funcionais indicam as limitações no sistema e no seu desenvolvimento, como por exemplo, “o sistema deve ser desenvolvido para operar em ambiente Web”, “o sistema deve estar pronto em oito meses”.

Os requisitos relacionados com a lógica de operação são em geral direcionados para a linha Autoria e os requisitos relacionados com a lógica de funcionamento, são por sua vez, direcionados para linha de Infra-estrutura. O direcionamento sugerido tem por objetivo a organização dos requisitos, porém, deve-se atentar que os requisitos direcionados para a linha de Autoria, por exemplo, certamente refletirão na lógica da linha de Infra-estrutura e vice-versa.

Todos os desenvolvedores devem participar da coleta de requisitos, pois é um processo que envolve duas linhas de produção (Autoria e Infra-estrutura).

Para a coleta de informações podem ser utilizadas técnicas de entrevista, questionário, grupos de foco ou observação. A equipe deve ater-se às preocupações com os fatores de risco quanto aos requisitos mal-entendidos (Tabela 1, p. 66).

5.1.3 *Storyboards* (artefato opcional)

Os *storyboards* compreendem uma seqüência de desenhos que devem representar as interações entre os usuários e o sistema. Os *storyboards* devem ser feitos em folhas grandes e coladas em uma parede ou um quadro, para que sejam validados pelos usuários e pela equipe de desenvolvimento com base nos requisitos de usabilidade.

O recurso de construção de *storyboards* pode ser utilizado quando for necessária a elucidação de requisitos relacionados com as interações entre o usuário e o sistema.

Os *storyboards* são artefatos resultantes das atividades produzidas pela linha de Autoria e de responsabilidade direta do Programador Visual com auxílio ou direcionamento do Analista Programador.

5.1.4 Protótipos de Papel

Os protótipos são esboços de telas, desenhados em folhas de papel, que possibilitem aos usuários, desenvolvedores e interessados interagirem com o produto desejado.

Elaborados a partir da coleta dos requisitos funcionais e referentes à lógica de operação do sistema, os protótipos de papel são as primeiras versões interativas do sistema.

Os protótipos, resultados das atividades de modelagem em papel, são produzidos pela linha de Autoria e de responsabilidade direta do Programador Visual com auxílio do Analista Programador. Para execução desta atividade recomenda-se que o desenvolvedor encarregado de desenhar os protótipos de papel não se sinta inibido em razão da qualidade de seus desenhos.

Para a atividade de modelagem de protótipos, o Programador Visual e o Analista Programador devem:

- Verificar a existência de *designs* semelhantes (*Designs* Alternativos) de forma concorrente ou de versões anteriores, para facilitar a escolha entre as diversas formas de interação;
- Atentar para os princípios e recomendações de usabilidade;
- Atentar para o fato de que os protótipos, mesmo os de papel, são passíveis de avaliações “Rápidas” (onde o Programador Visual faz reuniões informais com o usuário a fim de tirar dúvidas do *design* escolhido ou de algum requisito obscuro).

5.1.5 Mapa de Navegação

O Mapa de Navegação deve mostrar como os usuários navegarão pelas páginas da aplicação Web por meio de uma representação em árvore das páginas e dos *links* entre elas.

O Mapa de Navegação é resultado das atividades de produção da linha de Autoria e de responsabilidade direta do Programador Visual com auxílio do Analista

Programador.

Para a atividade de definição do Mapa de Navegação, o Programador Visual deve:

- Atentar para os princípios e recomendações de usabilidade;
- Atentar para o fato de que os Mapas de Navegação são passíveis de avaliações “Rápidas” (onde o Programador Visual faz reuniões informais com o usuário a fim de tirar dúvidas do *design* escolhido ou de algum requisito obscuro).

5.1.6 Diagramas

Os Diagramas dependem do padrão de análise desenvolvimento escolhido pelo time de desenvolvimento. O PDS permite duas linhas possíveis para a modelagem e codificação lógica do sistema: a orientada a objetos ou a estruturada.

Nos dois casos, os diagramas nesta fase, por serem iniciais, provavelmente não contemplam todas as necessidades do sistema, mas procuram atender os requisitos funcionais voltados para a lógica de funcionamento do sistema.

Para a análise estruturada devem produzidos os seguintes diagramas:

- Diagramas hierárquicos de funções;
- Diagramas de Fluxo de Dados (DFD);
- Diagramas de Contexto.

Para a análise orientada a objetos devem ser produzidos os seguintes diagramas:

- Diagramas de casos de uso;
- Diagramas de classe;
- Diagramas de seqüência.

Os diagramas (estruturados ou OO) são artefatos produzidos pela linha de Infra-estrutura e o responsável pela sua elaboração é o Analista Programador, com auxílio do Analista de Banco de Dados.

5.2 Análise e Implementação

O objetivo da fase de Análise e Implementação é a construção da versão final do sistema. Para este objetivo, as atividades da linha de Autoria refinam os protótipos até a versão final do sistema, e paralelamente, as atividades da linha de Infra-estrutura geram os artefatos necessários para darem suporte para as funcionalidades necessárias.

A fase de análise pressupõe um grau menor de participação dos usuários, principalmente pelo volume de trabalho exigido pelas atividades de Infra-estrutura (criação do banco de dados, refinamento do modelo lógico, codificação do modelo lógico e integração com as páginas), onde não há a necessidade de participação direta do usuário, porém, o contato com os usuários sempre deverá ser feito, a fim de elucidação de lacunas ou de requisitos não claros.

A execução desta fase produz seis artefatos: os Protótipos de baixa fidelidade, o Banco de Dados, os Diagramas, as Páginas com elementos de HTML, a Versão Evolutiva do Sistema, a Documentação do Sistema para ao usuário e a Versão Final do Sistema.

5.2.1 Protótipos de Baixa Fidelidade

Os protótipos de baixa fidelidade desta fase devem ser construídos em HTML, com base nos protótipos de papel e na consulta dos *designs* alternativos, elaborados na fase de Estudos Preliminares. Normalmente, os protótipos de baixa fidelidade não se assemelham muito ao produto final pois devem ser simples e de rápida produção.

Os protótipos de baixa fidelidade são produtos da linha de produção de Autoria e de responsabilidade do Programador Visual, com supervisão ou auxílio do Analista Programador.

Para execução da atividade o Programador Visual deve:

- Atentar para os princípios e recomendações de usabilidade;
- Atentar para o fato de que os protótipos de baixa fidelidade são passíveis de avaliações “Rápidas” (onde o Programador Visual faz

reuniões informais com o usuário a fim de tirar dúvidas do *design* escolhido ou de algum requisito obscuro) e também de Avaliações Heurísticas.

5.2.2 Banco de Dados

O artefato de Banco de Dados compreende: o Diagrama de Entidade Relacionamento (DER), os *scripts* com a estruturação do banco e o banco de dados propriamente dito (arquivo do banco instalado em computador e em condições de ser utilizado).

O DER é composto por um conjunto de itens gráficos que visa representar todos os objetos (entidades, atributos, relacionamentos, domínios, visualizações e procedimentos) de um modelo de entidade relacionamento.

Os *scripts*, também chamados de metadados, são arquivos do tipo texto, que descrevem o banco de dados os objetos do banco de dados e podem ser utilizados para recriar o banco a qualquer tempo.

O Banco de Dados é um artefato resultante da atividade de modelagem, produzida pela linha de Infra-estrutura e de responsabilidade direta do Analista de Banco de Dados, com participação do Analista Programador.

5.2.3 Diagramas

Nesta fase, os Diagramas são um refinamento da atividade de modelagem (estruturada ou OO) que foi iniciada na fase de Estudos Preliminares e vão depender do padrão de análise escolhido pelo time de desenvolvimento.

Para a análise estruturada devem ser aprimorados os seguintes diagramas:

- Diagramas hierárquicos de funções;
- Diagramas de Fluxo de Dados (DFD);
- Diagramas de Contexto.

Para a análise orientada a objetos devem ser aprimorados os seguintes diagramas:

- Diagramas de casos de uso;
- Diagramas de classe;
- Diagramas de seqüência.

Os diagramas (estruturados ou OO) são artefatos produzidos pela linha de Infra-estrutura e o responsável pela sua elaboração é o Analista Programador, com auxílio do Analista de Banco de Dados.

5.2.4 Páginas com elementos de HTML

As páginas com elementos de HTML constituem-se em refinamentos dos protótipos de baixa fidelidade, formatadas com os componentes de página já construídos.

Além da definição dos elementos das páginas, a estrutura definida pelo Mapa de Navegação também deverá ser contemplada.

As páginas com elementos de HTML são produtos da linha de Autoria e de responsabilidade do Programador Visual com supervisão do Analista Programador.

Para a elaboração das páginas, o Programador Visual deve:

- Atentar para os princípios e recomendações de usabilidade;
- Atentar que os protótipos são passíveis de avaliações “Rápidas” (onde o Programador Visual faz reuniões informais com o usuário a fim de tirar dúvidas do *design* escolhido ou de algum requisito obscuro) e também de Avaliações Heurísticas.

5.2.5 Versão Evolutiva do Sistema

A Versão Evolutiva do Sistema é totalmente interativa, define claramente o esquema de navegação e tem a mesma aparência do sistema final.

A construção da Versão Evolutiva compreende atividades nas linhas de Autoria e Infra-estrutura:

- Codificação do modelo lógico com base nos diagramas oferecidos pelo padrão de análise (estruturado ou orientado a objetos). A atividade de

codificação pertence à linha de Infra-estrutura e é executada pelo Analista Programador com a participação do Analista de Banco de Dados;

- Integração das páginas com elementos de HTML, produzidas pela linha de Autoria com os códigos do modelo lógico produzido pela linha de Infra-estrutura. Considera-se que esta atividade é crítica e envolve os desenvolvedores das duas linhas de produção e deve ser coordenada pelo Analista Programador.

A concepção da primeira Versão Evolutiva pode ser feita por módulos (por exemplo: módulo de cadastro, módulo de consulta etc.), deve ser refinada por atividades de testes de funcionalidade, avaliações (“rápidas”, heurísticas e também com usuários) e verificada quanto aos princípios e recomendações de usabilidade.

Outro recurso para o refinamento é a distribuição da Versão Evolutiva aos usuários. Este procedimento ajuda na descoberta de problemas e treina o usuário na utilização do sistema.

5.2.6 Documentação do Sistema (para o usuário)

A documentação do sistema orienta o usuário quanto à operação do sistema e deve contemplar os seguintes itens como: título e versão; índice; objetivo do sistema; histórico; características técnicas (necessidades em *software* e *hardware*); características operacionais; funcionamento do sistema; instruções de instalação; detalhamento dos módulos do sistema; segurança (senhas e níveis de acesso) e anexos.

5.2.7 Versão Final

Na Versão Final tem-se o sistema pronto com todas as funcionalidades implementadas conforme os requisitos coletados na fase de Estudos Preliminares, testado e pronto para ser entregue para o cliente.

As atividades para a concepção da Versão Final são executadas por todo o time de desenvolvimento:

- O Programador Visual e o Analista Programador devem atentar para os princípios e recomendações de usabilidade, realizar avaliações das interfaces do sistema (Rápidas, Heurística e se possível Testes com Usuários) e testes de funcionalidade;
- O Analista de Banco de dados deve executar testes de funcionalidade, atentar principalmente quanto ao registro, alteração, recuperação e seleção dos dados.

5.3 Implantação

A fase de Implantação compreende a homologação da Versão Final do Sistema pelo cliente e a confecção do material de treinamento. A participação do usuário nesta fase está voltada às atividades de homologação, onde será verificado se o sistema contempla todos os requisitos coletados durante a fase de Estudos Preliminares e os treinamentos.

Nesta fase são construídos dois artefatos: o Documento de Aceite do sistema pelo cliente (pode ser a Ata de Reunião de homologação) e o Material de Treinamento (documentação do sistema, apresentações de *slides* e atividades de exercício).

5.3.1 Documento de Aceite

O Documento de Aceite do Sistema constitui-se na formalização do recebimento do sistema e é assinado pelo solicitante do sistema (cliente), pelo chefe do 3º CTA, pelo chefe da Seção de Sistemas e pelo Gerente do Projeto.

O aceite do sistema pelo cliente é realizado com base na homologação que é realizada pelo cliente ou quem este designar, com a assistência dos desenvolvedores. A homologação compreende a verificação das funcionalidades e o atendimento de todos os requisitos coletados na fase de Estudos Preliminares.

O aceite também pode ser formalizado por uma Ata de Reunião de Homologação, onde se discriminará o atendimento dos requisitos pelo sistema.

5.3.2 Material de Treinamento dos Usuários

O material de treinamento constitui-se em apresentações e documentações do sistema. O material pode ser elaborado pela equipe de desenvolvimento ou por outros integrantes da Seção de Sistema, assessorados pelos desenvolvedores.

Também devem ser elaborados, sempre que possível, alguns exercícios para treinamento e aplicação no sistema. A realização de treinamentos estará limitada à disponibilidade dos usuários.

As atividades de treinamento podem ser executadas em campo, junto aos usuários em seu ambiente de trabalho.

5.4 Considerações sobre o Método Proposto

O Método proposto para aplicações Web define o fluxo de trabalho, as atividades, os artefatos, as funções dos envolvidos no processo e procura atender aos requisitos necessários de um processo de desenvolvimento de sistemas para uso em ambiente Web. Oferece mecanismos para guiar o time de desenvolvimento quanto à ordem das atividades, especifica quais os artefatos devem ser desenvolvidos, dirige as tarefas dos desenvolvedores de forma individual e o time de desenvolvimento como um todo.

O foco nos usuários durante o desenvolvimento de aplicações Web é uma característica básica da engenharia de usabilidade e é indicada no Método por um ícone () em atividades que podem ser consideradas cíclicas.

Ao contemplar a participação dos usuários no processo, o Método possibilita que os desenvolvedores tenham um melhor entendimento das atividades, das necessidades e dos objetivos desses usuários; permite que as expectativas dos usuários quanto ao sistema sejam realistas, evitando surpresas quando da entrega; e, favorece o sentimento de apropriação do sistema pelos usuários, o que é desejável, pois estes ficam mais receptivos para a aceitação do sistema.

A participação do usuário contemplada pelo Método pressupõe a existência de proximidade com o usuário, porém, sabe-se que em alguns projetos o universo

de usuários pode ser amplo e de difícil determinação. No caso do 3º CTA, os projetos desenvolvidos são voltados para o ambiente corporativo e os usuários são, na maioria, militares ou de setores específicos da sociedade civil.

A divisão da equipe de desenvolvimento em linhas de produção (autoria e infra-estrutura) impõe algumas condições que devem ser observadas quando do uso do Método. Conforme a estrutura e seqüenciamento das atividades, os artefatos produzidos pela linha de autoria e de infra-estrutura devem ser integrados para formarem uma versão evolutiva do sistema.

Cabe observar que as atividades e os artefatos podem ser realizados por módulos ou partes do sistema, que vão sendo parcialmente integrados para formarem a versão evolutiva do sistema. Desta forma, e também dependendo do sistema e usuários, podem ser distribuídos módulos do sistema aos usuários.

O sucesso da integração desses artefatos, proposta pelo Método, está principalmente relacionado com a capacidade e o conhecimento técnico do Analista Programador, que por atuar nas duas linhas de produção, deve coordenar os desenvolvedores das outras linhas, de forma a evitar o desenvolvimento conflitante que pode dificultar a integração.

Pela estrutura de atividades e funções proposta pelo Método, o Programador Visual deve possuir conhecimento técnico para elaboração de artefatos Web, tais como páginas, estruturas de navegação, elementos visuais como ícones, botões, menus, cores e painéis. O trabalho deste profissional está diretamente ligado às questões de usabilidade, razão pela qual este deve conhecer e desenvolver em conformidade com os princípios e as diretrizes de usabilidade.

O Analista de Banco de Dados tem suas atividades voltadas para a engenharia de software e, juntamente com o Analista Programador, deve possuir o conhecimento técnico para a elaboração dos modelos e do banco de dados.

Outro fator necessário é a existência de uma estreita comunicação entre os desenvolvedores, pois a falta ou a deficiência deste requisito pode restringir ou mesmo inviabilizar a utilização do Método, principalmente por grandes equipes de

desenvolvimento que estejam distribuídas por localidades diferentes.

A aderência do Método ao PDS do 3º CTA destaca-se pela sua subdivisão em três fases, que são semelhantes ao PDS: os Estudos Preliminares, a Análise e Implementação e a Implantação. As fases, além de compreenderem as atividades de engenharia de software convencional previstas no PDS, contemplam também as atividades, os artefatos e as técnicas e orientações voltadas para aplicações Web, com foco no usuário e em usabilidade.

6. Conclusão

Com o objetivo de oferecer sistemas que possam interagir de forma mais simples e atraente sem, no entanto, comprometer as funcionalidades, as empresas desenvolvedoras de software vêm incluindo conceitos, metodologias e técnicas que melhoram a qualidade das interfaces de seus sistemas. Desta maneira, visam também aumentar as chances de sucesso de seus produtos no mercado.

A Usabilidade, considerada como um sinônimo de facilidade de uso de um produto, é definida como um atributo de qualidade mensurável que engloba os componentes de uma interação: os usuários, as tarefas, o ambiente e os valores reais ou desejados de eficácia, eficiência e satisfação.

Tendo em vista o direcionamento da Usabilidade para as interfaces e interações, quanto aos sistemas computacionais, convém utilizar os princípios, as diretrizes e as técnicas de avaliação para a elaboração de interfaces mais usáveis.

As atividades necessárias para a concepção de interfaces usáveis são descritas pela Engenharia de Usabilidade, cujo foco está baseado em três características básicas: no desenvolvimento centrado no usuário, na identificação e documentação dos objetivos específicos dos usuários, e na possibilidade de realização de ciclos que permitam refinar o *design*.

O Processo de Desenvolvimento de Software (PDS) do 3º CTA, apesar de disciplinar o de desenvolvimento de sistemas, não trata das atividades de elaboração e avaliação de interfaces nem aborda as questões de usabilidade. Além disso, o PDS não contempla as particularidades do desenvolvimento de aplicações Web.

A presente dissertação, baseada nas lacunas mencionadas, apresentou a proposta de um Método de desenvolvimento de aplicações Web focado em usabilidade e suplementar ao PDS do 3º CTA. Para isto, foram analisados os princípios e as diretrizes de usabilidade que influenciam na concepção de interfaces, as atividades do processo da engenharia de usabilidade, alguns processos de desenvolvimento de sistemas e o próprio PDS.

Os elementos que buscam levar o Método proposto a reduzir os problemas das interfaces das aplicações Web durante o desenvolvimento e o tornam aderente ao PDS são:

- Desenvolvimento com foco no usuário;
- Atividades que considerem os princípios, as diretrizes de usabilidade e as avaliações de interfaces Web;
- Previsão de atividades cíclicas de análise, concepção e avaliações a fim de identificar e refinar continuamente as interfaces das aplicações Web e atender as exigências de usabilidade;
- Similaridade das atividades, dos artefatos e das fases do Método com as existentes no PDS.

Como trabalhos futuros a presente dissertação sugere a construção de ferramentas de software que disponibilizem os princípios e as diretrizes em formatos de fácil consulta pelos desenvolvedores de interfaces, a elaboração de guias e de modelos que sistematizem as atividades de avaliação durante o desenvolvimento e a condução de simulações com o Método para avaliação de sua eficácia e abrangência.

Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 9241-11: Requisitos Ergonômicos para Trabalho de Escritórios com Computadores: Orientações sobre Usabilidade**. Rio de Janeiro, 2002. 21 p.

_____. **NBR ISO/IEC 9126-1: Engenharia de software - Qualidade de produto**. Rio de Janeiro, 2003. 21 p.

BASTIEN, J. M. Christian; SCAPIN, Dominique L. **Ergonomic Criteria for the Evaluation of Human-Computer Interfaces**, RT nº 156. INRIA, junho de 1993. Disponível em: <http://www.inria.fr/rrrt/rt-0156.html>. Acesso em 16 de maio de 2009.

BRASIL. Ministério da Defesa. Exército Brasileiro. Portaria nº 007-DCT, de 13 de fevereiro de 2007. **Plano de Migração para Software Livre no Exército Brasileiro**. Lex: Boletim do Exército 08/2007, de 23 de fevereiro de 2007.

_____. Ministério do Planejamento, Orçamento e Gestão. **Cartilha de Usabilidade para Sítios e Portais do Governo Federal**. Modificada em 28 de maio de 2007. Disponível em: <http://www.governoeletronico.gov.br/biblioteca/arquivos>. Acesso em 16 de maio de 2009.

CERI, Stefano; FRATERNALI, Piero; BONGIO, Aldo. Web Modeling Language (WebML): a Modeling Language for Designing Web *Sites*. In: 9th INTERNATIONAL WWW CONFERENCE, Amsterdam, 15-19 de maio de 2000, **The Web: The Next Generation**. Disponível em: <http://www9.org/w9cdrom/177/177.html>. Acesso em 16 de maio de 2009.

CONALLEN, Jim. **Building Web Applications With UML**. Massachusetts: Addison-Wesley Professional, 2000. 320 p.

CYBIS, Walter; BETIOL, Adriana Holtz; FAUST, Richard. **Ergonomia e Usabilidade - Conhecimentos, Métodos e Aplicações**. São Paulo: Novatec Editora, 2007. 344p.

DE SOUZA, Clarisse Sieckenius, *et al.* Projeto de Interfaces de Usuário: perspectivas cognitivas e semióticas. In: JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA, 1999, Rio de Janeiro. **Anais...** Rio de Janeiro: Edições EntreLugar, 1999, v. 2, 425-476 p.

DIAS, Cláudia. **Usabilidade na WEB - Criando portais mais acessíveis**. 2. ed. Rio de Janeiro: Editora Alta Books, 2007. 312 p.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Editora Atlas, 2007. 175 p.

GONÇALVES, Rodrigo Franco, *et al.* Fundamentos de Engenharia Simultânea na Produção de Aplicações Web. In: XXVII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, Foz do Iguaçu, 2007. **A energia que move a produção: um diálogo sobre integração, projeto e sustentabilidade**.

_____. Uma Proposta de Processo de Produção de Aplicações Web. **Revista Produção**, São Paulo, v. 15, n 3, 376-389. Set/Dez, 2005.

International Organization for Standardization & International Electrotechnical Commission. **ISO/IEC9126: Software Product Evaluation: Quality, characteristics and guidelines for their use**. Switzerland, 1991.

International Organization for Standardization. **ISO 13407. Human-centered design for interactive system**. Switzerland, 1999.

MARTINEZ, Maria Laura. Um método de Web *design* baseado em usabilidade. In: 16º SIMPÓSIO NACIONAL DE GEOMETRIA DESCRITIVA E DESENHO TÉCNICO, 2003. Santa Cruz do Sul. **V International Conference on Graphics Engineering for Arts and Design**, Santa Cruz do Sul, 2003.

MARTINS, Cláudio Roberto de Lima. **Estratégia de Migração de Aplicações Legadas Visuais (tipo WIMP) para o Ambiente Web**. 2003. 104 f. Dissertação (Mestrado em Ciência da Computação) - UFRS, Porto Alegre, 2003.

MAYHEW, Deborah J. **The Usability Engineering Lifecycle. A practitioners's**

handbook for user interface design. San Francisco: Morgan Kaufman, 1999. 560p.

McALLISTER, Chad A. **Requirements determination of information systems: User and developer perceptions of factors contributing to misunderstandings**. 2006. 276 f. Tese (Doutorado em Filosofia) - Capella University, United States, Minnesota, 2006.

MENDES, Emilia; MOSLEY, Nile; COUNSELL Steve. The Need for Web Engineering: an Introduction. **Web Engineering**. Berlin, Germany: Springer-Verlag, 1-27 p., 2006.

OLIVEIRA NETTO, Alvim Antônio de. **IHC Interação Humano Computador - Modelagem e Gerência de Interfaces com o Usuário**. 2. ed. Florianópolis: Editora Visual Alta Books, 2006. 120 p.

NIELSEN, Jakob. **Usability 101: Introduction to Usability**. Alertbox. Agosto de 2003. Disponível em: <http://www.useit.com/alertbox/20030825.html>. Acesso em: 16 de maio de 2009.

_____. **Usability Engineering**. 13. ed. San Francisco: Morgan Kaufmann,1994. 376 p.

NIELSEN, Jakob; LORANGER, Hoa. **Usabilidade na Web - Projetando Websites com Qualidade**. Rio de Janeiro: Elsevier - Editora Campus, 2007. 430 p.

PFLEEGER, Shari Lawrence. **Engenharia de Software: teoria e prática**. 2 Ed. Trad. FRANKLIN, Dino. São Paulo: Prentice Hall, 2004. 537 p.

PRATES, Raquel Oliveira; BARBOSA, Simone Diniz Junqueira. Avaliação de Interfaces de Usuário – Conceitos e Métodos. In: IHC2003, 2003, Rio de Janeiro. **Anais da Jornada de Atualização em Informática**, Rio de Janeiro, 2003.

PREECE, Jennifer; ROGERS, Yvonne; SHARP, Helen. **Design de Interação - Além da interação homem-computador**. Trad. POSSAMAI, Viviane. Porto Alegre: Bookman, 2005. 564 p.

PRESSMAN, Roger S. **Engenharia de Software**. 6. ed. Trad. PENTEADO,

Rosângela Delloso. São Paulo: McGraw-Hill, 2006. 751 p.

ROBERTSON, Suzanne; ROBERTSON, James. **Mastering the Requirements Process**. London, UK: Addison-Wesley, 1999. 416 p.

ROCHA, Heloisa Vieira da; BARANAUSKAS, Maria Cecília Calani. **Design e Avaliação de Interfaces Humano-Computador**. Campinas: NIED/UNICAMP, 2003. 244 p.

SCHWABE, Daniel; ROSSI, Gustavo. Developing Hypermedia Applications using OOHDM. In: HYPERTEXT 98, Pittsburgh, 1998, **Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia**, Pittsburgh, PA, USA, June 20-24, 1998. 307 p.

SOMMERVILLE, Ian. **Engenharia de Software**. 8 ed. Trad. MELNIKO, Selma Shimizu, ARAKAKI, Reginaldo e BARBOSA, Edilson. São Paulo, Pearson Addison – Wesley, 2007. 552 p.

TAYLOR, A. (Janeiro de 2000). IT projects: sink or swin?, **The British Computer Society Bulletin**, janeiro de 2000. 24-26p. Disponível em: <http://archive.bcs.org/bulletin/jan00/article1.htm>. Acessado em: 16 de maio de 2008.

WATANABE, Ricardo Hisao; DUDUCHI, Marcelo. Avaliação da Interface de um Sistema Computacional Web do Exército Brasileiro. In: III WORKSHOP DE PÓS-GRADUAÇÃO E PESQUISA DO PROGRAMA DE MESTRADO EM TECNOLOGIA DO CENTRO PAULA SOUZA, São Paulo, 2008. **Anais...** São Paulo, 2008.

WINCKLER, Marco; PIMENTA, Marcelo Soares. Avaliação de Usabilidade de *Sites* Web. In: IHC2001, Florianópolis, 2001, **IV Workshop de Fatores Humanos em Sistemas Computacionais**, Porto Alegre: Sociedade Brasileira de Computação, 2001.

Glossário

Artefato: constitui-se de quaisquer peças de informação tais como modelos, diagramas e/ou documentos produzidos durante o fluxo do processo de desenvolvimento de sistemas.

As Avaliações Formativas (ou Construtivas): são realizadas ao longo de todo o processo de *design*, sempre que os projetistas precisarem compreender melhor o que os usuários desejam e precisam, ou quando precisam verificar se suas idéias atendem as necessidades conhecidas dos usuários.

Avaliação Heurística: é um método de inspeção sistemático, realizada por um grupo de avaliadores, com base em princípios de usabilidade, também conhecidos como heurísticas, com o objetivo de encontrar problemas de usabilidade na interface a fim de serem analisados e corrigidos ao longo de um processo de design.

Avaliações Preditivas: são realizadas por especialistas que aplicam seus conhecimentos acerca dos usuários e de situações típicas de uso para prever problemas de usabilidade.

Avaliações Rápidas: constituem-se em uma prática muito comum durante a concepção de interfaces. São realizadas por meio de reuniões informais entre usuários e desenvolvedores, com o objetivo de se obter apreciações sobre as interfaces e confirmar se as idéias dos desenvolvedores vão ao encontro das necessidades dos usuários.

Avaliações Somativas (ou Conclusivas): são realizadas nas etapas finais de cada ciclo do desenvolvimento ou quando o produto está pronto. Nesta fase são avaliados os protótipos intermediários ou finais da aplicação.

DECIDE: é um acrônimo em inglês formado pelas iniciais das palavras *determine*, *explore*, *choose*, *identify*, *decide*, *evaluate*, que respectivamente significam: determinar, explorar,escolher, identificar, decidir e avaliar.

Diretrizes de Usabilidade: proporcionam uma orientação detalhada sobre as

características de usabilidade em pontos específicos de uma determinada interface e normalmente são acompanhadas de notas explicativas, exemplos e comentários.

Design: abrange a idéia da representação do leiaute gráfico e pode englobar, também, todo o processo no qual se insere a representação ou o desenvolvimento do leiaute.

Designer: profissional que tem conhecimento técnico para a elaboração do leiaute gráfico.

Designs Alternativos: compreende as atividades de buscar e utilizar produtos de terceiros que tenham funcionalidades semelhantes às do software em desenvolvimento para realizar testes empíricos. Dessa forma pode-se analisar como o usuário age ao interagir com um sistema real na realização de tarefas com as quais ele se deparará durante a utilização do sistema.

Engenharia de Usabilidade: é a disciplina que fornece métodos estruturados para alcançar a usabilidade em projetos de interface de usuário durante o desenvolvimento de um produto.

Interação: é o processo que compreende as ações do usuário sobre a interface do sistema e suas interpretações a partir das respostas reveladas pela interface.

Interface: é o nome dado a toda a porção de um sistema com a qual um usuário mantém contato.

Mapa ou Diagrama de Navegação: é uma representação em árvore das páginas e dos *links* entre elas.

Modelo Conceitual: define um processo coerente que possibilita a unificação da identificação e análise dos dados coletados com todas as decisões de detalhes do design, sendo o primeiro passo real na concepção da interface do usuário.

Modelos Prescritivos de Processo: constituem-se de um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalho que são necessários para fazer engenharia de software com qualidade.

Princípios de Usabilidade: constituem-se de aconselhamentos gerais sobre as características de usabilidade de uma determinada interface.

Protótipo: representa uma versão interativa do sistema (desenhos de telas) e pode ser feito de papel e cartolina. Utilizados no decorrer do processo de desenvolvimento de sistemas, os protótipos tornam-se parte do software, pois passam a se parecer com o produto final.

Protótipos de Baixa Fidelidade: não se assemelham muito ao produto final, são simples, baratos e de rápida produção. Por serem facilmente modificáveis, oferecem excelente suporte à exploração de *designs* e idéias alternativas e são particularmente indicados nos primeiros estágios do desenvolvimento.

Protótipos de Alta Fidelidade: oferecem componentes de interface, aparência e comportamento bem parecidos com o futuro sistema, são desenvolvidos por meio da utilização de ferramentas de software.

Requisito: consiste em uma declaração sobre um produto pretendido que especifica o que ele deveria fazer ou como deveria operar.

Storyboards: também chamados de narrativas gráficas são protótipos de baixa fidelidade que são utilizados para representar as interações entre o usuário e o sistema. A representação é feita por uma seqüência de desenhos de esboços de tela e elementos do contexto de uso.

Usabilidade: para a norma brasileira NBR 9241-11, que trata de qualidade de pacotes de softwares bem como das características ergonômicas do produto, a usabilidade é medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso.

Apêndice A

Autorização de Pesquisa Científica

MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DCT - CITEx
3º CENTRO DE TELEMÁTICA DE ÁREA
(CPD 2/1981 e STE/1921)

AUTORIZAÇÃO DE PESQUISA CIENTÍFICA

Eu **SILVIO RENAN PIMENTEL BETAT – Tenente Coronel**, chefe do 3º Centro de Telemática de Área informo que estou ciente e concordo que seja realizada a pesquisa intitulada "**CONSTRUÇÃO DE UM MÉTODO FOCADO EM USABILIDADE PARA APLICAÇÕES WEB ADERENTE AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO 3º CTA**", executada pelo Sr. **RICARDO HISAO WATANABE,** mestrando do **Centro Estadual de Educação Tecnológica Paula Souza**, sob orientação do **Prof. Dr. MARCELO DUDUCHI FEITOSA**.

Para isto autorizo a realização dos trabalhos relacionados à pesquisa na Divisão Técnica, considerando que serão objetos do estudo:

- O Sistema de Alistamento de Serviço Militar (SASM), quanto às questões sistemáticas e problemáticas de apoio ao usuário;
- O Processo de Desenvolvimento de Software (PDS);
- Documentos oficiais que não tenham caráter reservado ou sigiloso.

São Paulo, 21 de maio de 2009.

SILVIO RENAN PIMENTEL BETAT – Ten Cel
Chefe do 3º CTA

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo