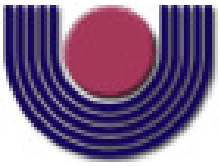


UNIOESTE - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Informática
Curso de Bacharelado em Informática

Estudo sobre Padrões de Interação Humano-Computador para Desenvolvimento de Sistemas com Ênfase em Usabilidade

Everton Pedrolo

CASCAVEL
2008

EVERTON PEDROLO

ESTUDO SOBRE PADRÕES DE INTERAÇÃO HUMANO-COMPUTADOR PARA DESENVOLVIMENTO DE SISTEMAS COM ÊNFASE EM USABILIDADE

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Carlos José Maria Olguín

CASCAVEL
2008

EVERTON PEDROLO

ESTUDO SOBRE PADRÕES DE INTERAÇÃO HUMANO-COMPUTADOR PARA DESENVOLVIMENTO DE SISTEMAS COM ÊNFASE EM USABILIDADE

Monografia apresentada como requisito parcial para obtenção do Título de *Bacharel em Informática*, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Carlos José Maria Olguín (Orientador)
Colegiado de Informática, UNIOESTE

Prof. Ivonei da Silva Freitas
Colegiado de Informática, UNIOESTE

Prof. André Luiz Brun
Colegiado de Informática, UNIOESTE

Cascavel, 02 de dezembro de 2008.

DEDICATÓRIA

Dedico inteiramente este trabalho aos meus pais, ao meu amigo Thiago, amigo eternizado em minhas memórias, e a minha querida Fabiana. Devo tudo isto a vocês.

...when you lose small mind, you free your life...

System of a Down – Aerials

AGRADECIMENTOS

Primeiramente, sem Deus nada é possível. Agradeço pela oportunidade de agraciar o dom da vida, todos os dias de minha vida. Agradeço também pelo apoio de meus amigos que muitas vezes resgataram minhas forças e ajudaram a continuar a luta (não é Darlon?), apesar de todas as dificuldades que passei durante a faculdade. Quero agradecer especialmente a minha namorada Fabiana, que despendeu um tempo substancial preocupada com meus problemas, dando apoiando, acreditando em minha capacidade e esforço. Você compreendeu quando deixava de te ver, quando passava noites em claro, quando chegava a sua casa mais morto que vivo, e fez com que o desafio ficasse menos difícil.

Aos meus pais, que deram todo o suporte, carinho, dedicação, apoio existentes neste mundo. Agradeço do fundo do meu coração e vou sentir-me sempre em débito com vocês.

Deus abençoe o inventor do rock e as variações do metal, pois embalam muitas horas de sono e de batalha na faculdade. *Hail metal*!

Voltando aos agradecimentos, quero fazê-lo aos meus amigos de faculdade que dividem ou dividiram moradia pelos anos de convivência, compartilhando gastos, risadas, sujeiras (no sentido literal da palavra), sofrimento, strogonoff´s (hehe) e muitas vitórias.

Devo agradecer também ao Luis, da Uniocópias, pelos anos de impressões de materiais para provas e mais provas e este trabalho de conclusão de curso. Valeu parceiro!

Finalmente, agradeço aos meus amigos Pedro (vulgo Pedrão), Ricardo (Moscão), Alex, Felipe, Rodrigo, Juliana, Harry e Nílvia pelo companheirismo, compreensão, acolhida e carinho dedicado a mim durante os anos que morei em Palotina e, especialmente, ao Thiago, 16 anos de amizade de plano terrestre, dividindo muitas alegrias, tristezas, pensamentos, histórias, piadas, diversões, sonhos, pesadelos, ambições, filosofias e deixando muita saudade não somente à minha pessoa, mas também a outro amigo especial, Pedrão, na qual formávamos “os três mosqueteiros”, amigos para toda hora (mesmo).

Vocês estarão sempre em meu coração, apesar da distância.

Lista de Figuras

Lista de Abreviaturas e Siglas

ACM	*Association for Computer Machinery*
DEC	*Digital Equipment Corporation*
DOS	*Disk Operating System*
DP	*Design* Participativo
ES	Engenharia de *Software*
GUI	*Graphical user interface*
HCI	*Human-Computer Interaction*
IHC	Interação Humano-Computador
IBM	*International Business Machines Corporation*
ISO	*International Organization for Standardization*
IEC	*International Electrotechnical Comission*
OOPSLA	*Object-Oriented Programming, Systems, Languages, and Applications*
PCU	Projeto Centrado no Usuário
PDA	*Personal Digital Assistant*
PLoP	*Pattern Languages of Programs*
PLML	*Pattern Language Markup Language*
POSA	*Pattern Oriented Software Architecture*
SIGCHI	*Special Interest Group on Computer-Human Interaction*
SO	Sistema Operacional
UI	*User Interface*
XML	*Extensible markup language*

Sumário

Resumo

O projeto de interface com o usuário deve ser essencialmente cuidadoso uma vez que é parte fundamental e indiscutível do processo de desenvolvimento de um *software*. O entendimento do sistema depende de como as funcionalidades do sistema estão organizadas, visando obter maior usabilidade e aplicabilidade. A área de Interação Humano-Computador propõe modelos, técnicas, métodos e ferramentas para o desenvolvimento de sistemas, e o uso de padrões auxiliam projetistas a desenvolver sistemas computacionais, técnica difundida através de Alexander, nos modelos de processo de desenvolvimento de sistemas com interfaces gráficas. Como interfaces necessitam de muitos elementos para realizar a interação, padrões podem estar relacionados com outros, o que justificou a investigação de uma coleção ou linguagem de padrões para Interação Humano-Computador. As linguagens de padrões podem funcionar como *lingua franca* entre desenvolvedores e usuários finais, capturam experiência de projetos prévios e provêm reuso. Como resultados, este trabalho levantou a falta de estudos empíricos sobre a aplicação de linguagens de padrões e padrões no desenvolvimento de sistemas e indicou como exemplo de linguagem de padrões para interface, a linguagem de Tidwell, e como linguagem para interação entre humanos e dispositivos computacionais, a linguagem de Borchers. Quanto à como deve ser organizada essas linguagens e como deve parecer-se, inexiste um formato que seja amplamente aceito entre a comunidade de Interação Humano-Computador.

Palavras-chave: IHC, padrões, interação humano-computador.

Capítulo 1

Introdução

1.1 Objetivo

Este trabalho visou realizar uma investigação de padrões existentes de Interação Humano-Computador, possivelmente uma coleção de padrões ou uma linguagem de padrões, que desenvolvam e melhorem a segurança, eficiência, eficácia e principalmente, a usabilidade da interface, auxiliando o processo de desenvolvimento de sistemas com base nos modelos contidos nesta área multidisciplinar.

Caso, ao final da investigação, não fossem encontrados padrões adequados ao desenvolvimento de sistemas, segundo a Interação Humano-Computador, este trabalho deveria apresentar um ou mais padrões que atendessem os requisitos de usabilidade.

1.2 Justificativa

Segundo recomendações de Sommerville [57], o projeto de interface com o usuário deve ser essencialmente cuidadoso haja vista que é parte fundamental e indiscutível do processo de desenvolvimento de um *software*. A interface deve combinar habilidades, experiências e principalmente corresponder às expectativas do usuário.

A interface geralmente deve seguir aspectos de usabilidade que, de acordo com a norma ISO 9241 parte 11 [35], é "A forma de como um produto pode ser utilizado por usuários específicos para atingir objetivos específicos, com eficácia, eficiência e satisfação num contexto de utilização específico".

Muitos erros e reclamações por parte do cliente se dão pela dificuldade em como usar o sistema em questão, uma vez que o processo tradicional da Engenharia de *Software* não se

preocupa com as necessidades específicas do ambiente de trabalho e dos usuários (Sommerville [57]).

O impacto que o sistema causará está diretamente ligado com a qualidade da informação contida na aplicação que está disponível para seus usuários.

Produzir uma boa interface não é uma simples tarefa e conforme Winckler [72], "A sua construção implica em um aumento da complexidade do gerenciamento da comunicação do usuário com a informação". Baseados em afirmações como estas, engenheiros de *software* começaram a discutir normas para a criação de interfaces, com ênfase em aspectos de usabilidade, ergonomia e estética visual.

De acordo com Campos [17], a interface é responsável por 50% do esforço de desenvolvimento de um *software*. Um bom projeto de interface economiza recursos. Nielsen [44] estima que esta economia chega a US$ 39,000 em projetos pequenos de *software* e a US$ 8,2 milhões em projetos grandes, adicionalmente, um bom projeto evita erros grosseiros, tal como o mal entendimento do sistema pelas várias pessoas envolvidas no processo de desenvolvimento que podem ser engenheiros de *software*, usuários e programadores e a não comunicação entre as partes interessadas.

1.3 Organização do Trabalho

O trabalho está organizado em seis capítulos e quatro anexos.

O Capítulo 2 apresenta o histórico e os conceitos que a área multidisciplinar chamada Interação Humano-Computador preocupa-se em abordar.

No Capítulo 3 apresentam-se padrões, uma introdução sobre como surgiram e a estrutura de padrões em três contextos diferentes: Arquitetura Civil, Engenharia de *Software* e Interação Humano-Computador.

O Capítulo 4 abrange os principais modelos de processo propostos pela Interação Humano-Computador, tais como Engenharia de Usabilidade, Engenharia Cognitiva e *Design* Participativo e o Projeto Centrado no Usuário, e suas características conceituais.

Já o Capítulo 5 deverá apresentar uma discussão sobre a aplicabilidade de padrões propostos por pesquisadores da área de Interação Humano-Computador no modelo de processo Engenharia de usabilidade.

O sexto capítulo mostrará as conclusões sobre a discussão da aplicabilidade dos padrões de Interação Humano-Computador e a análise dos padrões, coleções e linguagens de padrões

encontrados de Interação Humano-Computador no processo de desenvolvimento de sistemas com interfaces gráficas.

O Anexo A contém alguns padrões de Interação Humano-Computador, de autoria de Tidwell.

No Anexo B exemplifica-se um padrão de Alexander, com suas peculiaridades, ou seja, nomenclatura, formatação, contexto e aplicabilidade.

Já o Anexo C contém um exemplo de padrão componente da linguagem de padrões de Interação Humano-Computador do pesquisador Borchers.

Por fim, o Anexo D contém várias linguagens de padrões, com o título correspondente, fonte, número de padrões que compõem esta linguagem e o ano de publicação.

Capítulo 2

Interação Humano-Computador

2.1 Considerações Iniciais

A natureza da computação está em constante mutação. O primeiro computador eletrônico digital, ENIAC, concluído em 1946, foi construído para resolver problemas militares, tal como o cálculo de trajetórias balísticas. Entre 1950 e 1970 assistiu-se a uma grande expansão nos usos militares e uma grande aplicação dos computadores digitais no comércio e na indústria. No final dos anos 70, os computadores pessoais entraram nos lares, e na década de 1980, desenvolveram-se interfaces mais amigáveis ao usuário. Na década de 1990 viu-se a transformação da Internet num importante meio de comunicação, que culminou com a expansão da *World Wide Web* alcançando um bilhão de pessoas, no mundo todo (Bainbridge [8]).

Durante os últimos vinte anos, nota-se o crescimento na área de Interação Humano-Computador (IHC), que propõe-se a melhorar a comunicação entre humanos e máquinas e prover eficiência no uso do computador.

2.2 Definição de Interação Humano-Computador

Segundo Kumar [39], a Interação Humano-Computador é o estudo e a prática da usabilidade. Trata-se da compreensão e a criação de *software* e outras tecnologias que as pessoas terão interesse em usar, serão capazes de usar, e alcançarão eficiência quando fazê-lo. A IHC herda muitos problemas e soluções da recente história da computação, por exemplo, manipulação de interfaces e instruções de ajuda orientadas às tarefas computacionais.

Uma das definições mais aceita é a fornecida pelo *Special Interest Group on Computer-Human Interaction* da *Association for Computer Machinery* (ACM SIGCHI [1]):

> *A Interação Humano-Computador é uma disciplina relacionada com o projeto, o desenvolvimento e a implantação de sistemas interativos computacionais para uso humano e o estudo dos fenômenos que o cercam.*
>
> ACM SIGCHI

A IHC é uma área multidisciplinar com ênfases diferentes (Figura 2.1). A Ciência da Computação preocupa-se com a engenharia e o *design* das interfaces das aplicações. A psicologia concentra o foco da Interação Humano-Computador nas teorias que envolvem processos cognitivos e a análise do comportamento do usuário. A sociologia e a antropologia consideram as interações entre tecnologias, trabalhos e organizações. Já a área de desenho industrial projeta produtos buscando ergonomia, conhecida nos Estados Unidos como *ergonomics* (ACM SIGCHI [1]).

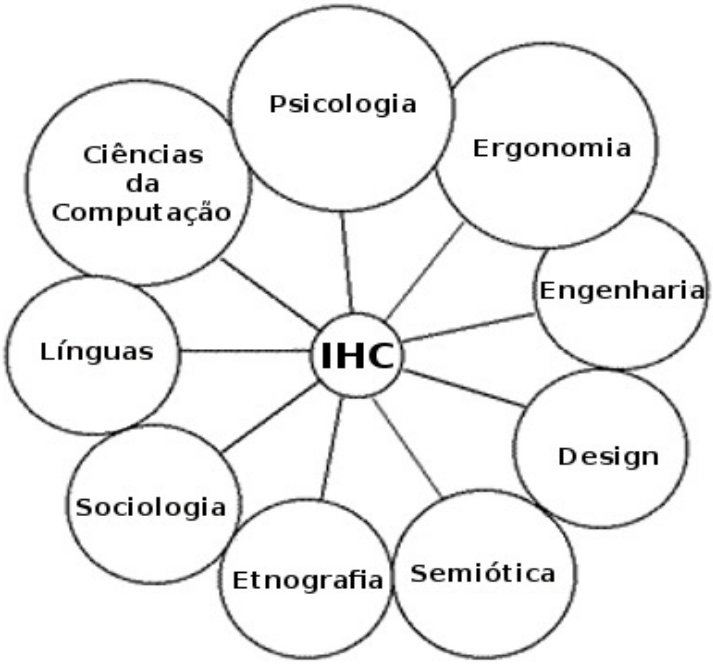


Figura 2.1: Disciplinas envolvidas na Interação Humano-Computador. Adaptada de (Nanni [43]).

Para caracterizar a IHC como um campo de pesquisa, a ACM SIGCHI lista algumas preocupações que são abordadas na área, tais como o desempenho de tarefas comuns entre seres humanos e máquinas, a estrutura de comunicação, a capacidade de aprendizagem no uso de máquinas e o processo de especificação, desenho e implantação das interfaces.

A fim de delimitar o escopo de atuação e especificar as conexões entre as áreas de atuação, a ACM SIGCHI organiza-os em 16 tópicos (Tabela 2.1) que derivam de cinco aspectos inter-relacionados da Interação Humano-Computador (Figura 2.2).

Tabela 2.1: Campos de estudo da Interação Humano-Computador. Adaptado de (ACM SIGCHI [1]).

Área	Sigla	Tópico
(N)atureza da IHC	N1	(Meta-)Modelos de IHC
(U)so e contexto dos Computadores	U1	Organização social e de trabalho
	U2	Áreas de aplicação
	U3	Adaptação Homem-Máquina
Caracteristicas (H)umanas	H1	Processamento de informação humana
	H2	Linguagem, Comunicação, Interação
	H3	Ergonomia
Arquitetura da Interface e sistema (C)omputacional	C1	Dispositivos de Entrada e Saída
	C2	Técnicas de diálogo
	C3	Modos de diálogo
	C4	Computação gráfica
	C5	Arquitetura de diálogo
Processo de (D)esenvolvimento	D1	Abordagens de projeto
	D2	Técnicas de Implementação
	D3	Técnicas de Avaliação
	D4	Estudos de casos e exemplos de sistemas

Existem sistemas computacionais em larga escala no meio organizacional e de trabalho (U1). Dentro deste contexto, existem aplicações para as quais deseja-se empregar sistemas computacionais (U2). Mas o processo de colocar computadores para trabalhar significa que os aspectos dos meios humanos, técnicos, e de trabalho da situação da aplicação devem "encaixar" entre eles através da aprendizagem, permitir que o sistema seja usado de maneiras diferentes, ou outras estratégias (U3).

Para o contexto social e de utilização dos computadores, divide-se a tarefa em dois lados, o computacional e humano.

Ao levar em conta o lado humano precisa-se de informações sobre as características dos usuários, isto é, como as informações são processadas pelas pessoas (H1), como se comunicam (H2) bem como aspectos ergonômicos (H3).

No lado computacional, uma variedade de tecnologias foi desenvolvida para suportar a interação com os seres humanos: dispositivos de entrada e saída (C1), usados em uma série de técnicas para organizar um diálogo (C2), utilizadas para a execução de expressões da interface (C3). Para realizar o suporte ao diálogo, utilizam-se recursos técnicos de computação gráfica (C4). Diálogos complexos levam a considerações necessárias na arquitetura do sistema para

suportar recursos como aplicações interconectadas, janelas, resposta em tempo real, comunicação em rede, interfaces cooperativas e sistemas multiusuário (C5).

Finalmente, o processo de desenvolvimento incorpora o projeto (D1) para os diálogos, técnicas e ferramentas entre computadores e humanos (D2), técnicas para avaliação (D3), e um número de projetos clássicos para estudo (D4). Cada um destes componentes do processo de desenvolvimento está mutuamente relacionado com influência de escolhas feitas em uma área afetando diretamente outras áreas.

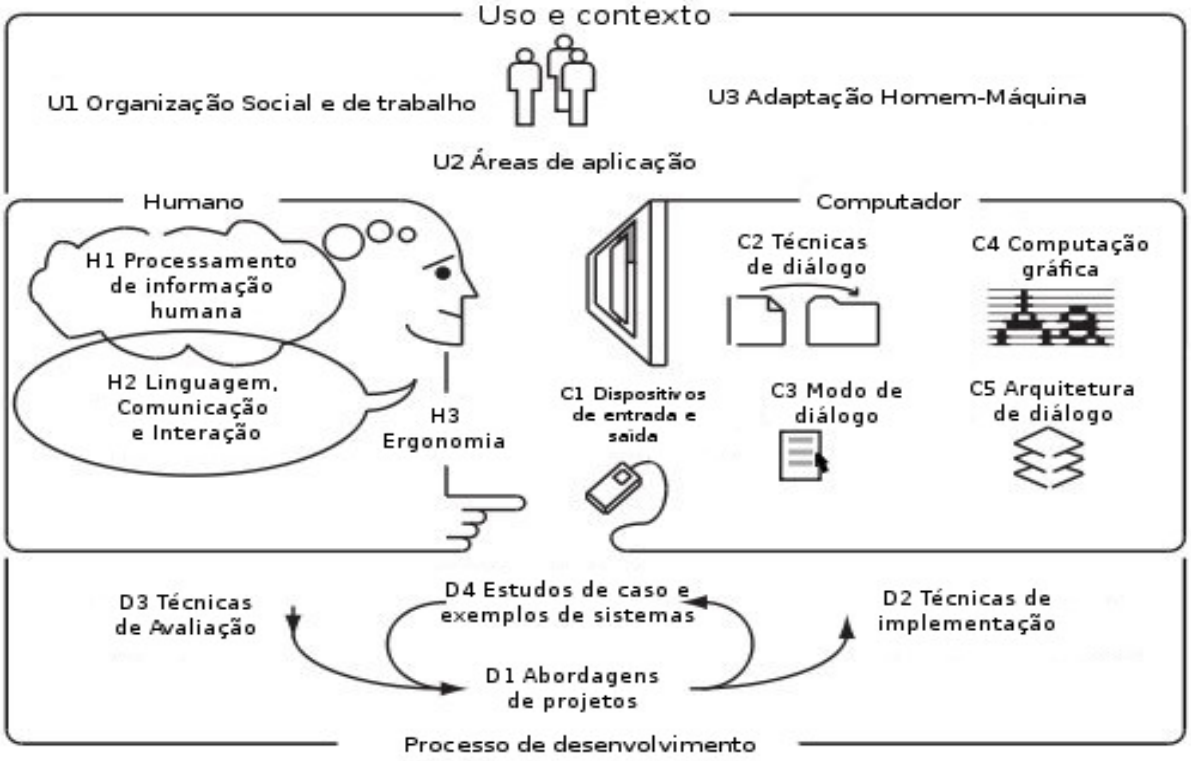


Figura 2.2: Processo de Interação Humano-Computador. Adaptado de (ACM SIGCHI [1]).

Este inventário de itens elencado pela ACM SIGCHI representa os temas atuais da Interação Humano-Computador que possuem resultados concretos e que podem ser ensinados às pessoas interessadas nesta área computacional.

2.3 Usabilidade

A expressão usabilidade começou a ser usada no início da década de 80, principalmente na área de psicologia e ergonomia, como um substituto do termo amigável (*user friendly*), considerado vago até então (Soares [56] *apud* Dias [19]).

Ainda sobre a inadequação sobre o termo amigável, Soares [56] *apud* Dias [19] considera que o usuário não necessita que a máquina seja amigável, apenas não interfira nas tarefas a serem realizadas e reforça que um mesmo sistema, dependendo do usuário, pode ser amigável ou não.

Sabe-se que a usabilidade é a habilidade do *software* em permitir ao usuário final o alcance de suas metas de interação, podendo-se acrescentar que a utilidade é a habilidade do *software* de permitir que o usuário alcance suas metas fundamentais. Portanto as deficiências de usabilidade comprometem a qualidade de um produto, gerando impactos negativos na eficácia, na produtividade, na segurança e na satisfação, todavia a eficiência em usabilidade não significa necessariamente qualidade de uso do produto (Soares [56] *apud* Moraes *et al*. [42], Bitencourt [11]).

A intuitividade e a produtividade existentes hoje em sistemas operacionais (SO) é o resultado de muitos anos de preocupação por parte da indústria de *software* com a interface de seus produtos. Considerado como uma das aplicações mais complexas com relação à usabilidade, os SOs exigiram muito estudo e testes de interface por parte das equipes desenvolvedoras (Soares [56]).

Sair do mundo de interfaces de fundo preto e letras brilhantes (DOS), para chegar à metáfora colorida do *desktop* com utilização de ícones, levou algum tempo e muita pesquisa. Os sistemas operacionais são os precursores de sistemas com interfaces com boa usabilidade, tão necessárias e procuradas hoje em dia, para os sistemas acessados via web (Soares [56]).

Para a interface, daí em diante, ficou a incumbência de seduzir o usuário em potencial e torná-lo cada vez mais próximo dos sistemas informatizados, por meio de comunicação mais intuitiva, metafórica e menos abstrata, que buscou proximidade do sistema cognitivo humano (Soares [56] *apud* Lévy [40]).

Nielsen [44] explora o *design* e propõe princípios que levem a um aumento da usabilidade que, como já citado, é um dos critérios que definem a aceitabilidade de um sistema. Ele explicita os princípios de *design* a partir de quatro *slogans*, que são definidos como "*slogans* de usabilidade", tais como:

a) **o usuário sempre está certo**: a atitude do desenvolvedor quando verifica que tem problemas de interação com um determinado aspecto da interface, não deve julgar que o usuário é ignorante, ou então, que ele não tentou o suficiente. O projetista de interfaces deve ser humilde a ponto de aceitar a necessidade de modificar o que ele achava ser uma "grande idéia" de forma a resolver os problemas dos usuários.

b) **o usuário não está sempre certo**: também não se deve ir ao extremo e construir interfaces somente a partir do que os usuários gostariam. Usuários freqüentemente não sabem o que realmente é bom para eles.

c) **usuários não são desenvolvedores**: uma solução simples para atender a diversidade de usuários seria a de prover interfaces flexíveis que pudessem ser amplamente customizadas, assim cada usuário teria exatamente a interface que melhor lhe satisfizesse. Mas existem alguns bons motivos para isso não ser uma realidade:

i) o processo de customização também vai exigir uma interface e, portanto, adiciona complexidade;

ii) muita customização leva a que cada usuário tenha uma interface diferente de outro e isso dificultaria, por exemplo, o pedido de ajuda de um para o outro;

iii) usuários nem sempre adotam as decisões de *design* mais apropriadas.

d) **desenvolvedores não serão os usuários**: desenvolvedores são diferentes dos usuários comuns em diversos aspectos como, por exemplo, experiência computacional e o conhecimento dos conceitos de *design* do sistema. Isso faz com o que projetista olhe para uma tela ou mensagem e acredite que a mesma está perfeitamente clara e adequada, mesmo que estas sejam incompreensíveis para quem não conhece o sistema. Conhecer o sistema é uma via de mão única, tornando-se impossível o desenvolvedor voltar e fazer o papel de um usuário comum.

Nielsen também cita dez heurísticas de usabilidade, que podem ser consideradas como *Guidelines*, que são recomendações para que o sistema possua o atributo usabilidade (Nielsen [44]):

a) visibilidade do estado do sistema;

b) equivalência entre o sistema e o mundo real;

c) controle do usuário e liberdade;

d) consistência e métricas;

e) prevenção de erros;

f) reconhecimento ao invés de relembrar;

g) flexibilidade e eficiência de uso;

h) estética e *design* minimalista;

i) auxilio ao usuário para reconhecer, diagnosticar e recuperar-se de erros;

j) ajuda e documentação.

A interface geralmente deve seguir aspectos de usabilidade que, de acordo com a norma ISO 9241 parte 11 [35], é "A forma como um produto pode ser utilizado por usuários específicos para atingir objetivos específicos, com eficácia, eficiência e satisfação num contexto de utilização específico".

Além de *Guidelines*, existem outras duas estratégias para que o sistema possa alcançar usabilidade, abordadas na próxima subseção.

2.4 Estratégias de IHC

De acordo com Borchers [15], existem muitas maneiras de transmitir eficientemente experiências de projeto, citando os *Style guides*, *Guidelines* e *Standards*. Embora todas tenham o mesmo objetivo, possuem diferenças entre si. A seguir é feito uma descrição das características de cada abordagem:

2.4.1 *Style guides*

Preece *et al.* [49] explica que um *style guide* consiste em uma coleção de regras de *design* específicas e em princípios dos quais derivam as regras. Estes são utilizados para assegurar uma experiência visual consistente em um conjunto de aplicações, sendo mais amplamente conhecidos os utilizados para o desenvolvimento do Microsoft Windows, e do Apple Macintosh.

Os *Style guides* criados para uma corporação específica podem ser empregados visando proporcionar uma imagem particular à corporação, sendo denominado *Style guides* corporativos ou comerciais (Preece *et al*. [49]).

Uma das primeiras atividades do grupo de desenvolvimento de interface, conforme opinião de Silva [54] é produzir inicialmente um documento com os *Style guides* de interação específicos para um determinado projeto ou conjunto de projetos. O grupo pode basear-se nos padrões de interação com o usuário (se existentes), nos guias de estilo de interação comerciais e, possivelmente, até mesmo nos *Style guides* particulares de outros projetos. Enquanto os *Style guides* comerciais e as diretrizes são aplicáveis numa larga faixa de interfaces com o usuário, os customizados são geralmente utilizados em projetos e organizações específicas.

A maioria dos guias comerciais contém seções com descrições de componentes de interação particulares, tais como, janelas, menus, controles, caixas de diálogo; uso do *mouse* e do teclado; e projeto do *help* e mensagens do sistema. Também contém descrição textual e ilustrações de exemplos de componentes de interação. Alguns exemplos de guias de estilo comerciais disponíveis atualmente no mercado citados por Silva [54]:

a) *Human Interface Guidelines*, APPLE [6];

b) *Common User Access* (CUA™), IBM [34];

c) *OpenLook*™, AT&T/Xerox/Sun [58];

d) *Motif*™, *Open Group* [46].

Guias de estilo customizados, como guias de estilo comerciais, são baseados em vários objetos e estilos de interação que o grupo de projeto considera apropriado para a interface que está sendo desenvolvida. Um guia de estilo customizado deve conter pelo menos as seguintes informações (Silva [54]):

a) Uma introdução que explica o estilo de interação utilizado (por exemplo, manipulação direta, interface multi-janelas), por que este estilo foi escolhido e qualquer outra informação específica da organização ou projeto que influenciou esta escolha (por exemplo, objetivos de usabilidade, restrições da organização e limitações de *hardware* e *software*);

b) Uma seção sobre dispositivos de entrada e saída para o qual o guia de estilo de interação é apropriado;

c) Uma seção com algumas telas representativas do sistema, descrevendo formatos e *layouts*[1];

d) Uma seção para cada objeto de interação e para cada estilo de interação incluído no guia, tais como: janelas, *menus*, botões, linguagens de comando e objetos de manipulação direta. Cada estilo deve ser descrito em termos dizendo o que é, com que se parece (com o máximo de detalhes possível), como o usuário interage com ele e quando ele pode ser usado na interface;

e) Uma seção de mensagens, incluindo erro, informações do sistema, aviso (*warnings*), com possíveis padrões de frases dessas mensagens.

[1] Disposição dos elementos de um projeto gráfico

Muitos desses *Style guides* estão associados a *toolkits* disponíveis comercialmente. O *toolkit* suporta o estilo definido no *style guide* de interação correspondente e, tipicamente, contêm objetos de interação pré-codificados (freqüentemente chamados de *widgets*). Entretanto, alguns cuidados devem ser tomados ao se decidir usar um determinado *style guide* de interação, pois eles não são suficientes, por si só, para garantir a usabilidade de uma interface (Silva [54]).

Para Borchers [15], *Style guides* para interfaces gráficas são úteis para garantir um visual comum entre os diversos componentes, porém como explicitado anteriormente, estão ligados a um *toolkit*, o que pode resultar em inutilização no caso de mudança de ambiente. Como exemplo, cita o caso da mudança do Windows 3.1 para o Windows 95, o que gerou uma dificuldade para aplicar muitos dos *Style guides* existentes na versão anterior, condenando-os a não-utilização. Eles limitam as opções do *design* e podem não ajudar o desenvolvedor em algumas situações particulares da construção de interfaces.

2.4.2 *Guidelines*

De acordo com Preece *et al*. [48] *Guidelines* surgiram de duas áreas diferentes, teorias de psicologia e experiência prática de profissionais especializados em IHC, e seu termo é usado tanto para expressar "*princípios* de projeto" bem como "*regras* de projeto".

Uma regra de projeto é uma instrução que pode ser seguida de acordo com a interpretação do projetista, tomemos como exemplo o campo para informar uma data. Nos Estados Unidos, o formato de data é MM-DD-AA enquanto que outros países seguem o padrão DD-MM-AA (Preece *et al*. [48]).

Princípios de projeto são disponibilizados em um nível alto de abstração, são amplamente aplicáveis e possuem formatos diferentes, o que afeta sua interpretação. Generalizações tais como "usuários vão tratar sistemas computacionais como se este for humano" são baseadas em induções advindas da experiência, situação que exige do profissional que propôs o *guideline* ser um *expert* em IHC (Preece *et al.* [48]).

Borchers [15] classifica *Guidelines* em duas categorias, abstratos e concretos. Os *Guidelines* abstratos são princípios, assim como as oito regras de ouro (Shneiderman [53]) e podem ser aplicados para realizar avaliações de projeto. Estes *Guidelines* não sugestionam construtivamente como resolver um problema de projeto na hora que o profissional depara-se com ele. Também não criam um vocabulário de soluções aplicáveis, e não resolvem problemas de comunicação.

Já *Guidelines* concretos, como o *Macintosh Human Interface Guidelines*, são adaptados para o uso de um conjunto ou *toolkit* de objetos de interfaces. Isto os torna obsoletos rapidamente, a menos que o projetista tente extrair as características do *guideline.*

Guidelines são utilizados para capturar conhecimento de projetos e para ajudar os projetistas a construir interfaces, indicando ações e possibilitando a prevenção de possíveis erros no projeto. Contudo, aplicar estas diretrizes é uma tarefa difícil e problemática (Welie *et al* [70]).

Welie *et al*. [70] listam alguns problemas existentes em *Guidelines*:

a) são muito simples ou muito abstratos;

b) podem ser difíceis de selecionar quais usar;

c) podem ser difíceis de interpretar;

d) podem ser conflitantes;

e) muitos são de autores desconhecidos, o que pode comprometer sua validade.

Um dos *Guidelines* mais famosos é proposto por Ben Shneiderman [53] no livro *Designing the user interface* em 1998, conhecido como "Oito Regras de Ouro":

a) buscar consistência;

b) fornecer atalhos para usuários avançados;

c) oferecer *feedback* informativo;

d) projetar diálogos para encerrar ações do sistema;

e) prevenir e manipular os erros de modo simples;

f) permitir reversão das ações;

g) fornecer controle e iniciativa ao usuário;

h) reduzir a carga sobre a memória de curta duração.

Ferramentas de *software* que contemplam *Guidelines* podem ajudar o projetista a suavizar os problemas de qual estratégia seguir, porém não resolve todos os problemas. Ao invés de utilizar estas ferramentas, Welie *et al*. [70] propõem o uso de padrões de IHC para resolver alguns deles, assunto que será abordado no próximo capítulo.

Guidelines como as Oito Regras de Ouro são úteis para categorizar características ruins existentes em uma interface e interpretar a regra que foi quebrada, porém a falta de exemplos concretos e construtivos, não ajudam o projetista na hora de criar um novo sistema (Borchers [15]).

Padrões explicitamente focam no contexto e dizem ao *designer* quando, como e porque a solução pode ser aplicada. Desta forma, padrões podem ser mais úteis aos projetistas que os *Guidelines* (Welie *et al.* [70]).

2.4.3 *Standards*

Standards normalmente são traduzidos como padrões. Porém, para que não haja confusão entre *Patterns* e *Standards*, neste trabalho a tradução do termo *Standards* é modificada para norma.

O conceito de norma envolve uma maneira prévia de apresentar, falar, desenvolver ou alguma coisa que busque uma coerência entre produtos do mesmo gênero (Preece *et al.* [48]).

Os responsáveis por desenvolver e promover o uso de *Standards* são diferentes organizações, formadas por comitês especializados, que envolvem diferentes profissionais, instituições governamentais, acadêmicos e consumidores, classificadas pelo domínio que pertencem (Preece *et al.* [48]).

As instituições mais proeminentes existentes são a *International Organization for Standardization* (ISO) e a *International Electrotechnical Comission* (IEC). A primeira cobre o ramo mecânico e o segundo é responsável pelas normas no ramo elétrico (Preece *et al.* [48]).

Algumas normas internacionais regem o desenvolvimento de sistemas interativos. São coleções de princípios e regras que fornecem aos *designers* uma estrutura baseada na experiência de outros profissionais. Os mais pertinentes à área são os seguintes (Preece *et al.* [49]):

a) ISO 9241: *Ergonomics of human-system interaction* [35];

b) ISO 13407: *Human-centered Design Processes for Interactive Systems* [36];

c) ISO 14915: *Software ergonomics for multimedia user interfaces* [37].

Normas como a ISO 14915 são um ótimo meio de estabelecer regras em um campo experiente de desenvolvimento. Porém como não possuem referências de exemplos construtivos, e possuem um modo informal de escrita, não são ideais para a aprendizagem de *design* de interfaces (Borchers [15]).

Preece *et al.* [48] comentam que utilizar *Standards* no *design* de interfaces pode trazer as seguintes conseqüências:

a) Terminologia comum;

b) Facilidade de manutenção e evolução;

c) Identidade comum;

d) Redução na necessidade de treinamento;

e) Segurança e saúde.

Como a implementação desta técnica possui estrutura interna e estilo em comum, a manutenção e a possível adição de funções complementares é facilitada, reduzindo o período de treinamento dos usuários, e ainda reduzindo chances de causar *stress*, porque o sistema terá um comportamento esperado a quem utiliza.

Utilizar estes modos alternativos para a comunicação de experiência pode ser muito válido, e devem ser utilizados em conjunto. No próximo capítulo é feita uma discussão ampla sobre padrões, desde a origem histórica e o uso em conjunto com *Style guides*, *Guidelines* e *Standards*.

2.5 Considerações Finais

Humanos interagem com computadores de várias maneiras, e a interface entre humano e computador é crucial para facilitar esta interação. De acordo com Nanni [43], a Interação Humano-Computador é tanto uma arte como uma ciência. Existe uma interdependência da funcionalidade de um sistema e sua interface, portanto deve-se despender algum esforço extra para que a interface consiga suportar as tarefas desejadas pelo usuário.

Ainda não existe uma definição amplamente aceita pelos pesquisadores da área, devido à abrangência de conteúdos relacionados e as diversas áreas que à compõem. A definição mais aceita até agora é a proposta pela ACM SIGCHI, na qual a Interação Humano-Computador é uma disciplina relacionada com o projeto, desenvolvimento e implantação de sistemas interativos para uso humano e todos os fenômenos que o cercam.

A área de IHC possui várias estratégias para que humanos consigam utilizar sistemas computacionais da melhor maneira possível, tal como os padrões de IHC, que são inspirados na idéia de Alexander, assunto abordado no próximo Capítulo.

Capítulo 3

Padrões

3.1 Considerações Iniciais

O estudo de padrões ganhou notoriedade quando Christopher Alexander publicou o livro "*The Timeless Way of Building*" em 1979 (Alexander *et al.*[3]), mostrando várias observações realizadas, conceitos e características na Arquitetura Civil. Seus padrões não tiveram grande sucesso entre os arquitetos, porém a idéia da utilização de padrões foi bem aceita em outras áreas, tais como a Engenharia de *Software* (ES) e a Interação Humano-Computador.

Contudo, não existe uma definição de padrões amplamente aceita entre os pesquisadores da área de Engenharia de *Software* e Interação Humano-Computador. Inicialmente foram descritos como uma "*solução para um problema em um contexto definido"* (Alexander *et al.* [3] *apud* Vlissides [67]).

Vlissides [67] critica essa definição, pois o problema que o padrão propõe-se a resolver deve ocorrer várias vezes, destacando a necessidade da solução ser relevante em situações que não são imediatas, e que um padrão seja uma solução comprovada e eficiente. Apesar de toda a crítica, Vlissides não apresenta uma definição de padrões aceitáveis por alegar ser de difícil entendimento.

Outra definição de padrões, porém com enfoque nos aspectos de IHC, é dada por Borchers [15], que descreve que um padrão de projeto captura uma solução comprovada para um problema de projeto recorrente em uma forma compreensível a humanos. Essa definição também não satisfaz todas as críticas apontadas por Vlissides para a definição dada por Alexander.

De acordo com Rihle e Züllighoven [50] *apud* Appleton [7], um padrão é uma abstração que essencialmente aponta uma solução a um problema recorrente dentro de um contexto específico, mas alerta que a técnica deverá ser apresentada de forma a ser compreendida e

devidamente usada por outras pessoas em seus domínios, exigindo do desenvolvedor de padrões uma grande experiência na área, o que não satisfaz as críticas de Vlissides.

Pode-se notar através dessa breve explanação sobre o conceito de padrões segundo vários pesquisadores, que uma definição amplamente aceita ainda é inexistente. Contudo, existem esforços para que isto ocorra, exemplos são a *HCI International Conference* e a *Pattern Languages of Programs* (PLoP).

Este capítulo contém uma discussão das idéias de Alexander e a influência em diferentes áreas do conhecimento, tal como a Arquitetura Civil, a Engenharia de *Software* e a Interação Humano-Computador.

3.2 Padrões na Arquitetura

A equipe de Christopher Alexander, na década de 60, elaborou um novo método de arquitetura, construção e planejamento baseado em exaustivas observações de como as pessoas interagiam com espaços físicos, procurando assim uma solução para satisfazer parte da população que criticava a arquitetura urbana moderna. Ao estudar a relação entre a forma física do ambiente e os comportamentos sociais, notaram que certos ambientes faziam com que os usuários sentissem-se melhor. Assim, Alexander criou padrões de projeto como forma de soluções para a arquitetura.

Alexander observou que os padrões podem estar relacionados, chamando o agrupamento de padrões de linguagem de padrões. O método descrito em Alexander [3] permite aos usuários destes ambientes participar da construção do projeto através do uso da linguagem de padrões. Os padrões foram utilizados a fim de fornecer um vocabulário aos usuários finais, fazendo com que estes pudessem opinar nas decisões do projeto e, sobretudo, expressar melhor suas idéias.

Visando descrever os padrões propostos, no total 253, Alexander definiu elementos e uma ordem de apresentação destes, escolhidos rigorosamente para fornecer o suporte necessário ao seu entendimento, tanto por parte de um profissional quanto por um leigo. Seguindo a proposta de Alexander, os componentes são:

a) nome: expressa a idéia central do padrão e é usado para referenciá-lo;

b) *ranking*: valor que indica a confiabilidade do autor no padrão. Classificados com asteriscos, variando de zero a dois;

c) ilustração: figura que demonstra a aplicação do padrão. Normalmente uma fotografia de um ambiente que representa um bom exemplo da aplicação do padrão;

d) contexto: explicita quais padrões que este padrão específico auxilia a implementar;

e) descrição curta do problema: apresenta uma visão geral do problema;

f) descrição detalhada do problema: baseada em estudos empíricos. São apresentadas as forças, chamadas de interesses ou tensões de projeto, que podem estar em conflito. Também apresenta soluções que existem no mercado atualmente;

g) solução: este elemento apresenta uma solução geral de todas as soluções apresentadas na *descrição detalhada do problema*. A solução que foi apresentada lida com as forças conflitantes do problema;

h) esboço: fornece uma esquema básico para visualização através de um diagrama, que representa a idéia central do padrão.

i) referências: fornecem nomes de padrões que os pesquisadores recomendam a utilização no decorrer do projeto.

O nome do padrão é apresentado por Alexander em letras maiúsculas. Já o problema e a solução estão em negrito, em parágrafos separados, e os diagramas estão em forma de esboços feitos a mão. O contexto é iniciado por "...", referências são concluídas com "...", a solução vem após a palavra *"Therefore*" em uma linha separada (Anexo B). O objetivo de Alexander ao utilizar este modo de apresentação é fazer com que as palavras repetitivas não distraiam o leitor (Borchers [15]).

Uma das características dos padrões de Alexander é o nível de abstração em que foram escritos, para não se tornarem muito abstratos, o que exigiria o redescobrimento de como aplicá-los, e tampouco específicos demais, o que impediria a diversificação de aplicações.

Isto permitiu aos leigos o entendimento de padrões; fato esse que não foi bem aceito por outros arquitetos, que alegam que o poder exclusivo de criação de ambiente foi retirado das mãos dos profissionais com a criação dos padrões. O sucesso dos padrões foi grande entre os amadores, fundamentalmente àqueles que desejavam remodelar seus ambientes (Borchers [15]).

3.3 Padrões da Engenharia de *Software*

A área de Engenharia de *Software*, observando a aplicação de padrões na arquitetura, resolveu buscar uma solução direcionada ao seu domínio, e em 1987 na conferência *Object-Oriented Programming, Systems, Languages, and Applications* (OOPSLA), os autores Beck e Cunningham propuseram o uso de padrões, buscando uma metodologia apropriada para a programação orientada a objetos, pois a análise estrutural, a programação estruturada e os modelos entidade-relacionamento mostravam-se ineficientes para esta nova abordagem.

Beck e Cunningham [10] documentaram 6 padrões, voltados a SmallTalk, para a construção de interfaces. O resultado repercutiu em vários pesquisadores da Engenharia de *Software*, o que motivou a criação de *workshops*, tal como o *Pattern Languages of Programs* (PLoP), que tinha o objetivo de apresentar, trocar e refinar os padrões e as linguagens de padrões, sendo realizado pela primeira vez em 1994, na Universidade de Illinois.

No ano de 1995, os pesquisadores Gamma, Helm, Johnson e Vlissides publicaram um livro contendo uma coleção de padrões para a abordagem de *Software* Orientado a Objetos, chamado *Design Patterns: Elements of Reusable Object-Oriented Software*, e ficaram conhecidos como *Gang of Four* (Gamma *et al.* [28]). Observando os componentes do padrão de arquitetura (Seção 3.2) nota-se uma diferenciação, que objetiva um melhor entendimento dos padrões em si. Os componentes são:

a) nome: expressa a idéia central do padrão e serve como referência;

b) classificação: os padrões da coleção de Gamma *et al.* [28] são divididos em três categorias que variam de acordo com o seu propósito:

i) criação**:** padrões que preocupam-se com o processo de criação do objeto;

ii) estrutura: padrões voltados à composição de classes e objetos;

iii) comportamento**:** caracterizam meios em que classes e objetos interagem e distribuem responsabilidades.

c) intenção: parágrafo que descreve o propósito do padrão, raciocínio básico e intenção, e a qual problema ou assunto particular de projeto o padrão relaciona-se;

d) também conhecido como: apresenta nomes alternativos pelos quais o padrão é conhecido (apelido(s));

e) motivação: apresenta um possível cenário ilustrando um problema de projeto e como classes e objetos estruturados no padrão podem resolver o problema;

f) aplicabilidade: dá informações sobre quais situações é recomendando aplicar o padrão, maus exemplos de projeto, e como reconhecer estas situações;

g) estrutura: apresenta graficamente uma representação das classes do padrão;

h) participantes: descreve objetos e classes que fazem parte do padrão de projeto e responsabilidades destes;

i) colaborações: demonstra as responsabilidades da colaboração dos *participantes*;

j) conseqüências: descreve os resultados do uso do padrão, e quais aspectos da estrutura do sistema é permitido modificações;

k) implementação: mostra dificuldades e técnicas exigidas para implementar o padrão, como também características específicas de linguagem;

l) exemplo de código: apresentada-se pedaços de código que demonstram como implementar o padrão usando a linguagem C++ e/ou *SmallTalk*;

m) usos conhecidos: cita exemplos conhecidos de implementação do padrão correspondente;

n) padrões relacionados: relaciona padrões adicionais usados na implementação, bem como diferenças entre eles e outros padrões que poderiam ser utilizados.

Diferentemente de Alexander, que projetou os padrões compreensíveis e utilizáveis por leigos, Gamma *et al.* tentaram manter a linguagem simples e direcionaram para o uso desta técnica por profissionais (Fincher [25]).

O livro *Design Patterns: Elements of Reusable Object-Oriented Software* é considerado um marco no uso de padrões na ES, pois possibilita a utilização dos padrões para transferir conhecimentos de desenvolvedores com muita experiência em construção de projetos para desenvolvedores com pouca experiência.

3.4 Padrões de Interação Humano-Computador

A pesquisa com padrões de IHC começou devido ao sucesso da aplicação de padrões para a Engenharia de *Software*. Os principais pesquisadores da área são Jennifer Tidwell, Jan Borchers e Martijn van Welie, por trazerem avanços significativos para a Interação Humano-Computador (Fincher [25]).

Primeiramente, deve-se fazer a distinção entre uma coleção e uma linguagem de padrões. Para Sinnig [55] uma coleção de padrões pode ser vista como um dicionário ou catálogo de padrões, sem regras de interação ou hierarquia entre os padrões. Já uma linguagem de padrão contém informação dizendo quando e de que maneira padrões podem ser combinados.

Segundo Sinnig [55] uma linguagem de padrões contém estruturas hierárquicas. Conexões entre padrões existem no mesmo nível e diferentes níveis de hierarquia. Como resultado, linguagens de padrões são estruturadas em dimensões horizontais e verticais. A dimensão horizontal captura a estrutura hierárquica da linguagem, enquanto a dimensão vertical captura inter-relações de padrões do mesmo nível hierárquico.

> *Os objetivos de uma linguagem de padrões de IHC são compartilhar soluções de sucesso entre profissionais de IHC, e prover uma lingua franca*[2] *de projeto de IHC para qualquer envolvido no projeto, desenvolvimento, avaliação ou uso de sistemas interativos.*
>
> Borchers [13]

O trabalho de Tidwell possui o maior destaque, devido ao seu pioneirismo de pesquisa e ao fato da linguagem de padrões proposta ser a maior em escala e influência (Fincher [25]). Tidwell apresentou vários padrões voltados especificamente para a implementação de interfaces com o usuário, e estes são organizados em uma linguagem (Tidwell [59] e [60]) e em uma coleção de padrões (Tidwell [63]). Tidwell apresenta padrões para projeto de interação entre humanos e qualquer tipo de artefato em sua linguagem de padrões, nomeada *Common Ground*.

Baseada em sua experiência com a escrita e uso de padrões, Tidwell [62] acredita que exemplos, aplicabilidade e evidência são as características mais importantes para quem está planejando implementar padrões em seus sistemas. Mesmo que o trabalho de Tidwell seja o maior em escala, ele não está finalizado, pois vários padrões que foram identificados estão incompletos (Fincher [25]).

Os padrões tanto de ES quanto de IHC de Jan Borchers preocupam-se em como representar o modelo mental do usuário e tornar a interação mais atrativa. No livro *A Pattern Approach to Interaction Design* o autor apresenta um *framework* que, através da utilização de padrões de

[2] Linguagem comum para pessoas de diversos domínios de conhecimento.

arquitetura de *software*, de projeto de interação e do domínio da aplicação de um projeto, proporciona uma melhor comunicação entre as equipes de projeto.

Welie [71] apresenta em seus trabalhos diversos padrões para a *Web*, para interfaces GUI (*Grafical User Interface*) e para interfaces de sistemas móveis (celulares, *smartphones*, PDA´s). Além de identificar padrões, Welie também realiza pesquisas relacionadas à classificação de padrões de IHC e ao uso de ferramentas para apoiar a escrita e o uso de padrões.

Há esforços da comunidade de desenvolvimento de padrões para aproximar a qualidade do trabalho que está sendo realizado com os padrões na área de IHC com a qualidade do trabalho de Alexander na Arquitetura (Borchers [15]).

A realização de workshops de padrões de IHC tem demonstrado a preocupação em criar ferramentas que ofereçam apoio à identificação, escrita e uso de padrões e também à criação de uma *lingua franca*. Espera-se que as atividades de escrita e uso de padrões sejam potencialmente facilitadas através do uso de ferramentas (Fincher *et al.* [27]).

A adoção de uma *lingua franca* é justificada pela necessidade de interação entre os desenvolvedores, usuários e outros participantes do projeto. Segundo Erickson [22] a utilização de outro meio, salvo padrões de interação, para a comunicação entre os diferentes envolvidos (Figura 3.1) resulta em atraso de projeto e dificuldades de interação.

Figura 3.1: Diferentes participantes em um projeto de *software*. Adaptado de (Erickson [22]).

Um workshop que discutiu sobre quais componentes os padrões de IHC devem ter foi o CHI 2003, e os resultados desta discussão apresentados por Fincher *et al.* [27]. Segundo estes autores, a utilização de diferentes formatos, exemplificado na figura 3.2, torna difícil o entendimento do escopo e do potencial do formato por parte de um usuário inexperiente. Fornecer múltiplas interpretações oculta as forças conceituais devido à preocupação com detalhes de baixo nível e pouca consideração com a essência de um padrão e uma linguagem de padrões.

Tidwell (2006)	Van Duyne et al. (2003)	Van Welie	Yahoo!	Casaday (1997)	Borchers (2001)	Dearden et al. (2002)	Chung et al. (2004)
Nome	Nome	Título	Título		Título	Título	Título
	Background						Background
Descrição	Problema	Problema	Sumário do problema	Problema	Problema	Problema	Problema
Exemplo	Exemplo	Exemplo	Exemplo		Exemplos existentes	Ilustração	Exemplo
Porquê		Porquê	Razão	Forças			
Usar quando		Usar quando		Contexto			
	Solução	Solução	Solução	Solução	Solução geral	Solução	Solução
Como		Como					
Diagrama		Diagrama				Diagrama	
	Considera outras partes				Referências	Referências	
		Implementação					
		Literatura					Referências a literatura
			Acessibilidade				

Figura 3.2: Comparativo de elementos de Padrões de IHC. Traduzido de (Wania [68]).

Um dos resultados obtidos após o workshop CHI 2003 foi a criação de uma especificação para estas linguagem de padrões, e foi chamada de *Pattern Language Markup Language* - PLML, escrita na linguagem XML. Segundo Fincher *et al.* [23], a meta é organizar as diferentes estruturas de padrões existentes. A inconsistência, tanto por estruturas diferentes entre áreas dissimilares de conhecimento quanto por área igual, torna-se um problema quando deseja-se utilizar padrões de estruturas diferentes e referenciá-los.

A PLML foi especificada com o propósito de englobar as várias estruturas de padrões existentes. Os elementos componentes da PLML são:

a) *pattern id*: identificador do padrão na coleção correspondente;

b) nome: nome adotado para referência do padrão;

c) *alias*: nomes alternativos para o padrão (apelido(s));

d) ilustração: exemplo de aplicação real do padrão;

e) problema: explicita o conflito a qual o padrão propõe-se a resolver;

f) contexto: caracteriza possíveis situações para aplicações do padrão;

g) forças: apresenta-se as possíveis tensões de projeto envolvidas no problema;

h) solução: apresenta possível solução do problema, podendo ser uma generalização dos exemplos implementados em outros sistemas;

i) sinopse: sumário do padrão;

j) diagrama: esquema graficamente a *solução* do padrão;

k) evidências: contém exemplos da instanciação do padrão. Composto por dois elementos:

i) exemplo: abrange usos conhecidos;

ii) lógica: demonstra o raciocínio utilizado para aplicação do padrão;

l) confiabilidade: apresenta, de forma quantitativa, a confiabilidade do autor no padrão;

m) literatura: referencia outros trabalhos, se houver a necessidade;

n) implementação: possui detalhes técnicos de implementação, código ou fragmento(s) de código(s);

o) padrões relacionados: referencia outros padrões.

A PLML possui ainda componentes que indicam autoria do padrão, gerenciamento de mudanças e referencia a outros padrões.

A linguagem de padrões de Tidwell [60] consiste de mais de 50 padrões e mais alguns que não foram detalhados ainda. Os padrões são agrupados conforme o domínio do problema com vários níveis de detalhes, e a maioria consta de instruções de como implementar e como utilizá-lo da melhor maneira possível.

Borchers [15] considera a coleção de padrões de Tidwell o esforço mais promissor em criar uma linguagem de padrões de IHC. Enquanto possui poucos pontos fracos, tal como alguns

padrões ainda não estão completos, ela pode servir como exemplo de como uma linguagem de padrões de IHC deve-se parecer.

3.5 Considerações Finais

Assim como na Engenharia de *Software*, não existe uma classificação de padrões de Interação Humano-Computador aceita e difundida. Há indícios que existam dois tipos de padrões em IHC (Alpert [4]): padrões de interação e padrões de interface com o usuário.

Os padrões de interação estão relacionados com preocupações de alto nível, e algumas vezes com *Guidelines*, envolvendo a psicologia do usuário. Já os padrões de interface com o usuário são mais relacionados a problemas de interação específicos e sua solução é baseada em componentes de interface com o usuário.

Os padrões podem ser agrupados com finalidade de melhorar o projeto de um sistema, porém há dois modos de fazer esse agrupamento, que pode ser feito através de: coleções e linguagens de padrões.

Há uma grande diferença entre coleções de padrões e linguagens de padrões. Uma linguagem de padrão contém uma coleção de padrões, contudo indicando como os padrões podem ser usados em conjunto, e explicitando a hierarquia entre os padrões, bem como a inter-relação entre eles.

Como é uma área de pesquisa relativamente nova, não existe ainda uma definição amplamente aceita de como uma linguagem de padrões de IHC deve-se parecer. Para tentar resolver este problema, no CHI 2003 foi proposta a criação da linguagem de marcação PLML, que parece ser promissora, porém ainda não foi adotada entre os pesquisadores, muitas vezes por não conseguir abranger todos os elementos componentes de padrões de domínios diferentes.

De acordo com Borchers [15], várias estratégias de IHC possuem o propósito de comunicar efetivamente experiências de projeto, assim como os padrões. Exemplos mais usuais são os *Style guides*, os *Guidelines* e os *Standards*.

Capítulo 4

Desenvolvimento de Sistemas com Base na Interação Humano-Computador

4.1 Considerações Iniciais

Nos primórdios da computação os sistemas de *software* não eram desenvolvidos para serem interativos. Com o advento dos computadores pessoais no final dos anos 70, os sistemas começaram a ser construídos visando uma maior integração com o homem. Toda esta busca pela interatividade impactou no aumento da porcentagem do esforço e investimento destinado para o desenvolvimento de interfaces com o usuário na produção de um sistema.

Além do aumento do custo do *software*, a exigência de requisitos por parte dos clientes levou os especialistas a desenvolver novos modelos de processo para entender, estruturar e melhorar o processo de desenvolvimento de sistemas interativos.

A área de Interação Humano-Computador visa oferecer suporte ao desenvolvimento desse tipo de *software*, considerando a perspectiva do usuário final, pois os modelos de processo da ES não estimulam o uso de notações e técnicas para captura dessa perspectiva (DIX *et al.* [20]).

Este Capítulo aborda modelos de processo utilizados por profissionais da área de IHC para o desenvolvimento de sistemas e as possíveis utilizações de padrões nesses modelos. Entende-se por modelo de processo as várias abordagens encontradas na literatura para desenvolvimento de sistemas interativos de acordo com a Interação Humano-Computador.

As abordagens apresentadas estão divididas em 3 grandes ramos: Projeto Centrado no Usuário (PCU), *Design* Participativo (DP) e Cognitivismo (Brown [16]).

4.2 Modelos de Processo Propostos pela IHC

O Projeto Centrado no Usuário diferencia-se dos modelos de processo da ES por basear-se no *feedback* do cliente para produzir o artefato, considerando sua visão para construir a interação, e não a visão que o projetista tem do sistema. Enquanto o PCU produz sistemas **para** o usuário, o *Design* Participativo produz o sistema **com** o usuário e a Engenharia Cognitiva enfatiza o entendimento do sistema por parte dos mesmos para produzi-lo.

4.2.1 Projeto Centrado no Usuário

A *International Standards Organization* (ISO) divulgou a norma 13407 em 1999, a *Human-Centred Design Processes for Interactive Systems*, que fornece direções para o modelo de processo PCU (Jokela *et al.* [38] *apud* ISO/IEC [36]). Esse modelo enfatiza a necessidade de desenvolver *software* mais usáveis, enfocando a perspectiva do usuário no processo de desenvolvimento, através de uma visão geral para o planejamento e gerenciamento de projetos centrados no usuário, sem detalhar métodos e técnicas a serem aplicados (Jokela *et al.* [38]).

Alguns princípios descritos pela norma (Gulliksen *et al.* [32] *apud* ISO/IEC [36]) são a alocação apropriada de funções entre o usuário e o sistema, o envolvimento ativo do usuário, a iteração de soluções de projeto e times de projeto multidisciplinares, e define quatro atividades essenciais para o PCU, apresentadas na Figura 4.1 (Jokela *et al.*[38]).

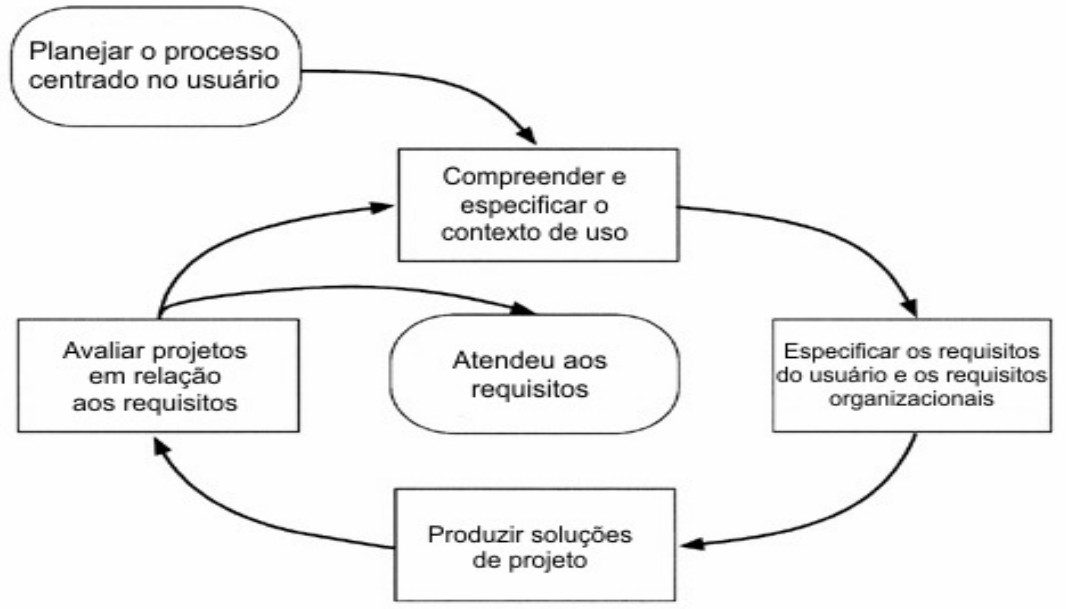


Figura 4.1: Atividades do Projeto Centrado no Usuário. Adaptada de (ISO/EIC [36]).

A atividade *Entendimento e Especificação do Contexto do Usuário* possui o objetivo de capturar informações sobre as características dos usuários, as tarefas que realizam e o ambiente no qual utilizarão o programa, servindo como base para as atividades de avaliações posteriores. Na atividade *Especificação dos Requisitos do Usuário e da Organização* serão especificados os requisitos do usuário e da organização, além de uma classificação dos requisitos por ordem de prioridade e o fornecimento de um *benchmark*[3] para testar o projeto.

Já a *Produção do Projeto e Protótipos do Sistema* explora possíveis soluções de projeto, transformando-as em protótipos. As primeiras soluções de projeto podem basear-se em experiências anteriores ou utilizar *Standards* e *Style guides*, aperfeiçoados através de *feedback* do usuário. A última atividade da iteração, *Execução da Avaliação Baseada em Usuário do Sistema ou Protótipo*, tem como objetivo confirmar o nível em que os objetivos da organização e dos usuários foram alcançados, fornecendo também informações para o refinamento do projeto. Esses objetivos são atingidos através da realização e análise de testes com usuários, ou com especialistas em testes de usabilidade, documentando os resultados obtidos e realizando as mudanças necessárias.

O modelo de processo PCU é divido em três etapas, conforme apresentado na Figura 4.2. Segundo Preece [48], na etapa de Projeto são coletadas e sintetizadas informações sobre as necessidades e capacidades do usuário. Essas informações podem ser obtidas através de técnicas como análise de requisitos (objetiva compreender o que o sistema poderia fazer), análise de tarefas (permite compreender como poderiam ser realizadas as tarefas) e testes de usabilidade (define o desempenho aceitável do sistema).

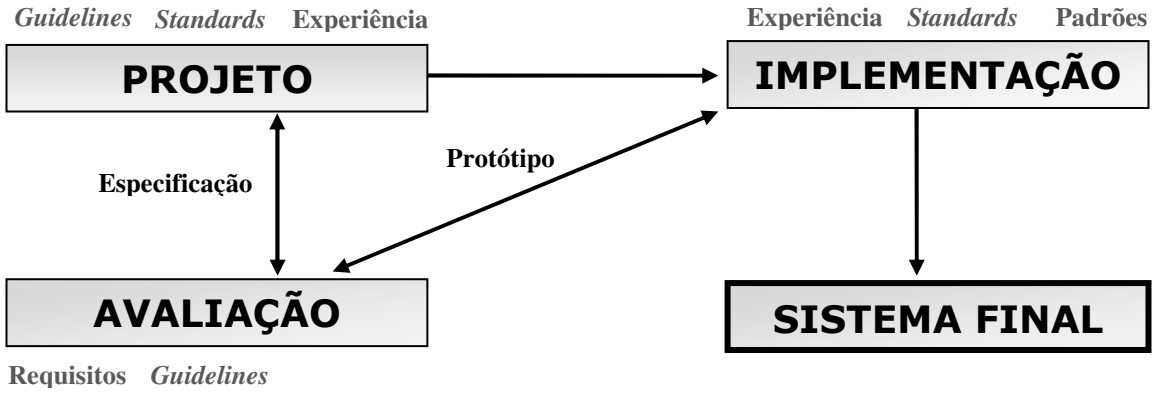


Figura 4.2: Etapas do processo do Projeto Centrado no Usuário. Adaptada de Preece *et al.*[48].

[3] *Benchmark*: Possibilita a comparação de algum item através de um teste com um padrão pré-estabelecido.

Com estas informações a equipe de projeto desenvolve um modelo conceitual do sistema com o auxílio de ferramentas como *Standards* e *Guidelines*. Esse modelo conceitual geralmente são maquetes do sistema que permitem serem testadas com os usuários. A próxima etapa, implementação, na qual o modelo conceitual gerará um esquema para a produção de um protótipo ou o sistema é analisado posteriormente. Durante a avaliação serão coletados dados que dirão se o sistema possui a usabilidade esperada ou se será necessário projetar novamente o sistema.

4.2.2 *Design* Participativo

Originário da Escandinávia, do início da década de 70, este modelo de processo tem como principal característica incorporar o usuário final como um membro do time de projeto, permitindo-lhe participar de todas as atividades de desenvolvimento, não somente da etapa de testes e avaliação. A participação do usuário deve ser ativa, não devendo ser passiva e submetida à vontade do projetista (Dix *et al*. [20]).

Houve duas influências no surgimento deste trabalho (Preece *et al.* [49]):

a) O desejo de poder comunicar informações sobre sistemas complexos;

b) O movimento de sindicatos trabalhistas reivindicando que os trabalhadores tivessem um controle democrático das mudanças em seu trabalho.

O motivo da inclusão do usuário no processo de desenvolvimento deve-se ao seu conhecimento do contexto do trabalho, de forma que através de sua participação ativa, traga contribuições efetivas em todas as fases do processo de desenvolvimento resultando num projeto que reflita as perspectivas e necessidades do usuário (Rocha; Baranauskas [51]).

Outro motivo para a inclusão do usuário no processo de desenvolvimento é que, através da inclusão de um novo sistema de *software*, o processo organizacional e o contexto de trabalho serão alterados. Por meio da participação deste, seria possível que estas mudanças fossem mais aceitas, ou seja, aumentaria a aceitabilidade do sistema.

Envolver usuários em decisões de *design* não é uma tarefa simples. As diferenças culturais podem ser agudas quando usuários e *design*ers são solicitados a trabalharem juntos, a fim de realizarem a especificação de um sistema (Preece *et al.* [49]).

Dix *et al.* [20] descrevem três características específicas do modelo *Design* Participativo (DP):

a) objetiva melhorar as tarefas e o ambiente de trabalho através da introdução do usuário final na equipe, fazendo com que o projeto e a avaliação orientada ao trabalho ou contexto seja melhor do que a orientada aos sistemas;

b) é categorizado pela colaboração, ou seja, através da inclusão do usuário na equipe de desenvolvimento, e pode contribuir em todos os estágios;

c) é uma abordagem iterativa na qual o projeto é sujeito à avaliação e revisão em cada estágio de desenvolvimento.

Segundo Brown [16], este modelo de processo não defende uma metodologia em particular, justificando que, para a produção de um bom produto de *software*, não é necessária uma metodologia como ponto central, somente enfatizar a qualidade da comunicação entre o usuário e o projetista. Nesse modelo de processo são definidos os tipos de comunicação desejados e os métodos que podem auxiliar usuários e projetistas a realizarem um projeto melhor.

4.2.3 Engenharia Cognitiva

Definido por Norman e Draper [45] em 1986, este modelo de processo prioriza o entendimento e a modelagem de uma atividade na forma como ela é entendida pelo usuário. Obtendo o modelo correto, é possível usá-lo para criar um projeto que será intuitivo ao usuário (Brown [16]).

As metas principais estão focadas em entender os princípios fundamentais da ação humana e desempenho que são relevantes à engenharia do projeto e criar sistemas agradáveis de usar, possibilitando engajar na utilização do *software* (Rocha e Baranauskas [51]). Logo, nesse modelo de processo, o projetista está interessado em entender o porquê dos usuários se comportarem de tal modo ou o motivo de um projeto apresentar-se melhor do que outro (Brown [16]).

Considerando a realização de tarefas complexas por usuários não experientes como uma atividade de resolução de problemas, a Engenharia Cognitiva visa facilitar a realização da tarefa através de um bom modelo conceitual do sistema físico (Rocha e Baranauskas [51]).

Para o entendimento de como as pessoas realizam coisas, Norman e Draper [45] apresentam uma teoria aproximada à teoria da ação, pois não existe ainda uma formalização desta. Nessa teoria aproximada, uma pessoa interage com o sistema, no caso de IHC, um computador. Assim, temos duas entidades que desejam interagir, no entanto as metas da

pessoa são representadas em termos relevantes à ela, ou seja, em termos psicológicos, e os mecanismos e estados do sistema são expressos em termos relativos à ele, ou seja, em termos físicos. A impossibilidade atual de mapeamento de termos psicológicos para mecanismos físicos, devido à diferenciação de forma e conteúdo, causa algumas discrepâncias chamadas de *Golfo da Execução* e *Golfo da Avaliação*, conforme Figura 4.3. Um projeto, seguindo as diretrizes da Engenharia Cognitiva, tenta minimizar o esforço utilizado para percorrer estes dois golfos através de um bom modelo conceitual do sistema.

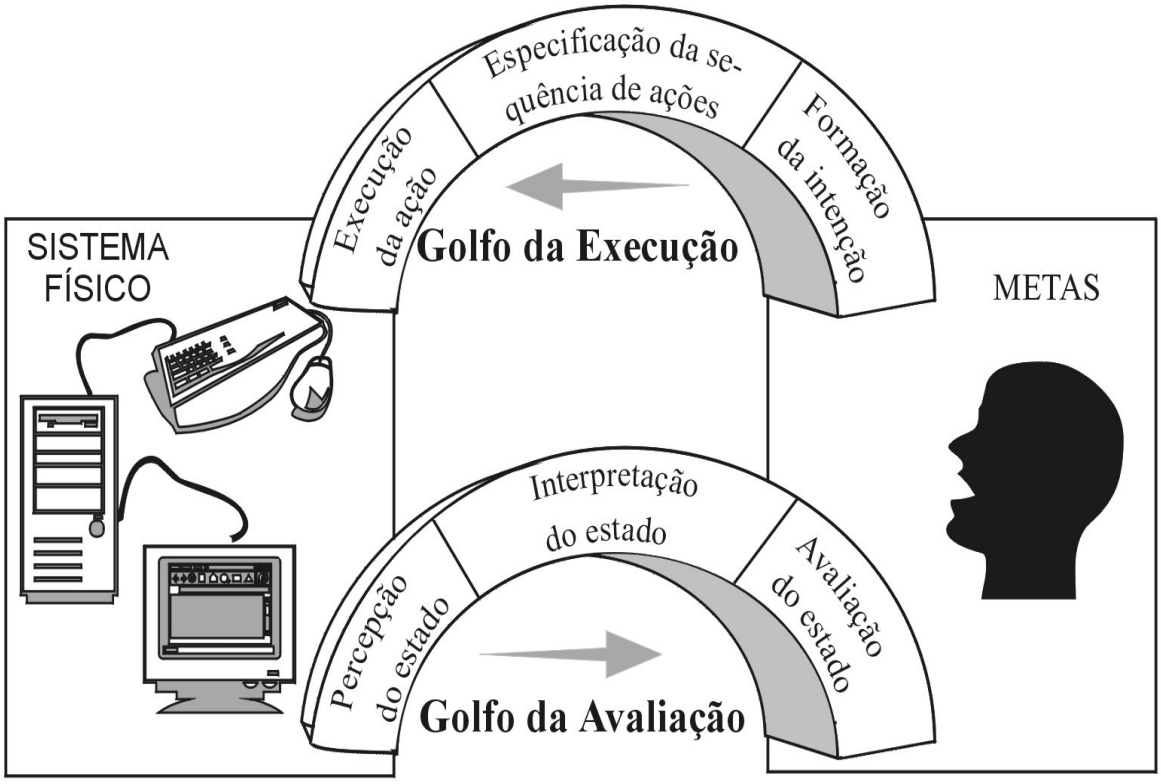


Figura 4.3: Golfo da Execução e Golfo da Avaliação. Adaptada de Norman e Draper [45]

O *Golfo da Execução* é um *gap* que vai das metas do usuário para o sistema físico. Na teoria aproximada apresentada por Norman e Draper, existe uma ponte, dividida em três segmentos, que conecta estes dois elementos. Essa ponte ocorre quando o usuário tenta utilizar um sistema. A pessoa possui metas expressadas em termos psicológicos, enquanto o sistema apresenta o seu estado em termos físicos.

A avaliação requer uma comparação entre a interpretação do estado do sistema com as metas e intenções originais. Devido à interpretação, encontram-se problemas na determinação do estado atual do sistema na comparação com a meta desejada. Embora dispositivos de saída sejam utilizados para facilitar a obtenção do estado atual do sistema, existe um *gap* entre os

termos físicos do sistema e os termos psicológicos do usuário. Esse *gap* é o *Golfo da Avaliação*.

Um projetista deve analisar as etapas do processo de interação usuário-sistema para tentar abreviar esses golfos, a fim de reduzir os problemas que ocorrem durante a interação (Silva [54]).

O projetista pode reduzir o golfo de execução através de uma análise cuidadosa das tarefas dos usuários e da escolha de elementos de entrada e acionamento de comandos compatíveis com estas tarefas, respeitando-se às limitações físicas e cognitivas dos usuários (Silva [54]).

Para abreviar o golfo de avaliação, o projetista pode alterar as características dos elementos de saída do sistema e as mensagens de *feedback* (Silva [54]).

Quando o projetista falha em abreviar os golfos de execução e avaliação, cabe aos usuários todo o esforço para atravessá-los. Eles têm que alterar a maneira como pensam e realizam suas tarefas para se adequar ao modo como o sistema requer que isto seja feito (Silva [54]).

4.2.4 Engenharia da Usabilidade

Engenharia de Usabilidade é o termo que se usa para definir o processo de projeto de sistemas computacionais que objetivam a facilidade de aprendizado, de uso, e que sejam agradáveis para as pessoas (Rocha e Baranauskas [51]).

O processo de projeto para usabilidade foi inicialmente uma recomendação de vários pesquisadores independentes (Gould e Lewis [30], Nielsen [44]) e grupos de pesquisa na *Digital Equipment Corporation* (DEC) e *International Business Machines Corporation* (IBM) em Engenharia de Usabilidade que, já na década de 80, constataram que confiar na experiência do *designer* e em padrões, *Guidelines* ou em várias filosofias de projeto racionais e analíticas não era suficiente para chegar a bons sistemas de computador (Rocha e Baranauskas [51]).

A Engenharia de Usabilidade propõe a aplicação de métodos empíricos ao projeto de sistemas baseados no computador (Rocha e Baranauskas [51]).

Nielsen [44] apresenta o modelo de processo da Engenharia de Usabilidade composto por 11 atividades, chamadas de estágios, separados em três fases: pré-projeto, projeto e pós-projeto, conforme apresentado na Figura 4.4.

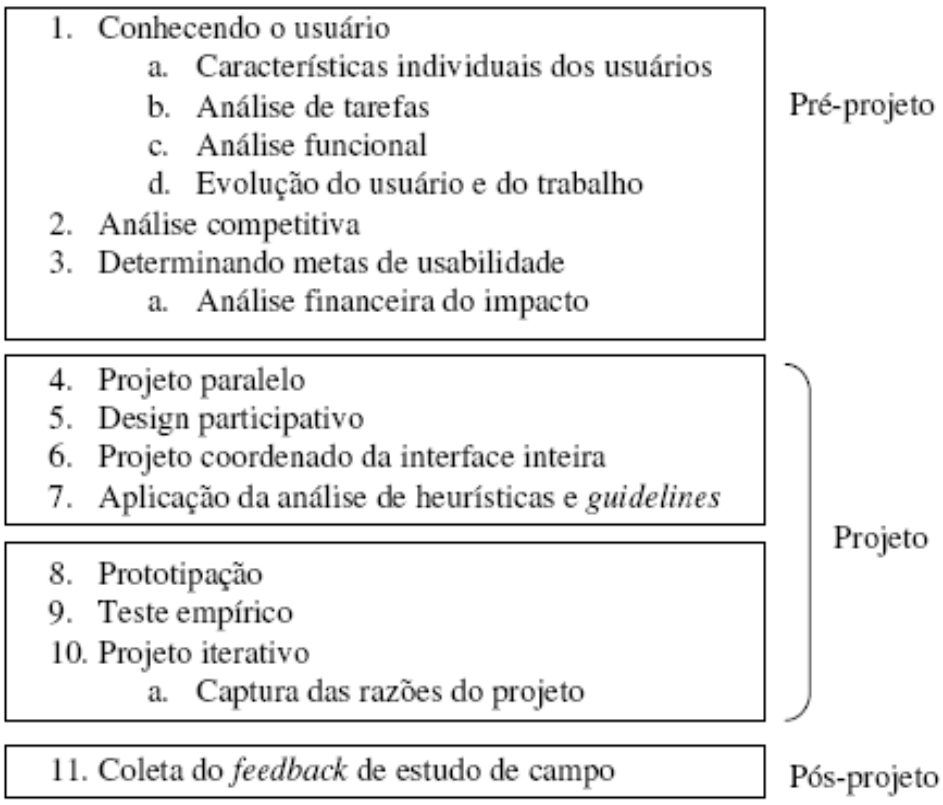


Figura 4.4: Estágios do modelo de processo da Engenharia de Usabilidade. Adaptada de Nielsen [44].

A fase de pré-projeto é caracterizada pela busca de informação e conceituação sobre o usuário e seu contexto de trabalho e sobre sistemas relacionados, padrões de interface, *Guidelines*, ferramentas de desenvolvimento, etc. A fase do pré-projeto é constituída da especificação inicial da interface. A próxima fase é a do desenvolvimento iterativo alimentado por *feedback* de testes até que os objetivos tenham sido alcançados. Finalmente há a fase do pós-projeto com a instalação do sistema no local de trabalho do usuário e acompanhamento com medidas de reação e aceitação do sistema pelo usuário final (Rocha e Baranauskas [51]).

Segundo Gould *et al.* [29] os estágios do projeto para usabilidade ilustram os quatro princípios básicos que fundamentam esse processo: foco no usuário mais cedo, medição empírica, projeto iterativo e integrado de todos os aspectos de usabilidade do sistema (Rocha e Baranauskas [51]).

A etapa de projeto preocupa-se em explorar ao máximo possíveis soluções de projeto antes da escolha de solução final para o sistema. Logo, diversas alternativas são desenvolvidas e analisadas, retirando as melhores idéias e combinando-as, formando novos projetos. Em seguida é realizado o *Design* Participativo, discutido anteriormente, objetivando retirar dúvidas e suposições que os projetistas fizeram nas etapas anteriores. Ainda nessa etapa, o

projeto será avaliado através de aplicações de heurísticas e *Guidelines*, além de verificar a consistência entre versões e outras partes do sistema, como a documentação para o usuário (Nielsen [44]).

Muito importante nesta fase é o *design rationale*, um registro que explicita cada decisão de *design*. Um formalismo que pode ser utilizado para esse registro é o gIBIS (*graphical Issue-Based Information System)*, que captura as questões-chave, idéias e argumentos durante uma conversação e grava em forma de diagramas. O *design rationale*, além de manter a memória do processo de projeto, ajuda a manter a consistência ao longo de diferentes versões do produto (Rocha e Baranauskas [51]).

O estágio de pós-projeto caracteriza-se por conduzir estudos de campo do produto em uso, para obter dados para nova versão e produtos futuros. Esses estudos devem ir além do registro imediato de reclamações buscando avaliar o impacto do produto na qualidade do trabalho do usuário. Os usuários devem ser visitados em seu local de trabalho e devem ser colecionados registros de sessões de uso do sistema para análise (Rocha e Baranauskas [51]).

Entre os principais benefícios da Engenharia de Usabilidade citados na literatura está o tempo economizado em não implementar funções que a análise de usabilidade mostrou não serem utilizadas pelos usuários. Estudo de caso documentado de Nielsen mostrou também uma economia financeira equivalente ao dobro do que foi investido, com redução de treinamento para determinados produtos. Além da economia de tempo e dinheiro, a adoção de produtos adicionais é quase certa, se são fáceis de usar (Rocha e Baranauskas [51]).

Capítulo 5

Aplicação de Padrões de IHC no Processo de Desenvolvimento de Sistemas com Interfaces Gráficas

5.1 Considerações Iniciais

Projetar sistemas computacionais que cumpram as expectativas de usabilidade, é uma tarefa difícil, e pesquisadores da área de Interação Humano-Computador precisam trabalhar em conjunto com os membros da equipe de vários domínios. Em especial, estes precisam cooperar com especialistas no domínio da aplicação para identificar os conceitos, tarefas e terminologia do ambiente, e juntamente com a equipe de desenvolvimento garantir que o projeto do sistema suporte as técnicas de interação requisitada (Borchers [14]).

5.2 Modos de Uso de Padrões em IHC

Borchers [14] *apud* Bayle *et al.* [9] explicitam que existem vários modos de uso de padrões em IHC:

a) captura e descrição: descreve as características fundamentais de uma situação ou acontecimento em uma maneira sensível ao contexto;

b) generalização: generaliza em diferentes situações, conservando a concretude;

c) prescritivo: padrões podem ser usados para prescrever soluções para problemas comumente encontrados em uma determinada concepção de domínio. Assim, padrões podem ser usados para representar *Guidelines* de IHC, ou *Guidelines* organizacionais de *design*;

d) retórico: a concretude de padrões, e o fato de serem originados de situações que ocorreram várias vezes no domínio específico, fazem o uso de padrões apropriados para tornarem-se *lingua franca* para os envolvidos no projeto do artefato;

e) preditivo: padrões podem proporcionar um mecanismo "*what-if*" por refletir sobre possíveis impactos de modificações destes para um lugar ou situação (analisando os impactos diretos nas mudanças de padrões, e tracejar ramificações dessas possíveis mudanças através da interligação de padrões).

Erickson [21] durante o *workshop* UPA'98 refinou estas idéias, justificando que os padrões podem ser usados como um modo de descrever ambientes de trabalho, o que possibilita aos *stakeholders*[4] identificar problemas recorrentes de *design* e incentiva a participação dos usuários no projeto, aumentando a acessibilidade dos resultados do processo de desenvolvimento do *software* (Borchers [15]).

Tidwell [61] enxerga vários benefícios que os padrões de IHC podem trazer ao desenvolvedor de interfaces e à comunidade de *design* de interfaces, tais como configurar uma *lingua franca* para todos os envolvidos no projeto (Borchers [15]):

a) padrões podem capturar conhecimento de *designers* experientes de IHC e correlatos;

b) padrões possibilitam ao desenvolvedor pensar em soluções que não estejam implementadas em *frameworks* ou *toolkits*;

c) excelentes protótipos de interface não possuem aceitação ampla, e padrões podem ajudar a extrair as características principais destas soluções, e divulgar na comunidade de *design* de interfaces;

d) padrões podem ser utilizados como um ponto de partida para implementar novas interfaces;

e) alguns padrões podem ser implementados em *software*.

Dearden e Finlay [18] acrescentam ainda algumas características importantes em padrões, pois todos são validados pelo uso e refletem valores de *design* e resolvem problemas através de diferentes escalas.

[4] Participantes do processo de desenvolvimento do sistema.

Segundo Borchers, não há necessidade de seguir um modelo de processo específico de IHC para utilizar a abordagem de linguagens de padrões. Dix *et al.* [20] comentam que provavelmente 90% do valor da técnica de *design* de interface constitui-se em obrigar o desenvolvedor a lembrar que outra pessoa, além dele, irá utilizar o sistema em construção. Não existe necessidade; porém como o projeto necessita de usabilidade, os modelos de processo são recomendados e padrões de desenvolvimento são dispostos em todas as fases deste.

5.3 Padrões de IHC Aplicados na Engenharia de Usabilidade

Um modelo que utiliza a abordagem do uso de linguagens de padrão é a Engenharia de Usabilidade de Nielsen, na qual padrões "encaixam-se" nas 11 atividades que o modelo sugere. A seguir são apresentadas as considerações de Borchers [13] sobre a forma de como os padrões poderiam ser utilizados em cada uma das fases da Engenharia de Usabilidade:

5.3.1 Conhecendo o Usuário

Na primeira fase determina-se se o domínio da aplicação é adequado para expressar seus conceitos e métodos em forma de padrões. Isto será verdade se o trabalho neste envolver atividades criativas, *design*, ou resolução de problemas, devido a regras e *Guidelines* que levam as atividades a serem formuladas como padrões com as características específicas deste domínio.

A análise das tarefas correntes e desejadas pode utilizar-se de padrões para documentar essas tarefas, após uma instrução clara dos conceitos e formatos do padrão por um especialista do domínio ou na área de Interação Humano-Computador.

Outra vantagem é que este formato possibilitará acessibilidade a este material para a equipe de desenvolvimento, facilitando também o reconhecimento dos usuários quando estes padrões forem utilizados nos objetos da interface e o produto final. Nesta fase os padrões podem ser usados como forma de captura e descrição.

5.3.2 Análise Competitiva

Durante esta fase, produtos existentes no mercado são examinados para encontrar soluções diferentes para problemas na área de atuação do *software*, e podem ser usados para comparar usabilidade antes mesmo de o sistema ser projetado. Enquanto a arquitetura interna não é acessível, as linguagens de padrões de IHC podem generalizar soluções de sucesso observadas em produtos concorrentes e sugerir um *design* para o novo sistema. Nesta etapa, os padrões são sugeridos como forma de generalização.

5.3.3 Determinando Metas de Usabilidade

Os vários aspectos de usabilidade, tal como facilidade na aprendizagem, eficiência de uso e baixa taxa de erros, precisam ser analisados e priorizados, tornando-se forças de padrões de IHC de alto nível, balanceando os aspectos para que não haja conflitos entre eles. Um *design* de sistema usado principalmente por usuários *experts* pode tender mais à eficiência de uso do que à facilidade de aprendizagem. Nesta fase os padrões são usados de modo preditivo.

5.3.4 Projeto Paralelo

Para a construção de interfaces, muitos protótipos iniciais podem ser projetados por equipes independentes. Neste caso, e para estágios posteriores, padrões gerais de IHC podem assumir o papel de *Guidelines* de *design*, e ajudar a criar um "terreno comum" preservando as metas de usabilidade derivadas da terceira fase entre as equipes trabalhando em modo paralelo. Nesta fase os padrões são usados de modo prescritivo.

5.3.5 *Design* Participativo

Esta parte do processo é baseada no princípio do *design* participativo, proposto para incluir o usuário que é considerado especialista no domínio da aplicação na equipe de desenvolvimento do sistema, e no conceito de padrões interdisciplinares, integrando diferentes domínios através de uma *lingua franca* (modo de uso retórico) para que todos possam contribuir na construção de métodos, interface e modos de interação entre máquina e homem.

5.3.6 Projeto Coordenado de Toda a Interface

Um projeto coordenado visa criar uma interface consistente, documentação, ajuda *on-line* e tutoriais, todos sobre o produto atual e versões passadas que podem estar envolvidas em

outros produtos da mesma equipe. Padrões de IHC, especialmente os que tratam de *design* podem servir de vocabulário dentro da equipe de desenvolvimento (modo de uso retórico), para assegurar consistência do sistema. Dicionários de termos da interface são necessários para suportar este processo.

5.3.7 Aplicação de *Guidelines* e Análise de Heurísticas

Style guides, *Guidelines* e *Standards* são modos de capturar experiência de projeto, muito próximo da função dos padrões. Porém estes podem incrementar estas técnicas devido ao conteúdo e formato estruturado, combinando exemplos concretos e soluções gerais, com explanação perspicaz não somente da solução, mas também do contexto do problema, e da integração com outros padrões dentro de uma linguagem.

Essa cobertura de múltiplas camadas de abstração e experiência é similar à distinção que ocorre entre *general Guidelines*, *category-specific* e *product-specific Guidelines* (Nielsen [44]).

Category-specific Guidelines são produtos de memória corporativa, geralmente proveniente de projetos anteriores que são generalizados e disponibilizados para projetos futuros.

Product-specific Guidelines são desenvolvidos comumente como parte de projetos individuais assim que os membros do projeto melhoram o entendimento sobre aspectos de usabilidade do sistema. Este entendimento só pode vir através de análise de heurísticas e testes adicionais dos protótipos do novo sistema. Nesta fase, os padrões servem como modo de generalização dos *Guidelines*, *Style guides* e *Standards* e prescreve soluções para o problema (modo prescritivo).

5.3.8 Prototipação

O tradicional modelo em cascata (*waterfall model*) para desenvolvimento de *software* sugere criar os programas em si somente no fim do ciclo de desenvolvimento. Protótipos ajudam na rapidez da demonstração da interface ao cliente, embora limitado em termos de funcionalidade, desempenho e estabilidade. Este aspecto da Engenharia de Usabilidade é orientado à implementação, e padrões tem um papel importante nesta fase. Se o grupo de desenvolvimento expressa as idéias de projeto, os *Standards* arquiteturais e os componentes em forma de padrão, então o grupo de projeto da interface pode relacionar todos os conceitos envolvidos mais facilmente, e conseqüentemente, melhorar o entendimento do sistema por

parte de toda a equipe. Por exemplo, o grupo de projeto de IHC pode concordar em algumas funcionalidades implementadas no protótipo que sejam fáceis de codificar sem comprometer o funcionamento do protótipo. Nesta etapa, os padrões funcionam de modo preditivo, para estruturar uma possível mudança no sistema de acordo com os padrões utilizados.

5.3.9 Teste Empírico

Protótipos, desde os primeiros rascunhos até o sistema final, são testados com usuários especializados em descobrir problemas de *design*. Padrões de domínio da aplicação podem ser um recurso para construir cenários realistas para teste. Problemas descobertos são relacionados e se possível convertidos em padrões de IHC para possível resolução. Devido à opinião dos *experts*, os padrões aqui podem exercer um papel de generalização das recomendações destes, e também de modo prescritivo, pois através da descoberta do problema é proposto um padrão que os resolva.

5.3.10 Projeto Iterativo

Baseados principalmente no *feedback* do usuário, protótipos são redesenhados e melhorados em um processo iterativo. Padrões de IHC ou experiência em projetos de *software* são importantes para possibilitar aos *designers* sobre opções nesta fase, porque são construtivos, sugerindo como um problema pode ser resolvido, em contraste aos *general design Guidelines*, que tem natureza descritiva, apenas elencando características de um bom sistema interativo de modo prescritivo.

Naturalmente, padrões evoluem durante o curso do projeto, refletindo o progresso no entendimento do domínio do problema e melhorando o *design* do sistema. Soluções de sucesso tornam-se concretas e podem fornecer exemplo de um padrão utilizado, ou garantir a escrita de um novo padrão.

Após o projeto concluído, é feita uma documentação de todas as decisões que deram certo e também os padrões que não surtiram efeito esperado no projeto, denominados *anti-patterns*.

5.3.11 Coleta do *Feedback* de Estudo de Campo

Entregue o sistema, métodos como estudo de campo, estudos de *marketing* ou análise de chamadas de ajuda podem ser usados para atestar o resultado da Engenharia de Usabilidade. Em conversa com os usuários finais, os padrões do domínio da aplicação demonstram novamente um importante papel, servindo de vocabulário comum entre *experts* de interface e

usuários finais. *Feedback* pode ser usado para fortalecer os argumentos da escolha destes padrões usados e ajudar a concretizar o padrão como uma solução aprovada para o sistema em questão. Na última fase, os padrões são usados retoricamente, pois fornecem um vocabulário comum e também prescreve soluções para sistemas futuros.

5.4 Avaliação de Linguagens de Padrões de IHC

Existe hoje um grande número de padrões, coleções e linguagens de padrões de IHC. Wania [68] *apud* Henninger e Corrêa [33] explicitam que existem mais de 400 padrões publicados dentro de várias coleções e linguagem de padrões. O Anexo D contém alguns exemplos de linguagens de padrões, com o número de padrões que a compõem.

Os participantes do *workshop* CHI 2002 discutiram outros termos adequados para avaliar linguagens de padrões incluindo extensão, profundidade, aplicabilidade, clareza e conveniência (Todd *et al*. [64]). Todd *et al.* propõem seis questões que podem ser aplicadas para determinar a validade de uma linguagem de padrões (Wania [68]):

a) A ligação de referência e contexto entre os padrões, formam um mapa?

b) O mapa de contexto iguala-se ao mapa de referência?

c) O mapa pode ser ordenado em uma hierarquia de níveis?

d) Os níveis podem ser usados para descrever uma interface em diferentes graus de granularidade ou escala?

e) Quão ricas são as ligações entre cada nível da hierarquia?

f) Os padrões podem ser organizados sob classificações diferentes provendo pontos de vista alternativos?

Todd *et al.* [64] usaram estas questões para avaliar três coleções de padrões: van Welie [71] (para aplicações GUI), van Welie [71] (para aplicações *Web*) e Borchers [15] (para Interação entre o homem e qualquer dispositivo computadorizado, ver Anexo C). Nenhuma destas coleções passou em todos os testes. Após isto, os pesquisadores pediram a diminuição das exigências dos testes, o que permitiria a coleção de Borchers ser considerada uma linguagem de padrões. Mesmo estas seis questões parecerem válidas, não existem estudos empíricos que suportem a validação das perguntas para avaliação das linguagens (Wania [68]).

Ainda segundo Wania [68], vários problemas foram identificados no estado atual da pesquisa de padrões e linguagens de padrões de IHC. Alguns problemas reconhecidos pela comunidade de Interação Humano-Computador são:

a) falta de um formato padrão (Seffah e Javahery [52]);

b) falta de um princípio que sirva para organização dos padrões (Fincher [25], Fincher e Windsor [24], Welie e Veer [69]);

c) falta de ferramentas para acesso das coleções e linguagens de padrões (Seffah e Javahery [52]);

d) diferenças entre padrão e linguagem de padrões.

Enquanto Seffah e Javahery [52] e Welie e Veer [69] apontam como falhas de usabilidade e acessibilidade a falta de ferramentas para o acesso das linguagens de padrões e não existir um formato amplamente aceito para os padrões, atualmente, não está claro que comprometem realmente a usabilidade. Existe uma carência nesta área para fazer afirmações baseadas em estudos empíricos.

Fincher *et al.* [27] explicam que a maioria do esforço, envolvendo padrões na área de IHC, está concentrado em encontrar o formato ideal para os padrões. Esta é uma discussão válida, pois todos os pesquisadores poderiam beneficiar-se de um formato padrão. Porém, ainda não existem estudos que sugerem que um formato é melhor que o outro.

Apesar de haver um entendimento em comum que não existe uma linguagem universal de padrões ou um princípio que sirva para classificar as linguagens, não se sabe se isso será possível ou será útil. Novamente, precisa-se de mais estudos empíricos nesta subárea (Wania [68]).

5.5 Considerações Finais

Existem vários modos de aplicar padrões de Interação Humano-Computador no desenvolvimento de sistemas com interfaces gráficas. Pode-se citar o modo de captura e descrição, que descreve características principais em alguma situação; o modo de generalização que generaliza acontecimentos para formular um padrão; o modo prescritivo, que pode representar regras, diretrizes em padrões; o modo retórico que faz uso de padrões como linguagem em comum na equipe de desenvolvimento; e o modo preditivo, em que

padrões servem para analisar futuras mudanças no sistema e os impactos resultantes desta modificação.

Não há necessidade de seguir um modelo específico para utilizar a abordagem de linguagens de padrões, porém a Engenharia de Usabilidade mostra-se adequada para aplicação pois encaixa padrões nas onze fases que a compõe.

Além dos problemas discutidos na literatura, incluindo: acesso aos padrões, suporte de ferramentas, organização de padrões, e relacionamento entre eles, existe também a questão de padrões e linguagens serem de uso geral. Entende-se por uso geral os padrões e linguagens não tenderem a um domínio particular.

Todas estas questões apontam para um grande problema existente dentro da área de Interação Humano-Computador, que é a falta de estudos empíricos que atestem ou rejeitem as proposições de padrões e linguagens de padrões.

O próximo capítulo contém os resultados, bem como observações realizadas no decorrer do desenvolvimento deste trabalho e sugestões para pesquisas de trabalhos futuros.

Capítulo 6

Conclusão

Sabe-se que um padrão é uma solução comprovada para um problema recorrente de projeto. Os padrões dedicam uma atenção especial ao contexto no qual é aplicável, às forças conflitantes que precisam de equilíbrio, e às conseqüências positivas ou negativas de sua aplicação.

Projetar sistemas computacionais é uma tarefa complexa que envolve pessoas de diferentes disciplinas. Os maiores problemas em projetar uma interface incluem complexidade deste projeto e comunicação entre a equipe de desenvolvimento. Como muitos da comunidade de IHC sugeriram, padrões objetivam resolver alguns destes problemas de *design*. Embora haja uma grande esperança devido à promessa de linguagens de padrões em IHC (Pemberton [47]), há pouco trabalho empírico para apoiar estas deduções (Dearden e Finlay [18]).

Comunicação em equipes interdisciplinares é o maior problema para os pesquisadores de IHC. Borchers [13] sugere que todos os membros participantes do projeto do sistema, principalmente especialistas de IHC, engenheiros de *software*, e *experts* do domínio de aplicação, formulem seus métodos, experiências e valores na forma de linguagens de padrões, como originalmente aplicado na Arquitetura Civil. Para uma representação uniforme e suporte computacional, Borchers propõe uma representação formal de padrões e relações entre eles.

Aplicar a técnica de padrões em novos domínios de aplicações pode futuramente fortalecer o argumento de (Borchers [13]), o qual afirma que estruturar estes domínios em padrões é uma abordagem válida, e o uso intensivo de padrões de Interação Humano-Computador poderá demonstrar a utilidade destes no *design* de interfaces.

A referência de padrões de alto nível, descrevendo o contexto na qual pode ser aplicado, e de baixo nível, que podem ser usados para refinar a solução, formam uma hierarquia que estrutura uma coleção compreensiva de padrões em uma linguagem de padrões (Borchers [13]).

A linguagem de padrões faz com que os conhecimentos e suposições de diversos participantes do projeto tornem-se mais explícitos, e mais fáceis para que outras disciplinas o compreendam e possam referenciá-lo. Este vocabulário comum pode melhorar consideravelmente a comunicação entre a equipe, e também servir como uma memória corporativa de experiência de *design* (Borchers [13]).

Enquanto isso, um grande número de linguagens de padrões para interação tem sido sugerido. A linguagem de Tidwell [60], por exemplo, abrange um campo substancial na questão de projeto de interface (Borchers [13]).

Padrões compreendem uma ferramenta de sucesso para modelar experiência de *design* na Arquitetura Civil, na Engenharia de *Software* (com as limitações discutidas anteriormente) e, coleções existentes, tais como a de Tidwell, também em IHC (Borchers [13]).

Com referência ao trabalho de Alexander [3], a comunidade declara que os padrões podem constituir uma linguagem de projeto para comunicação entre engenheiros de *software* e usuários, assim como a linguagem de padrões de Alexander o fez entre construtores e habitantes (Borchers [13]).

Granlund e Lafreniere [31] usam padrões para descrever domínios de negócios, processos e tarefas para ajudar no *design* conceitual e definições prévias de sistema. Também declaram o valor da interdisciplinaridade dos padrões, tais como o meio de comunicação e a sua habilidade em capturar conhecimento de projetos anteriores (Borchers [13]).

De fato, não há razão pela qual a experiência, os métodos e os valores de qualquer domínio de aplicação não possam ser expressos na forma de padrão, desde que a atividade no domínio de aplicação inclua algum tipo de *design*, trabalhos criativos ou relacionados à resolução de problemas (Borchers [13]).

Isto levou o autor à idéia de que não somente profissionais de IHC e engenheiros de *software*, mas também *experts* em domínio de aplicação, pudessem expressar suas respectivas experiências e *Guidelines* na forma de linguagem de padrões (Borchers [13]).

Bayle *et al.* [9] explicitam que é uma tarefa difícil fazer bons padrões e não tem uma certeza sobre como deve ser construída uma linguagem de padrões. Ainda sobre a criação de padrões, os pesquisadores citam que realizar esta criação é um exercício de aplicar valores, ou seja, o que as pessoas acham ser importante ser mostrado em um padrão. Não existe um acordo, segundo eles, sobre o que faz um padrão ser um padrão e uma linguagem de padrões ser uma linguagem de padrões. Essa é uma área que ainda é carente de evidências empíricas (Wania [68]).

A comunidade de IHC precisa focar na identificação dos benefícios em usar padrões e linguagens de padrões. Até hoje, não existem evidências derivadas de pesquisas que sugiram que padrões e linguagens de padrões ajudam *designers* dentro do processo do projeto de interfaces ou que produzam *design*s de alta qualidade ou que possuam um *design* melhor que outro, feito através de outros meios. O que se sabe é que ajudam a produzir *designs* de boa qualidade (Wania [68]).

Este trabalho foi realizado em um período curto, com uma bibliografia substancial, trazendo como conclusões derivadas desta pesquisa, a linguagem de padrões de Tidwell, que é considerada pela comunidade o maior esforço para identificar e catalogar padrões, mesmo possuindo padrões que foram encontrados, mas não foram escritos ainda. Esta linguagem de padrões de Tidwell serve principalmente como *lingua franca* entre a equipe de desenvolvimento, capturando situações que necessitavam de usabilidade no *design* de uma interface, parte fundamental entre a interação entre homem e máquina, padrões estes considerados de baixo nível, aproximando-se do formato proposto por Gamma *et al*.

Os padrões de alto nível tem como maior representante o esforço de Borchers, pesquisador de renome na área de Interação Humano-Computador, pois envolvem aspectos da interação entre humano e máquina, sendo sua linguagem orientada ao domínio musical, aproximando-se do formato proposto por Alexander.

Estas linguagens de padrões estão organizadas de uma maneira lógica, que facilita aos utilizadores da abordagem a compreensão do padrão e como fazer as relações entre os padrões existentes nesta linguagem. Quanto ao formato, ainda não existe um formato ideal, apenas propostas que abrangem linguagens de diversos pesquisadores, orientadas à diversos domínios.

Como formatos de linguagens de padrões, podemos citar a PLML[5], o POSA[6] e o de GoF[7], e nenhum destes consegue abranger a diversidade de linguagens de padrões existentes hoje.

Estamos em um ponto em que estudos empíricos precisam ser a principal preocupação da comunidade de Interação Humano-Computador, ficando a frente de desafios que surgem com a evolução da informática.

---

[5] *Pattern Language Markup Language.*
[6] *Pattern Oriented Software Architecture.*
[7] *Gang of Four.*

Este trabalho expôs a falta de estudos empíricos na área de linguagem de padrões, padrões, e que vantagens obtêm-se ao utilizar estas abordagens no desenvolvimento de sistemas com interfaces gráficas. Sabe-se que podem funcionar muito bem como *lingua franca*, capturar a experiência de projetos anteriores, prover reuso, porém, não provou-se até o momento que atributos de usabilidade possam ser atingidos sem usar estes métodos específicos.

Através de observações feitas durante o desenvolvimento do trabalho apontam-se sugestões para pesquisas e possíveis trabalhos, tais como, pesquisa de um formato que suporte todas as linguagens existentes hoje; estudos empíricos da aplicação de uma linguagem de padrões conforme um cenário existente e realizar observações e testes sobre estes dados obtidos; a construção de um *framework* que auxilie a aplicação destes padrões e documente o projeto em todas as fases; verificar através de observações se uma linguagem orientada a um domínio específico pode ser adaptada a outro domínio e, finalmente, realizar uma pesquisa com usuários finais comparando projetos em que foram adotados padrões e linguagens de padrões e projetos que não tiveram nenhum tipo de orientação específica de Interação Humano-Computador para analisar sob o ponto de vista de usabilidade destes sistemas construídos.

Anexo A

Coleção de Padrões de IHC de Tidwell

A.1 *Wizard*

A.1.1 Descrição

Conduz o usuário pela interface passo-a-passo, executando tarefas em uma ordem pré-definida.

A.1.2 Quando Usar

Deve-se usar quando deseja-se projetar uma interface para uma tarefa complicada ou longa, e para o usuário é uma coisa que eles não realizarão com freqüência ou queiram um controle maior sobre a tarefa.

Esta técnica ajuda a resolver um problema de usabilidade apontado por Nielsen [44], que diz “Não me faça pensar, apenas diga-me o que eu faço a seguir”. Você não se preocupa em saber como é a estrutura de uma reserva de passagem, apenas quer comprá-la.

Mas em outros contextos, não é recomendada. Usuários avançados classificam o uso de *Wizard* frustrante e limitante, exemplo, na área de programação.

Tidwell [63] complementa que este padrão não mostra ao usuário as ações que ele realmente ocasiona, o que pode irritar os clientes deste sistema, necessitando um conhecimento sobre o nível de seus usuários.

A.1.3 Por quê?

Tidwell [63] cita: Dividir para conquistar. Dividir a tarefa em uma seqüência de passos, cada um dos usuários podem lidar com um discreto “espaço mental”, o que simplifica a tarefa. Através de um mapa pré-planejado para realizar a tarefa, evitando que o usuário realize o

esforço de entender a estrutura envolvida, tudo que eles precisam fazer é seguir os passos que a tarefa será concluída com sucesso.

A.1.4 Como

Dividir a tarefa em passos, numa seqüência lógica e dependendo da área de aplicação, incluir telas para seleção de produtos, informações de pagamento, endereço de contas e endereço de envio. Colocar escolhas relacionadas juntas simplifica a tarefa de preencher esses formulários.

Em um *software* de instalação, pode-se utilizar o *Wizard* para oferecer a instalação de pacotes adicionais e opções sobre eles. Interfaces dinâmicas são boas em representar tarefas ramificadas como esta, porque o usuário nunca precisaria ver o que é irrelevante para as escolhas que ele fez.

A parte difícil em projetar este padrão é saber balancear o tamanho da parte e o número de partes em que a tarefa será dividida. Tidwell comenta que é ridículo um *Wizard* com dois passos e entediante um com 15.

Quanto à estrutura física, este padrão pode apresentar cada passo em uma página ou tela diferente, com botões de navegação, *back* e *next*. Esse padrão é somente uma boa estratégia se permitir o avanço ou retrocesso dos passos, pois o usuário pode cometer erros, não intencionais, e comprometer o passo atual. Muitas interfaces dispõem de mapas ou uma visão geral dos passos, usando os benefícios de outro padrão: *Two-Panel Selector*.

A.1.5 Exemplo

Figura A.1: Reserva de passagens da companhia Grand Canyon Railway (Tidwell [63]).

A.2 *Two-Panel Selector*

A.2.1 Descrição

Coloca dois painéis lado a lado na interface. No primeiro lado, mostra um conjunto de itens que o usuário pode selecionar e no outro, mostra o conteúdo do item selecionado.

A.2.2 Quando Usar

Quanto está apresentando uma lista de objetos, categorias ou até mesmo ações. Mensagens em uma caixa de correio, seções de uma página web, músicas ou imagens em uma biblioteca, dados gravados de bancos de dados, e arquivos são bons candidatos. Cada item tem um conteúdo interessante associado, assim como textos de uma mensagem de e-mail ou detalhes sobre o tamanho ou data de criação de um arquivo. Quando o desenvolvedor quer que o usuário veja toda a estrutura da lista, mas também possibilitar ao usuário escolher a ordem que visita.

Fisicamente, o monitor dos computadores é grande o suficiente para mostrar dois painéis de uma vez. Porém celulares não beneficiam desta técnica, devido à limitação da tela.

A.2.3 Por quê?

O padrão é uma convenção, mas é extremamente poderoso e simples. Pessoas rapidamente aprendem que devem selecionar um item do primeiro painel para ver seu conteúdo no outro painel. Isto é importante, pois quando aprende-se a utilizar este dois painéis, o usuário pode beneficiar-se de muitos aplicativos, pois funcionam de maneira idêntica.

Quando os dois painéis são visíveis lado a lado, usuários podem mudar a atenção sobre os painéis rapidamente, observando toda a estrutura da lista ou vendo os detalhes do objeto. Essa integração tem muitas vantagens sobre outras estruturas, como duas janelas separadas ou detalhamento de uma janela.

O padrão reduz esforço físico. Os olhos do usuário não precisam viajar uma longa distância entre os painéis, e pode mudar a seleção com um simples clique do mouse ou pressionar de tecla, menos que navegar entre janelas e telas, que pode exigir cliques extras no mouse.

O padrão reduz a carga cognitiva visual. Quando uma janela aparece no topo, ou quando um conteúdo de uma página é completamente mudado (como acontece no detalhamento de

uma janela), o usuário sumariamente deve prestar mais atenção ao que está olhando agora e quando a janela fica mais estável, assim como acontece no padrão *Two-Panel Selector*, o usuário pode focar em uma área menor que mudou.

O padrão reduz a carga de memória do usuário. Pense em um *e-mail*. Quando o usuário olha simplesmente no conteúdo da mensagem, não tem nada que lembre onde esta mensagem estava na caixa de entrada. Se ele quiser saber, ele deve lembrar ou navegar novamente a lista de *e-mail*. Mas se a lista já está na tela, ele precisa simplesmente olhar, e não relembrar.

A.2.4 Como

Coloque a lista selecionável no topo do painel da esquerda e o painel de detalhes abaixo ou à sua direita. Isso faz com que aproveite o fluxo visual da maioria do usuário que lê da esquerda para a direita.

Quando um usuário selecionar um item, mostre imediatamente seu conteúdo ou detalhes no segundo painel (da direita). A seleção deve ser feita através de um clique simples. Enquanto você está no painel de seleção, permita que o usuário utilize as setas do teclado para mudar a seleção do item da esquerda. Isto faz com que reduza o esforço físico e o tempo requerido para a navegação e contribui para usabilidade “somente teclado”.

Faça com que o item selecionado seja visualmente realçado. A maioria dos toolkits de interfaces tem um modo particular de mostrar esta seleção, tal como inverter as cores de letra e fundo.

A aparência da lista selecionável depende da estrutura do conteúdo, ou tarefa a ser executada. A maioria dos visualizadores de sistemas de arquivos mostra a hierarquia do diretório, desde como estes são estruturados.

Você pode usar padrões de apresentação da informação tal como *Sortable Table*, que mostra os detalhes de arquivos em forma de tabela, e *Tree-Table*, que mostra os detalhes em forma de tabela com hierarquia, no padrão *Two-Panel Selector*, para componentes como listas e árvores. O padrão *Card Stack* relaciona-se diretamente com o padrão *Two-Panel Selector*.

A.2.5 Exemplo

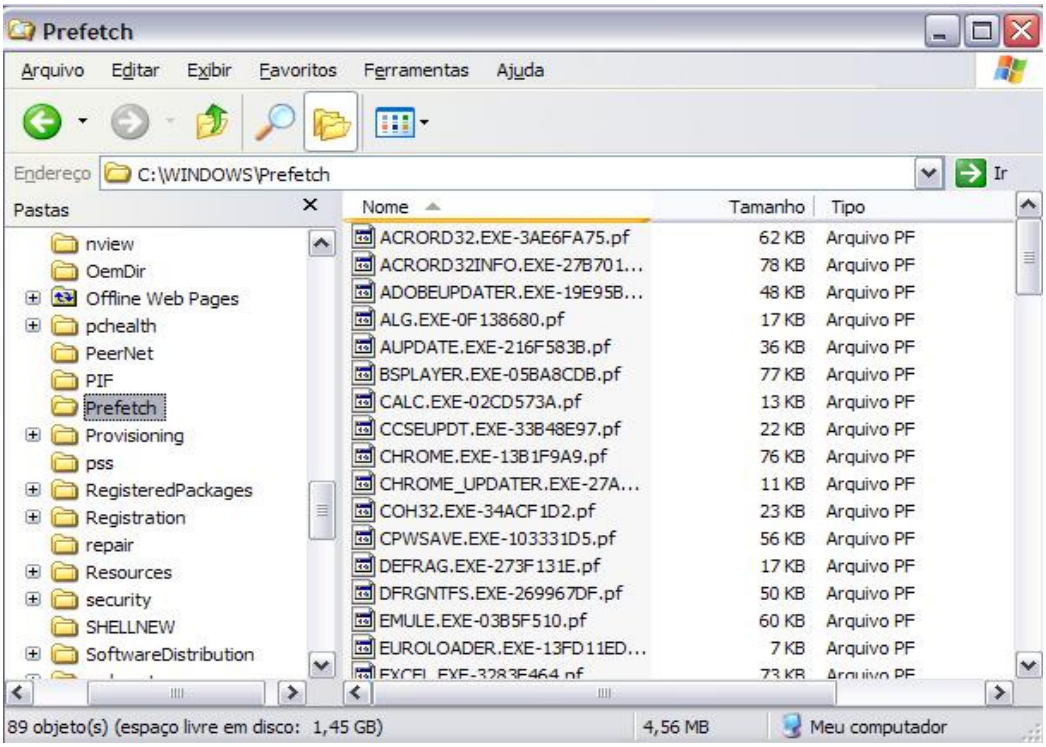


Figura A.2: Detalhes de arquivos através do Windows Explorer .

A.3 *Extras on Demand*

A.3.1 Descrição

Mostra o conteúdo mais importante, e esconde o resto. Permite o usuário acessar este conteúdo oculto via simples comando.

A.3.2 Quando Usar

Deve-se usar quando muitas coisas são mostradas na tela, e algumas delas não são tão importantes.

A.3.3 Por quê?

Uma interface simples é na maioria das vezes melhor que uma complexa, especialmente para usuários novatos, ou usuários que não precisam de todas as funcionalidades que o sistema pode prover. Deixar o usuário escolher quando ver todas as funcionalidades é melhor do que o programador escolhê-las para seus usuários.

Se o seu projeto faz 80% dos casos de uso fáceis, seu projeto está indo tão bem quanto o esperado. Quando feito corretamente, este padrão pode economizar muito espaço na interface.

A.3.4 Como

Estender a interface para baixo com os itens ocultos, fazendo que fiquem visíveis, é a estratégia mais utilizada. O resto das funcionalidades fica abaixo da opção escondida anteriormente.

Para indicar que existem mais opções, muitas interfaces colocam caracteres especiais como ">>" ou "..." dentro de um botão ou *link*. Esta seção que foi aberta deve ter um botão ou link que permita o usuário fechar estas opções acessadas.

Em algumas interfaces a janela literalmente expande até acomodar os detalhes da seção, e então encolhe quando o usuário fecha. O padrão *Closable Panels* é uma das maneiras de fazer isto. Muitas interfaces possuem um mecanismo para escolha de cor, que pode levar a uma caixa separada das opções normais.

A.3.5 Exemplo

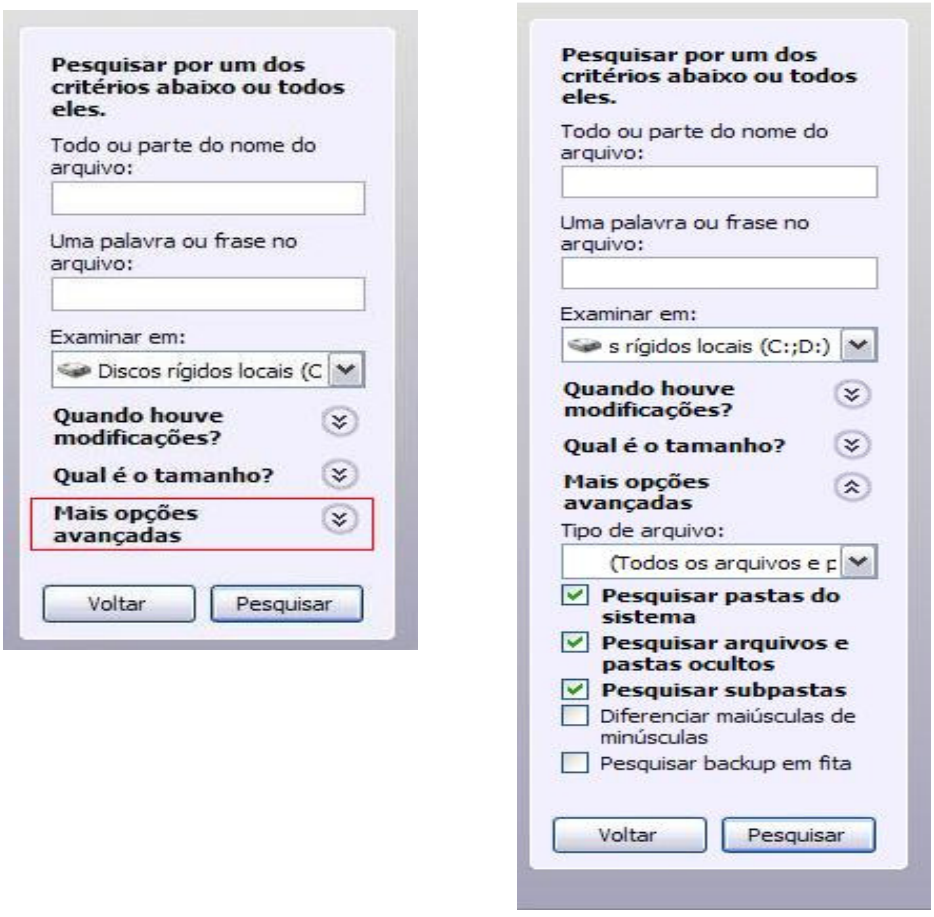


Figura A.3: Menu de pesquisa do Windows XP.

A.4 *Multi-level Help*

A.4.1 Descrição

Usa uma mistura de estratégias consideradas leves e pesadas de ajuda para usuários que possuem diferentes necessidades.

A.4.2 Quando Usar

Quando sua aplicação é complexa. Alguns usuários precisam de um bom sistema de ajuda, mas o desenvolvedor sabe que a maioria do tempo os usuários não irão usá-lo. O desenvolvedor deseja oferecer ajuda a estes usuários, então usa-se este padrão.

A.4.3 Por quê?

Usuários de *software*s precisam de diferentes níveis de ajuda para as tarefas que eles precisam cumprir. Àqueles que usam o programa pela primeira vez precisa de uma ajuda diferente do que a pessoa que usa o programa freqüentemente. Mesmo entre os usuários iniciantes, existem enormes diferenças na aprendizagem. Algumas preferem ler tutoriais, outras não, algumas acham dicas de grande ajuda, muitas acham irritante.

Textos de ajuda providos em vários níveis conseguem atingir quem precisa deles. Muitas boas técnicas de ajuda põem os textos de ajuda em um fácil alcance, mas não diretamente na visão do usuário.

A.4.4 Como

Criar ajuda em diferentes níveis, incluindo técnicas da lista abaixo:

a) legendas e instruções diretamente na página. Se os textos forem feitos concisos, usuários freqüentes não irão se importar com a ajuda, mas não se deve utilizar parágrafos inteiros de texto, pois poucos usuários irão lê-lo;

b) dicas. Use brevemente, ou seja, uma ou duas linhas no máximo, para descrever recursos da interface que não são óbvios de visibilidade. As desvantagens das dicas citadas são que, quando apresentadas, elas ocultam o que está no fundo da tela, e usuários não gostam de dicas aparecendo todo o tempo. Segundo estudos realizados por Tidwell, o tempo gasto para fechar estas janelas fica em torno de dois segundos.

c) descrições brevemente longas são mostradas dinamicamente assim que o usuário seleciona ou passa o mouse em certos elementos da interface. Cria uma área na própria página para isso.

d) textos de ajuda mais longos devem estar dentro de *Closable Panels*.

e) ajuda propriamente dita é mostrada em uma janela separada, geralmente constituindo-se ajuda on-line, ou específica do sistema operacional. Acessado através de itens de *menu*, ou botões em caixas de diálogo.

f) suporte técnico deve ser feito usualmente pelo *e-mail*, *Web sites*, ou telefone.

g) suporte informal, feito geralmente através de uma comunidade sobre o programa, quase sempre através de *Web sites*. Como alguns exemplos, podem-se citar comunidades do Photoshop [2], Ubuntu [65], Mac OS [5], ou MATLAB [41].

A.4.5 Exemplo

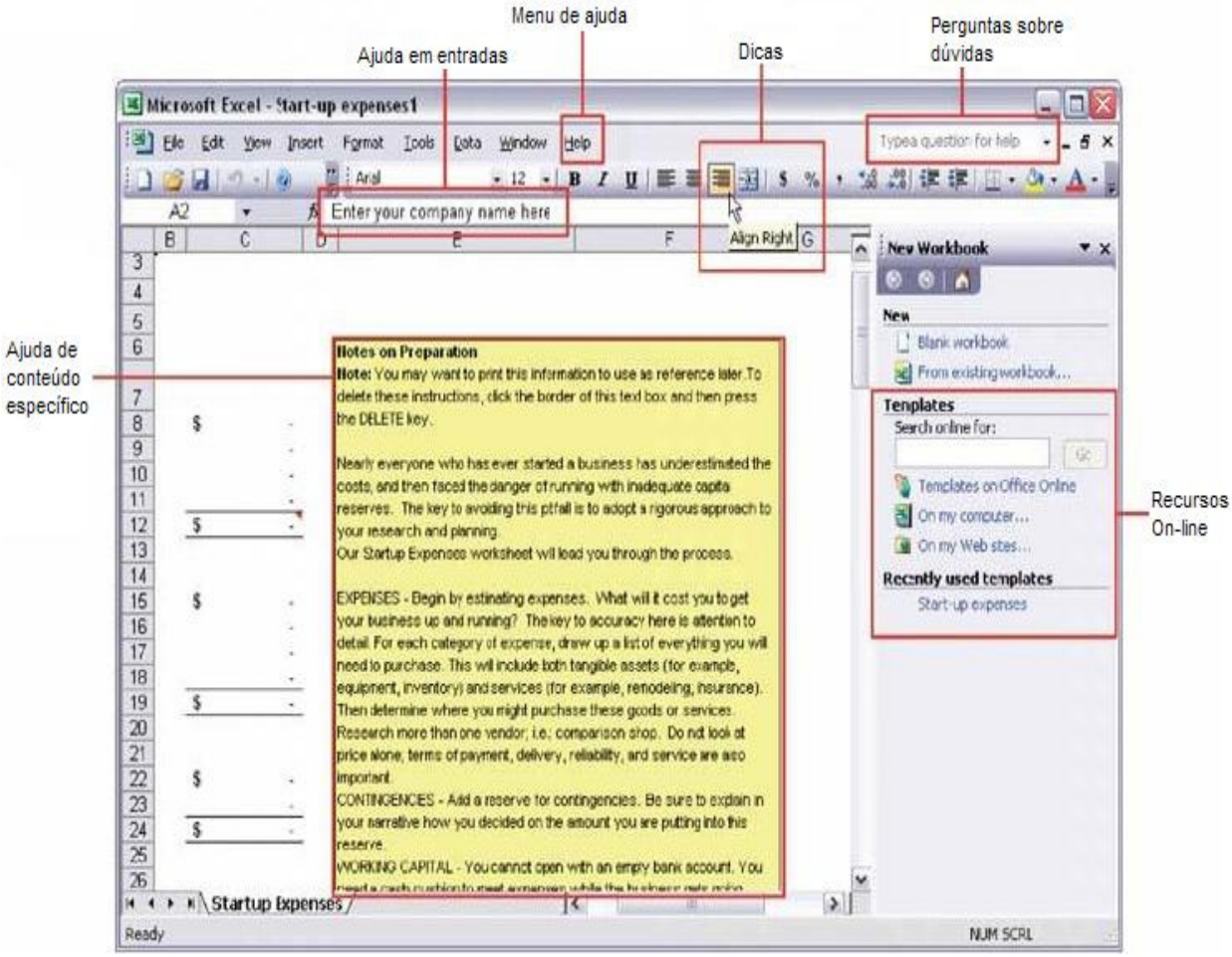


Figura A.4: Recursos de ajuda do Microsoft Excel 2003. Adaptada de Tidwell [63].

A.5 *Closable Panels*

A.5.1 Descrição

Seções de conteúdo em painéis separados, permitindo ao usuário abrir e fechar cada um deles separadamente.

A.5.2 Quando Usar

Existe muito conteúdo na página ou tela, mas deseja-se todo através de um clique. O conteúdo é divisível em seções, utilizando seções com título e o padrão *Card Stack*. O usuário pode querer ver 2 seções ou mais de uma vez.

Especificamente, é uma implementação útil do padrão *Extras on Demand*. Se o desenvolvedor organizar o conteúdo através do *Extras on Demand*, também irá utilizar alguma forma do padrão *Closable Panels* para suportar estes extras.

A.5.3 Por quê?

Este padrão pode ser usado associado ao *Card Stack*, dando uma maior flexibilidade. A seguir são dadas algumas características do padrão *Closable Panels*:

a) O padrão *Card Stack* contém seções de diferentes tamanhos. Se for utilizado, as seções menores devem ter tamanhos maiores de áreas em branco.

b) O usuário pode abrir várias seções de uma vez. Especialmente em ferramentas de autoria, e aplicações que precisam de muitos painéis abertos e vistos ao mesmo tempo.

c) Se o desenvolvedor cria apenas um *Closable Panel* e adiciona em uma pagina maior ou em uma caixa de diálogo, o painel parecerá menos importante que o conteúdo que está adicionado. Porém se o desenvolvedor coloca as seções em duas páginas, os painéis parecerão ter a mesma importância.

Para o uso, é recomendado um teste de usabilidade, pois a estratégia de *Closable Panel* não é familiar aos usuários iniciantes.

A.5.4 Como

Dividir o conteúdo em seções, nomear apropriadamente, e colocar em painéis que aparecem e desaparecem ao clique do mouse ou *link*. Colocar uma flecha, sinal “+”, triângulo, ou “>>” no botão ou link para o usuário poder abrir ou fechar a seção.

Adicionalmente, pode-se usar um painel com barra de rolagem, ou painéis que possibilitem alterar o tamanho deste.

O desenvolvedor pode usar o *Card Stack* dentro de um *Closable Panel*. Pode-se usar quando tem-se numerosas seções, e encaixam em uma organização hierárquica de dois níveis.

Esta técnica existe desde 1993, possivelmente antes. O construtor de interface baseada em Motif chamada UIM/X usou *Closable Panels* com setas para seleção de cor.

A.5.5 Exemplo

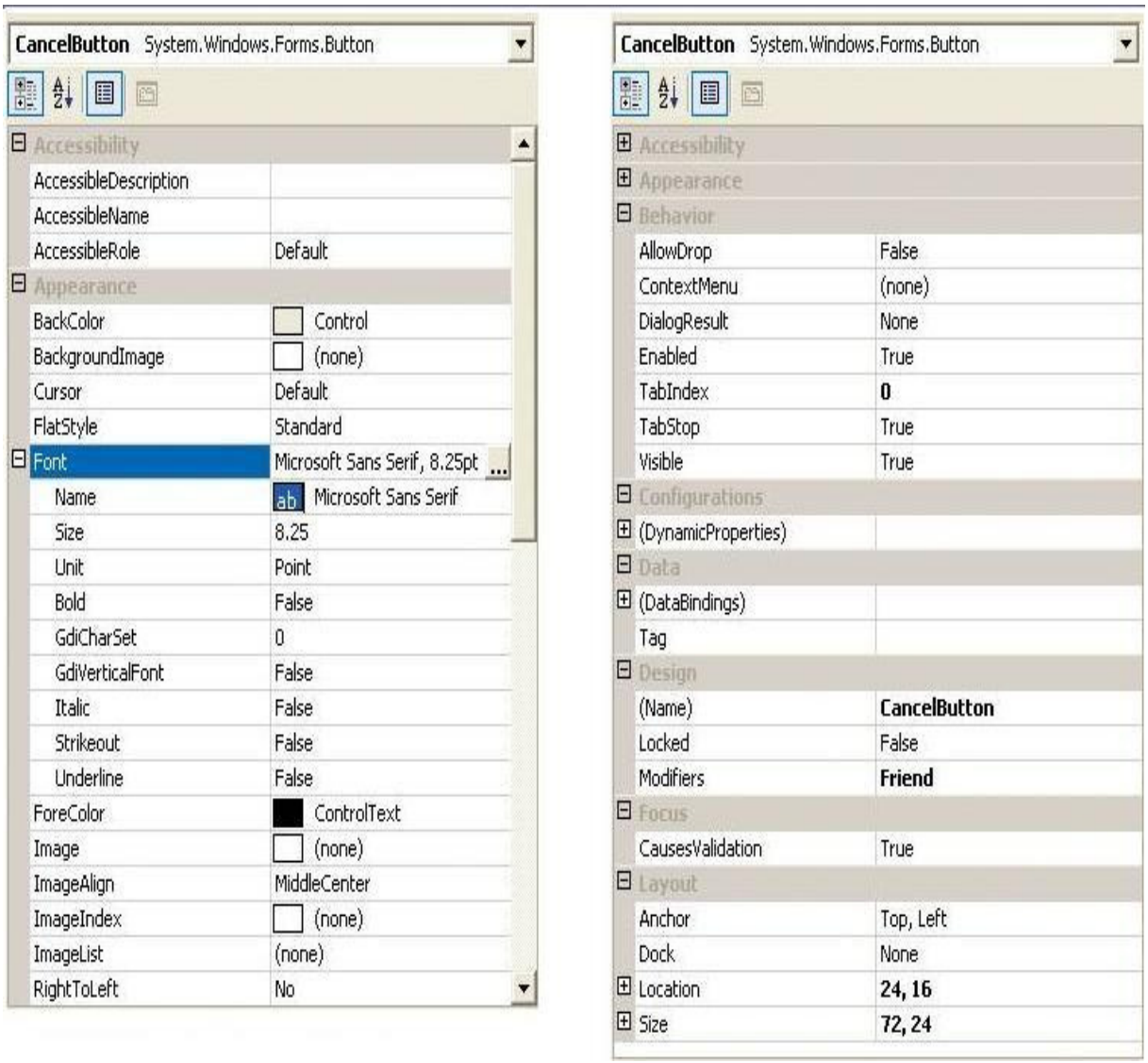


Figura A.5: Propriedades do Visual Studio. Adaptada de Tidwell [63].

A.6 *Liquid Layout*

A.6.1 Descrição

Quando o usuário altera o tamanho da janela, o conteúdo adequa-se ao tamanho da janela, que fica constantemente preenchida.

A.6.2 Quando Usar

O usuário quer o conteúdo em uma janela maior ou menor. Usualmente quando uma página possui muitos textos, ou tabela, ou um editor gráfico. Não funciona tão bem quando o conteúdo for limitado e não puder ser reajustado.

A.6.3 Por quê?

O tamanho de tela, fonte, e outras janelas são definidas pelas limitações e preferências de cada usuário. Este padrão permite que estes itens sejam customizados.

Dar ao usuário um pequeno controle sobre o layout da página faz a interface ser mais flexível sobre condições que modificam-se com freqüência.

A.6.4 Como

Faça o conteúdo da página continuamente preencher a janela quando o tamanho muda. Linhas múltiplas de texto mudam-se para a margem direita.

Páginas web e interfaces similares a ela devem permitir ao conteúdo preencher o novo espaço, e manter a sinalização e dispositivos de navegação ancoradas as margens superior e esquerda. Cores de fundo e padrões devem preencher esse novo espaço, mesmo se o conteúdo não puder ser reajustado.

Quando a janela fica menor que o conteúdo, deve-se colocar barras de rolagem sobre ele e os espaços em branco devem encolher o tanto quanto for necessário.

Se o conteúdo estiver na forma de parágrafos de texto, existe uma regra que diz que a frase torna-se quase ilegível quando muito alongada, forçando a visão do usuário para poder ler todo o conteúdo, o que não acontece quando a frase tem uma limitação de palavras aceitável.

Existem evidências de que linhas de texto com 100 caracteres são mais rápidas para leitura do que linhas curtas, mesmo os usuários preferindo linhas em torno de 55 caracteres (*US Department of Health and Human Services* [66]).

A.6.5 Exemplo

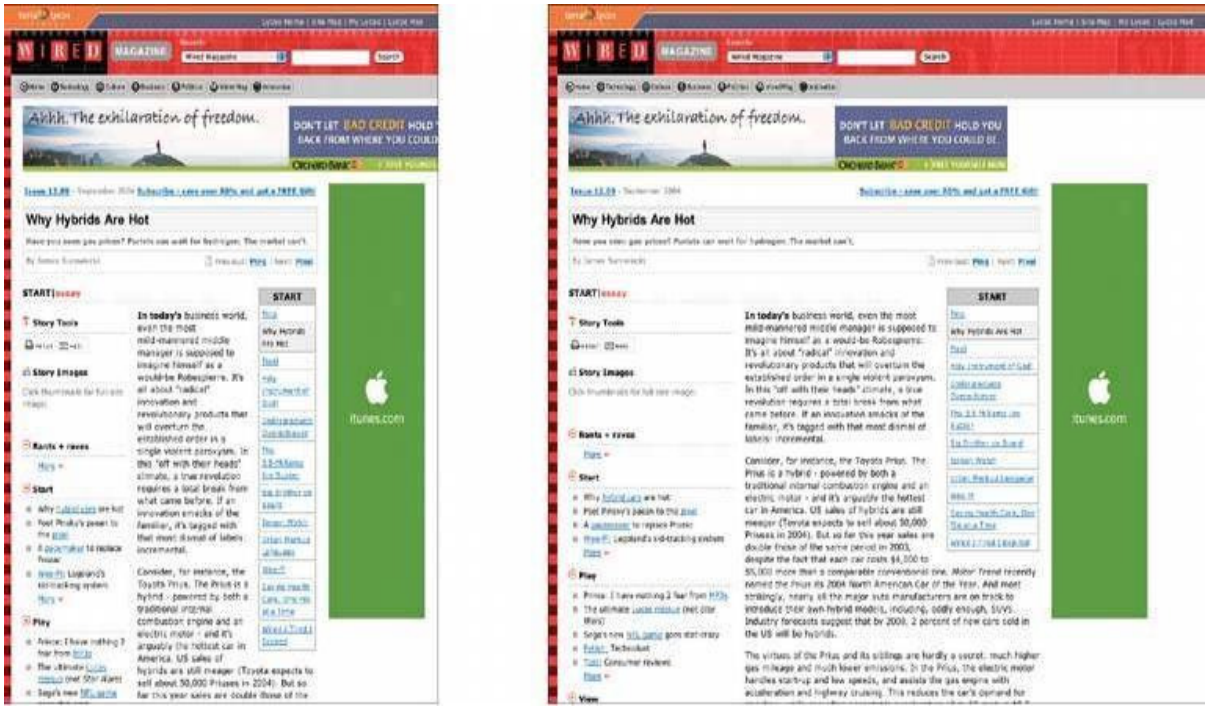


Figura A.6: Site do portal Wired News (Tidwell [63]).

A.7 *Card Stack*

A.7.1 Descrição

Agrupamento de seções de conteúdo, separado por painéis, de modo que somente um fique visível por vez, usando abas ou outras abordagens para que o usuário acesse o conteúdo.

A.7.2 Quando Usar

Quando existe muito material na página, muitos botões de controles ou textos, e a interface não possui uma boa estruturação, o usuário fica distraído. O desenvolvedor pode agrupar o conteúdo através de seções com título, porém, podem ficar tão grandes que não caberão na página de uma só vez. O mais importante a ser citado é a não necessidade do usuário ver mais que uma seção por vez.

A.7.3 Por quê?

A estrutura de conteúdo em abas intituladas é familiar a todos os usuários, e dividido de forma que possa ajudar a parcelar a “digestão” da interface.

A.7.4 Como

Deve-se dividir o conteúdo em tarefas coerentes, que dar um título que seja curto, uma ou duas palavras se possível, e de fácil memorização. O desenvolvedor deve lembrar que se a divisão do conteúdo for feita de modo errado, os usuários perderão tempo procurando pelo item que desejam acessar. Geralmente requerem seis ou mais abas para o padrão ser efetivo.

A.7.5 Exemplo

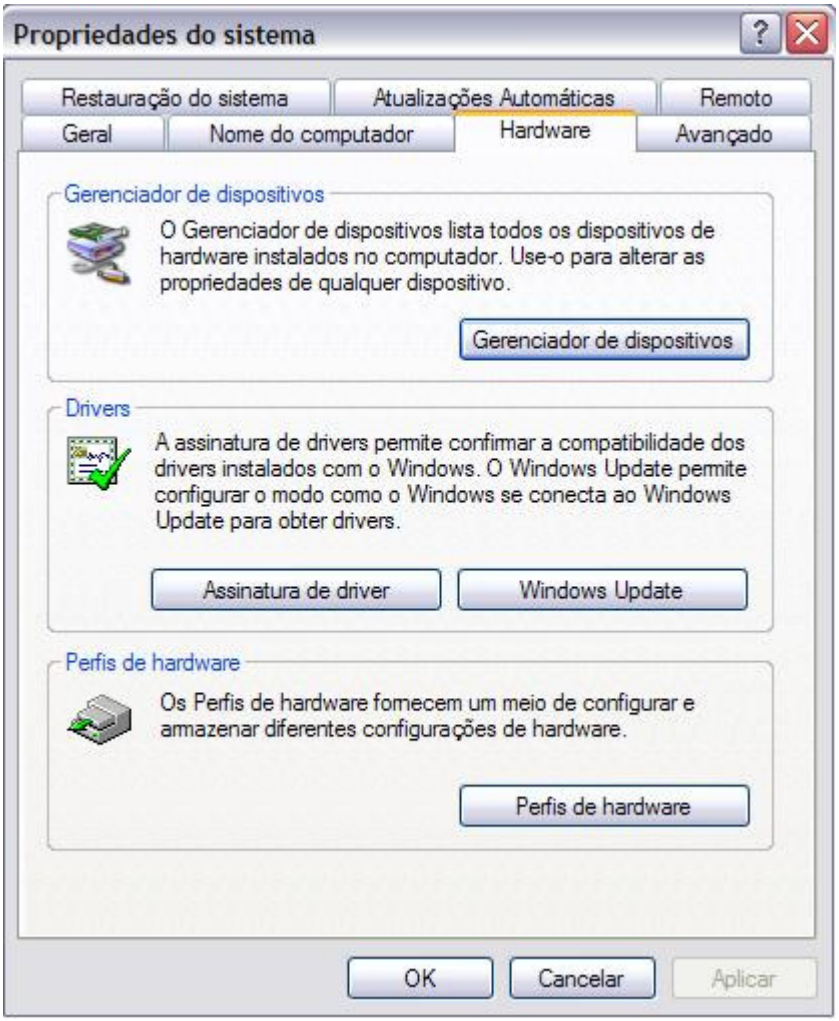


Figura A.7: Propriedades do Sistema Windows XP.

Anexo B

Exemplo de Padrão de Alexander

B.1 *Street Cafe*

88 STREET CAFE**

. . . vizinhanças são definidas por IDENTIFIABLE NEIGHBORHOOD (14); seus pontos de foco são dados por ACTIVITY NODES (30) e SMALL PUBLIC SQUARES (61). Esse padrão, e todos que seguem, dão a vizinhança e seus pontos de foco, sua identidade.

* * *

O *Street Cafe* provê uma configuração única, especial a cidades: um lugar onde pessoas podem sentar e aproveitar, e assistir o mundo passar lá fora.

Cidades consideradas mais humanas são cheias de "*Street Cafes*". Sabe-se que as pessoas gostam de misturar-se, em parques, esquinas, avenidas e "*Street Cafes*".
As pré-condições para que seja um bom estabelecimento parecem ser:

a) o estabelecimento dá a você o direito de sentar-se ali;
b) há algumas coisas que fazem parte: ler jornal, perambular, beber uma cerveja;
c) as pessoas sentem-se seguras para relaxar, distrair-se uns com os outros, e realizar encontros.

Nas cidades européias, existe um *Street Cafe* em cada vizinhança, assim como os postos de gasolina nos Estados Unidos. A existência desses lugares provê união entre a sociedade, transformando em um ambiente familiar. O *Street Cafe* ajuda a aumentar a identidade de uma vizinhança. É uma maneira de um novato na vizinhança pode começar a interagir com o pessoal que está ali há vários anos.

Os ingredientes de um *Street Cafe* de sucesso parecem ser:

a) clientela local estabelecida. Por nome, local e equipe de trabalho, o café é muito mais fixo na vizinhança que está situada;
b) além do terraço aberto à rua, o café pode conter outros espaços, com jogos, cadeiras macias, jornais, revistas. Isso faz com que sua clientela seja diversificada, e o estabelecimento esteja de acordo com estilos sociais diferentes;
c) o café serve comida simples e alguns *drinks*, mas não é um bar. É um lugar que você gostaria de estar pela manhã para iniciar o dia e a noite para tomar uma bebida.

Quando estas condições estão presentes, o estabelecimento oferece algo único à vida das pessoas que as usam: oferece uma maneira de encontro para discussão de variados assuntos, leituras, lugar metade publico e metade privado, e trocas de pensamentos entre pessoas.

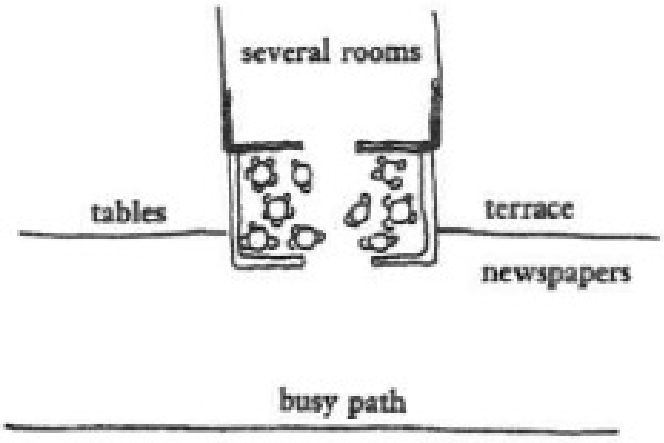
Alexander ainda compara a importância da discussão nos cafés com as instruções que os estudantes recebem na sala de aula, e entrevistou 30 estudantes para saber o motivo das pessoas irem ao café.

Em primeiro lugar, ficou a discussão durante um copo de cerveja e logo após conversar com um pequeno grupo de estudantes. Aparentemente as atividades informais em cafés contribuem para o crescimento intelectual dos estudantes, mais que tradicionais formações educacionais.

Assim como os estudantes, o *Street Cafe* influencia diretamente o crescimento de toda uma vizinhança, fazendo parte da vida de cada pessoa.

Therefore:

Encorajar cafés locais a estabelecer-se em cada vizinhança. Faça-os locais íntimos, com vários cômodos, aberto à rua, onde as pessoas possam sentar para tomar um café ou ver o tempo passar. Construa na frente do café um conjunto de mesas que combinem com o café.



* * *

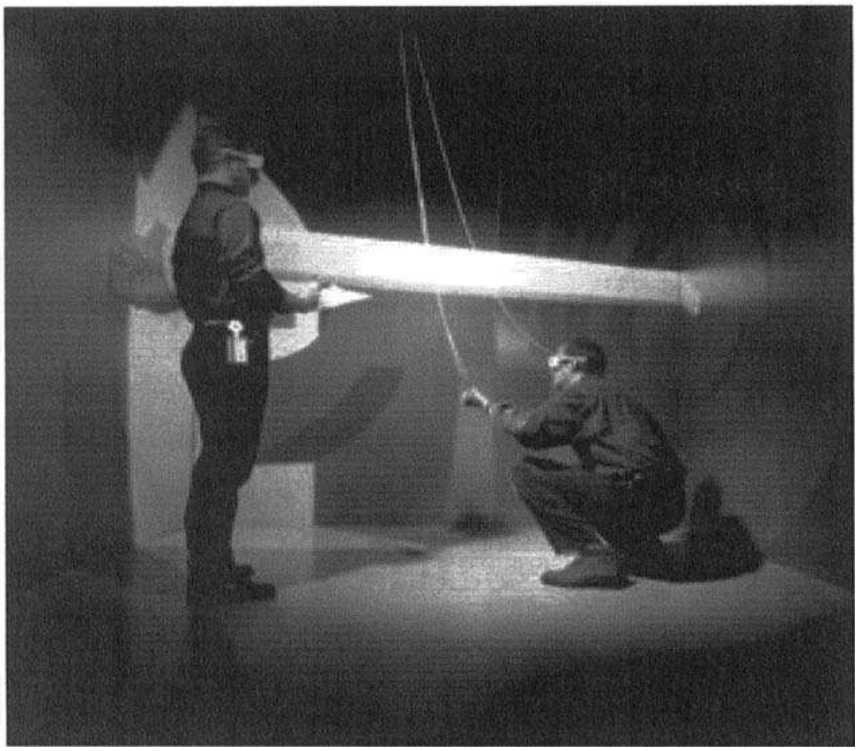
Construa uma abertura entre o terraço e as portas de entrada – OPENING TO THE STREET (165); faça o terraço duas vezes maior que o A PLACE TO WAIT (150) para paradas de ônibus e escritórios; use uma grande variedade de cadeiras e mesas – DIFFERENT CHAIRS (251); e dê ao terraço uma visão da esquina da rua – STAIRS SEATS (125), SITTING WALL (243), talvez um CANVAS ROOF (244). Para a forma do prédio ou estabelecimento, comece com o BUILDING COMPLEX (95). . . .

Anexo C

Exemplo de Padrão de Borchers

C.1 *Immersive Display*

H13 IMMERSIVE DISPLAY *



CAVE no *Ars Electronica Center Linz.*

. . . você decidiu criar uma exibição que várias pessoas podem experimentar simultaneamente - COOPERATIVE EXPERIENCE (H3). Agora você precisa encontrar uma maneira de projetar a saída deste sistema.

◊◊◊

Uso típico destes cenários em sistemas computacionais pode envolver interação humana com o computador em qualquer tempo, e o sistema é apenas uma pequena parte do ambiente do usuário. Mas geralmente, as exibições são visitadas por grupos de pessoas, e quando usuários interagem com as exibições, estão prontas para entrar no mundo da exibição.

A instalação *CAVE* no *Ars Electronica Center* em Linz usa projeções do tamanho das paredes envolvendo os visitantes, a fim de imergir os visitantes em uma realidade virtual. Vidros especiais sincronizam com estes *displays* para criar uma impressão 3D.

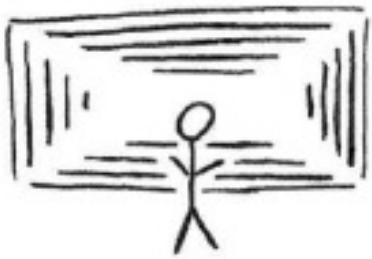
Virtual Vienna usa uma tela projetada no fundo de tamanho aproximado de 1,6 metros, e os usuários ficam a esta mesma distância da tela. Essa distância preenche a maioria do campo visual quando olha-se para a tela, e ajuda as pessoas a sentir que estão no lugar que o panorama é mostrado.

Personal Orchestra usa uma tela com área maior, cerca de 2,5 metros, com a mesma distância para visualização. Isto faz com que o espectador tenha a impressão de esta defronte a Filarmônica de Vienna.

Com esses sistemas, os largos *displays* permitem que vários observadores observem a exibição, sem ser um usuário direto do sistema.

Therefore:

Prefira um exibidor de imagens de larga escala, com no mínimo 1,5 metros de tamanho, a vários monitores menores, e àqueles que permitam apenas um espectador a participação da exibição. A distância entre o espectador e a tela deve ser igual ao tamanho do *display*.



◇◇◇

Se você esconder a tecnologia do *display*, ele torna-se uma "imagem mágica" - INVISIBLE HARDWARE (H14). . . .

Anexo D

Padrões e Linguagens de padrões

Linguagens de padrões disponíveis na internet para diversos domínios. Retirado de Henninger e Corrêa [33].

Title	Source	# of Patt.	Year
Patterns in Interaction Design	http://www.welie.com/	146	2005
"Analysis Patterns: Reusable Object Models"	"Analysis Patterns: Reusable Object Models"	95	1996
"Designing Interfaces: Patterns for Effective Interaction Design"	http://www.mit.edu/~jtidwell/common_ground_onefile.html	94	2005
Ajax Design Patterns	http://ajaxpatterns.org	70	2006
"Requirements Patterns and Antipatterns: Best (and Worst) Practices for Defining Your Requirements"	http://www.tabletuml.com/RPandAP/default.aspx	69	2007
"Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions"	http://www.eaipatterns.com/toc.html	65	2003
Yahoo! Design Pattern Library	http://developer.yahoo.com/ypatterns/	63	2005
"Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects"	"Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects"	55	2004
"J2EE Antipatterns"	"J2EE Antipatterns"	52	2003
"Patterns of Enterprise Application Architecture"	http://www.martinfowler.com/eaaCatalog/	51	2002
"Object Oriented Reengineering Patterns"	http://www.iam.unibe.ch/~scg/OORP/book.html	49	2002
A Generative Development-Process Pattern Language	http://users.rcn.com/jcoplien/Patterns/Process/index.html	48	1995
UML Pattern Language	http://www.ncc.up.pt/~zp/aulas/0607/es/geral/bibliografia/UML%20Pattern%20Language.pdf	46	2000
"Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems"	Addison Wesley Professional	44	2002
"AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis"	John Wiley & Sons	42	1998
WikiPatterns	http://www.wikipatterns.com/	42	2007
"Patterns for Effective Use Cases"	Addison Wesley Professional	32	2002
"Enterprise Solution Patterns Using Microsoft .NET Version 2.0: Patterns & Practices"	Microsoft Press	32	2004
"Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware"	John Wiley & Sons	32	2004
XML Design Patterns	http://www.xmlpatterns.com/	28	2000
Hypermedia Design Patterns Repository	http://www.designpattern.lu.unisi.ch/index.htm	28	1997
Embedded Design Patterns	http://www.eventhelix.com/RealtimeMantra/Patterns/	28	2004
"Small Memory Software: Patterns for Systems with Limited Memory"	http://hillside.net/patterns/books/Details/056.htm	27	2001
A Pattern Language for Pattern Writing	http://hillside.net/patterns/writing/patternwritingpaper.htm	26	1997
Experiences -- A Pattern Language for User Interface Design	http://www.maplefish.com/todd/papers/Experiences.html	26	2003
Data Access Patterns: Database Interactions in Object-Oriented Applications"	http://helloworld.siteburg.com/content/databases/db2/0131401572_toc.html	25	2003
GoF Patterns	http://www.vico.org/pages/PatronsDisseny.html	23	1995
Caterpillar's Fate: A Pattern Language for the Transformation from Analysis to Design	http://c2.com/ppr/catsfate.html	21	1995
User Interface Design Patterns	http://www.cs.helsinki.fi/u/salaakso/patterns/index.html	21	2003
Workflow Patterns	http://www.workflowpatterns.com/patterns/index.php	21	2000

Title	Source	# of Patt.	Year
Patterns for System Testing	"Pattern Languages of Program Design 3"	20	1997
Web Design Patterns Library	http://harbinger.sims.berkeley.edu/ui_designpatterns/webpatterns2/webpatterns/home.php	20	2006
A Pattern Language for Writers' Workshops	users.rcn.com/jcoplien/Patterns/WritersWorkshop/	19	1999
"Microsoft Integration Patterns"	http://download.microsoft.com/download/a/c/f/acf079ca-670e-4942-8a53-e587a0959d75/IntPatt.pdf	18	2004
Patterns Systems for Hypermedia	http://www-di.inf.puc-rio.br/schwabe//papers/PloP97.pdf	18	1997
POSA 1 Patterns	http://www.vico.org/pages/PatronsDisseny.html	17	1996
POSA 2 Patterns	"Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects "	17	2000
RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development	http://www2.umassd.edu/SWPI/ATT/pattern/rapel.html	17	1995
Understanding and Using the ValueModel Framework in VisualWorks Smalltalk	http://c2.com/ppr/vmodels.html	17	1994
An Input and Output Pattern Language: Lessons from Telecommunications	http://hillside.net/plop/plop98/final_submissions/P31.pdf	17	1999
New Clients with Old Servers: A Pattern Language for Client/Server Frameworks	http://citeseer.ist.psu.edu/156837.html	16	1995
Lazy Optimization: Patterns for Efficient Smalltalk Programming	"Pattern Languages of Program Design 2"	16	1996
EPISODES: A Pattern Language of Competitive	http://c2.com/ppr/episodes.html	16	1996

Referências Bibliográficas

[1] ACM SIGCHI. **Curricula for Human-Computer Interaction**. Nova Iorque, EUA: ACM Press, 1992.

[2] ADOBE. **Adobe – Comunidades**. Disponível em: <http://www.adobe.com/br/communities/>. Acesso em 06/11/2008.

[3] ALEXANDER, C. ***The Timeless Way of Building***. Nova Iorque: Oxford University Press, 1979.

[4] ALPERT, S. R. Getting Organized: Some Outstanding Questions and Issues Regarding Interaction Design Patterns. In: WORKSHOP ON "PERSPECTIVES ON HCI PATTERNS", no CHI, 20., 2003. Flórida. **Proceedings...** Flórida, EUA: ACM Press, 2003.

[5] APPLE. **Apple discussions**. Disponível em: <http://discussions.apple.com/index.jspa>. Acesso em 06/11/2008.

[6] APPLE. **Apple Human Interface Guidelines**. Disponível em: <http://developer.apple.com/DOCUMENTATION/UserExperience/Conceptual/AppleHIGuidelines/OSXHIGuidelines.pdf>. Acesso em 16/08/2008.

[7] APPLETON, B. **Patterns and Software: Essential Concepts and Terminology**. Disponível em <http://www.cmcrossroads.com/bradapp/docs/patternsintro.html>. Acesso em 03/04/2008.

[8] BAINBRIDGE, W. S. **Berkshire encyclopedia of human-computer interaction**. 1st edition. Great Barrington, Massachusetts, EUA: Berkshire Publishing group, 2004.

[9] BAYLE *et al*. BAYLE, E.; BELLAMY, R.; CASADAY, G.; ERICKSON, T.; FINCHER, S.; GRINTER, B.; GROSS, B.; LEHDER, D.; MARMOLIN, H.;

MOORE, B.; POTTS, C.; SKOUSEN, G.; THOMAS, J. Putting It All Together: Towards a Pattern Language for Interaction Design: A CHI 97 Workshop. ***SIGCHI Bulletin***, Nova Iorque, EUA, v. 30, n. 1, p. 17-23, jan., 1998.

[10] BECK, K.; CUNNINGHAM, W. Using pattern language for object-oriented programs. In: OOPSLA'87, 2., 1987, Orlando. **Proceedings...** Orlando, EUA: Tektronix Inc., 1987.

[11] BITENCOURT, R. S. **Avaliação da forma tradicional e macro ergonômica de identificação de requisitos para a concepção de projetos de software, sob o foco da qualidade em uso**. Porto Alegre: UFRGS, 2003. Dissertação.

[12] BORCHERS, J. O. CHI Meets PLoP: An Interaction Patterns Workshop. ***SIGCHI Bulletin***, Nova Iorque, EUA, v. 32, n. 1, p. 9-12. jan., 2000.

[13] BORCHERS, J. O. **A pattern approach to Interaction Design**. *AI & Society*, 15, (2001), p. 359 - 376.

[14] BORCHERS, J. O. Interaction Design Pattern: Twelve Theses. In: WORKSHOP ON PATTERN LANGUAGE FOR INTERACTION DESIGN, no CHI, 17., 2000, Hague. **Proceedings...** Hague, Holanda: ACM Press, 2000.

[15] BORCHERS, J. O. **A Pattern Approach to Interaction Design**. Chichester: John Wiley & Sons Ltd., 2001.

[16] BROWN, J. Methodologies for the Creation of Interactive Software. Relatório Técnico – Departamento de Ciência da Computação, Victoria University of Wellington, Wellington, Nova Zelândia. 1996.

[17] CAMPOS, P. **Interacção Homem-Máquina**. Disponível em <http://dme.uma.pt/edu/ihm/slides/IHM1%20-%20Introdu%E7%E3o%20-%20Cor.pdf>. Acesso em 06/03/2008.

[18] DEARDEN, A.; FINLAY, J. **Pattern languages in HCI: A critical review**. Human- Computer Interaction, 21, 49-102. 2006.

[19] DIAS, C. **Usabilidade na web: criando portais mais acessíveis**. Rio de Janeiro: Alta books, 2003.

[20] DIX, A. J.; FINLAY, J. E.; ABOWD, G. D.; BEALE, R. **Human-Computer Interaction.** 2.ed. Inglaterra: Prentice Hall Europe, 1998. 638 p.

[21] ERICKSON, T. Interaction pattern languages: A *Lingua Franca* for interaction design? In: *UPA'98 Usability Professionals' Association Conference (invited talk),* Washington. **Proceedings…** Washington, DC, June 24**,** 1998.

[22] ERICKSON, T. Pattern Languages as Language. In: CHI, 2000, Hague. **Proceedings…** Hague, Holanda: ACM Press, 2000.

[23] FINCHER, S. What is a Pattern Language?. In: CHI, 16., 1999,Pittsburgh. **Proceedings...** Pittsburgh, EUA: ACM Press. 1999.

[24] FINCHER, S.; WINDSOR, P. Why patterns are not enough: Some suggestions concerning an organizing principle for patterns of UI design. In: *CHI 2000 Workshop on Pattern Languages for Interaction Design: Building Momentum.* **Proceedings…** The Hague, Holanda, 2000.

[25] FINCHER, S. Patterns for HCI and cognitive dimensions: Two halves of the same story? In: Psychology of Programming Interest Group. **Proceedings…** Brunel University, Londres, Reino Unido, 2002.

[26] FINCHER, S. **HCI Pattern-Form Gallery**. Disponível em <http://www.cs.kent.ac.uk/people/staff/saf/patterns/gallery.html>. Acesso em 17/05/2008.

[27] FINCHER *et al.* FINCHER, S.; FINLAY, J.; GREENE, S.; JONES, L.; MATCHEN, P.; THOMAS, J.; MOLINA, P. J. Perspectives on HCI Patterns: Concepts and tools. In: CHI 2003, Florida. **Proceedings...** Florida, EUA: ACM Press, 2003. v. 2, 1069 p. p.1044-1045.

[28] GAMMA *et al.* GAMMA, E.; HELM, R.; JOHNSON, R.; E VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Boston, EUA: Addison-Wesley, 1995. 395 p.

[29] GOULD et al. GOULD, J.D.; BOIES, S.J.; UKELSON, J. How to Design Usable Systems. Handbook of Human-Computer Interaction, M.G.Helander, T.K.Landauer, P.V.Prabhu (eds.) **Elsevier Science**, pp. 231-254. 1997.

[30] GOULD, J.D., LEWIS, C. Designing for Usability – key principles and what designers think. ***Communications of the ACM***, 28,300-311.1985.

[31] GRANLUND, A.; LAFRENIERE D. PSA: A pattern supported approach to the user interface design process. In: UPA'99 USABILITY PROFESSIONALS' ASSOCIATION CONFERENCE, 1999, Scottsdale. **Proceedings...** Scottsdale, AZ, June 29-July 2, 1999.

[32] GULLIKSEN *et al.* GULLIKSEN, J.; LANTZ, A.; BOIVIE, I. User Centered Design – Problems and Possibilities: a summary of the 1998 PDC & CSCW workshop. ***ACM SIGCHI Bulletin***, Nova Iorque, EUA, v. 31, n. 2, p. 25-35, abr., 1999.

[33] HENNINGER, S.; CORRÊA, V. Software pattern communities: Current practices and challenges. In: 14th Conference on Pattern Languages of Programs (PLoP 07), Monticello. **Proceedings…** Monticello, IL, 2007.

[34] IBM. **Common user acess**. Disponível em: <http://www.research.ibm.com/journal/sj/273/ibmsj2703E.pdf>. Acesso em 16/09/2008.

[35] ISO/IEC. *9241-11* - **Guidance on Usability**. ISO/IEC 9241, 1998.

[36] ISO/IEC. *13407* – **Human-Centered Design Processes for Interactive Systems**. ISO/IEC 13407, 1999.

[37] ISO/IEC. 14915 - **Software ergonomics for multimedia user interfaces.** ISO/IEC 14915, 2002.

[38] JOKELA *et al.* JOKELA, T.; IIVARI, N.; MATERO, J.; KARUKKA, M. The Standard of User-Centered Design and the Standard Definition of Usability: Analyzing ISO 13407 against ISO 9241-11. In: CLIHC - LATIN AMERICAN CONFERENCE ON HUMAN-COMPUTER INTERACTION., 1., 2003, Rio de Janeiro. **Proceedings...** Nova Iorque, EUA: ACM Press, 2003, 265 p. p. 53-60.

[39] KUMAR, R. R. **Human Computer Interaction.** 1st edition. Reino Unido: Laxmi Publications, 2005.

[40] LÉVY, P. **As tecnologias da inteligência: o futuro do pensamento na era da informática**. Rio de janeiro, 1993.

[41] MATLAB. **Matlab Central**. Disponível em: <http://www.mathworks.com/matlabcentral/>. Acesso em 06/11/2008.

[42] MORAES, A; DRESCH, A. “HUMAN COMPUTER INTERACTION” NAVEGANDO OU DIALOGANDO. In: Encontro Carioca de Ergonomia, 1994. **Proceedings...** Rio de janeiro: UERJ, 1994.

[43] NANNI, P. **Human-Computer Interaction: Principles of Interface Design**. 2004. Disponível em <http://www.vhml.org/theses/nannip/HCI_final.htm>. Acesso em 11/06/2008.

[44] NIELSEN, J. ***Usability Engineering***. California, EUA: Academic Press, 1993.

[45] NORMAN, D. A.; DRAPER, S. W. ***User Centered System Design: New Perspectives on Human-Computer Interaction***. Hillsdale, EUA: Lawrence Erlbaum Associate Publishers, 1986. 526 p.

[46] OPEN GROUP. **Motif**. Disponível em: <http://www.opengroup.org/motif/>. Acesso em 06/11/2008.

[47] PEMBERTON, L. The promise of pattern languages for interaction design. In: *Human Factors Symposium,* Loughborough. **Proceedings…** Loughborough, Reino Unido, 2000.

[48] PREECE *et al.* PREECE, J. ; ROGERS Y.; SHARP H.; BENYON D.; HOLLAND S.; CAREY T. **Human-Computer Interaction**. Reading, Mass., Addison-Wesley Publishing, 1994.

[49] PREECE *et al.* PREECE, J.; ROGERS Y.; SHARP H. **Design de interação: além da interação humano-computador**. Porto Alegre: Bookman, 2005.

[50] RIHLE, D.; ZÜLLIGHOVEN, H. **Understanding and Using Patterns in Software Development.** ***Theory and Practice of Object Systems***. Boston, v. 2, n. 1, p. 3-13, 1996.

[51] ROCHA, H. V. DA; BARANAUSKAS, M. C. C. **Design e Avaliação de Interfaces Humano-Computador**. Campinas: NIED/Unicamp, 2003.

[52] SEFFAH, A.; JAVAHERY, H. On the usability of usability patterns. In: Workshop Entitled Patterns in Practice, CHI. **Proceedings…** Mineapolis, 2002.

[53] SHNEIDERMAN, B. **Designing the User Interface: Strategies for Effective Human-Computer Interaction**. 3ª ed., EUA: Addison Wesley Longman Inc., 1998.

[54] SILVA, E. **Sistemas Interativos**. Ouro Preto: Universidade Federal de Ouro Preto, 2006.

[55] SINNIG, D. **The complicity of patterns and model-based UI development**. Montreal, Canadá: Concordia University, 2004. Dissertação.

[56] SOARES, L. G. **Avaliação de Usabilidade, por meio do índice de satisfação dos usuários, de um *software* gerenciador de *websites***. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2004. Monografia.

[57] SOMMERVILLE, I. **Engenharia de Software**. 8ª ed. São Paulo: Pearson Addison-Wesley, 2007.

[58] SUN. **Open Look**. Disponível em: <http://docs.sun.com/app/docs/doc/801-6735/6i13eq5el?a=view>. Acesso em 04/11/2008.

[59] TIDWELL, J. Interaction Design Patterns. In: PATTERN LANGUAGES OF PROGRAMS – PLOP. **Proceedings…** Monticello, EUA, 1998.

[60] TIDWELL, J. **Common Ground: a Pattern Language for Human- Computer Interface Design**. Disponível em <http://www.mit.edu/~jtidwell/interaction_patterns.html>. Acesso em 10/05/2008.

[61] TIDWELL, J. **User Interface Patterns and Techniques**. Disponível em <http://time-tripper.com/uipatterns>. Acesso em 18/04/2008.

[62] TIDWELL, J. Perspectives on HCI Patterns: Concepts and Tools. Position Paper. In: WORKSHOP ON "PERSPECTIVES ON HCI PATTERNS", CHI 2003. Flórida. **Proceedings...** Flórida, EUA: ACM SIGCHI, 2003.

[63] TIDWELL, J. Designing Interfaces: Patterns for Effective Interaction Design. 1st edition. O'Reilly Media: Sebastopol, CA, EUA: 2005.

[64] TODD *et al*. TODD, E.; KEMP, E.; PHILLIPS, C. What makes a good user interface pattern language? In: Conferences in Research and Practice in Information Technology (CRPIT '28). **Proceedings…** Darlinghurst, Austrália, 2004.

[65] UBUNTU. **The Ubuntu Community**. Disponível em: <http://www.ubuntu.com/community>. Acesso em 06/11/2008.

[66] US DEPARTMENT OF HEALTH AND HUMAN SERVICE. **The Research-Based Web Design & Usability Guidelines**. Disponível em: <http://usability.gov/pdfs/guidelines_book.pdf>. Acesso em 06/11/2008.

[67] VLISSIDES, J. Patterns: The Top Ten Misconceptions. **Object Magazine**, 1997.

[68] WANIA, C. E. **Examining the Impact of an Information Retrieval Pattern Language on the Design of Information Retrieval Interfaces**. Drexel University: 2008. Tese.

[69] WELIE, M. VAN; VEER, G.C. VAN DER. Pattern languages in interaction design: Structure and organization. In: Interact 2003. **Proceedings…** Zurique, Suiça, 2003.

[70] WELIE *et al.* WELIE, M. VAN; VEER, G. C. VAN DER; ELIËNS, A. Patterns as Tools for User Interface Design. In: INTERNATIONAL WORKSHOP ON TOOLS FOR WORKING WITH GUIDELINES. **Proceedings…** Biarritz, França, 2000.

[71] WELIE, M. VAN. **Pattern in Interaction Design**. Disponível em <http://www.welie.com/patterns>. Acesso em 17/05/2008.

[72] WINCKLER, M. A. A. **Proposta de uma metodologia para avaliação de usabilidade de interfaces WWW**. Porto Alegre: UFRGS, 1999. Monografia.