UNIVERSIDADE FEDERAL DE PERNAMBUCO

Centro de Informática
Graduação em Ciência da Computação

Trabalho de Graduação

PROTOTIPAÇÃO RÁPIDA NO CICLO DE DESIGN ITERATIVO DE APLICAÇÕES MULTIMÍDIA PARA FORMAÇÃO DE PROFESSORES

Filipe Levi Barros de Azevedo

Orientador: Alex Sandro Gomes
Co-orientadora: Stephania Padovani

Recife, 2005

UNIVERSIDADE FEDERAL DE PERNAMBUCO

Centro de Informática

Filipe Levi Barros de Azevedo

PROTOTIPAÇÃO RÁPIDA NO CICLO DE DESIGN ITERATIVO DE APLICAÇÕES MULTIMÍDIA PARA FORMAÇÃO DE PROFESSORES

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Alex Sandro Gomes

Co-orientadora: Stephania Padovani

Recife, 2005

Dedicatória

Dedico este trabalho à memória de
Edson Rosendo de Azevêdo Filho
e Rubem de Lima Barros.

Agradecimentos

A Edson e Rute, meus pais, e a toda minha família, pelos ensinos e suporte.

A Juliana, minha auxiliadora idônea, pelo amor, incentivo e orações.

Aos professores Alex e Stephania, pela orientação e confiança.

"Man is the only creature that consumes without producing. He does not give milk, he does not lay eggs, he is too weak to pull the plough, he cannot run fast enough to catch rabbits. Yet he is lord of all the animals. He sets them to work, he gives back to them the bare minimum that will prevent them from starving, and the rest he keeps for himself.

[…]

But they had not gone twenty yards when they stopped short. An uproar of voices was coming from the farmhouse. They rushed back and looked through the window again. Yes, a violent quarrel was in progress. There were shoutings, bangings on the table, sharp suspicious glances, furious denials. The source of the trouble appeared to be that Napoleon and Mr. Pilkington had each played an ace of spades simultaneously. Twelve voices were shouting in anger, and they were all alike. No question, now, what had happened to the faces of the pigs. The creatures outside looked from pig to man, and from man to pig, and from pig to man again; but already it was impossible to say which was which."

— GEORGE ORWELL, *Animal Farm: a Fairy Story*, 1945.

Resumo

A tecnologia vem sendo utilizada na educação como ferramenta para potencialização do modelo didático. Contudo, no Brasil, seu uso ainda está restrito ao ambiente de sala-de-aula: nossa tecnologia não tem contemplado O processo de capacitação do professor. Neste trabalho, propomos uma metodo logia para o desenvolvimento de software educativo, cujo ciclo de design iterativo envolve prototipação rápida com professores. Através dos resultados alcançados em uma situação real de mercado, argumentamos que tal técnica, integrada à colaboração com esp ecialistas em educação, garante usabilidade e qualidade de conteúdo pedagógico ao produto final.

Palavras-chave: prototipação rápida, design iterativo, learnware, formação de professores.

Abstract

Technology is being used in education as a tool for improving the didactic model. However, in Brazil, its use is still restricted to the classroom environment: our technology does not cover the teacher's training process. This work proposes a learnware development methodology whose iterative design cycle involves rapid prototyping with teachers. Considering the current achievements in a real business situation, we argue that such a technique, placed togheter with the colaboration of education experts, ensures usabili ty and pedagogical contents quality to the final product.

Keywords: rapid prototyping, iterative design, learnware, teacher's training.

Sumário

Lista de figuras

CAPÍTULO 1

Introdução

"Never regard your study as a duty, but as the enviable opportunity to learn to know the liberating influence of beauty in the realm of the spirit for your own personal joy and to the profit of the community to which your later work belongs."

— ALBERT EINSTEIN, *Universidade de Princeton*, 1933.

Pesquisa realizada em 2003 pelo Instituto Nacional de Estudos e Pesquisas (INEP) revelou que alunos do ensino básico manifestam dificuldades de raciocínio, principalmente quando o assunto é matemática. De acordo com a pesquisa, na região Nordeste, 75% dos alunos que estão prestes a concluir o ensino médio apresentam estágio crítico ou muito crítico em matemática; no Norte, o índice sobre para 82,11%. O governo federal aponta várias causas responsáveis por tal situação, dentre as quais a *capacitação de professores* figura como um dos principais problemas. No mês de maio do ano corrente, o ex-ministro da Educação, Tarso Genro, lançou o Plano de Qualidade para Educação Básica, cujo objetivo é elevar a qualidade do ensino básico no Brasil. O plano é resultado de três meses de avaliação e diagnóstico da situação do ensino básico no país. O projeto apresenta propostas estratégicas para a formação de professores, no ano que foi instituído pelo MEC (Ministério da Educação) como o da qualidade social da educação básica no Brasil.

A informática tem papel fundamental na reestruturação desse cenário. Abranches (2003) constata que o processo de ampliação da presença das tecnologias de informação e comunicação atinge várias áreas da sociedade mundial e do conhecimento. O autor completa que " *também a educação, mesmo que para alguns estudiosos de forma tardia, começa a ser profundamente afetada não só nos seus métodos como nos seus objetivos, com a presença cada vez mais freqüente de computadores no seu dia-a-dia*". Como ator-chave na educação, também o professor foi influenciado pela chegada da tecnologia nas escolas. Dimenstein (1997) nos alerta que "*hoje, o profissional que não se mantém atualizado com novos softwares, sistemas e tecnologias corre o risco de se ver completamente defasado com poucos anos de formado, necessitando adotar hábitos de aprendizagem permanente para poder continuar capaz de acompanhar as transformações do mercado*" (p. 10). Babin e Kouloumdjian (1989) defendem o apoio das diversas mídias nesse processo e acrescentam que " *é preciso que filme, televisão, audiovisual e imagem entrem na formação, às vezes como auxiliares na exploração intelectual magistral, às vezes inteiramente como outra maneira completamente diversa de compreender pelo prazer, pela arte, pela aproximação sensorial e intuitiva*" (p. 175).

Entretanto, estudos conduzidos por Rosen e Weil (1995) revelam, entre outras constatações, que — embora as escolas, de forma geral, estejam equipadas com computadores — muitos professores são 'tecnofóbicos', particularmente aqueles dos ensinos fundamental e médio. Mooij e Smeets (2001) alertam que, "*se os professores não estiverem confiantes em sua própria habilidade ou competência para utilizar computadores, isto pode criar obstáculos ao seu desejo de introduzir tecnologia em sua sala-de-aula*". Os autores ainda reportam que a razão mais importante que professores mencionaram em uma pesquisa internacional para não utilizarem tecnologias de informação e comunicação é que estas não lhes são familiares e eles não se sentem seguros em utilizá-las. Zhao e Cziko (2001) denominam isso de 'princípio do controle' — *i.e.* o professor precisa conhecer a tecnologia e sentir-se confiante no seu uso, como fator psicológico de motivação e capacitação.

Segundo Nascimento (2001), é essencialmente importante que haja, num primeiro momento, uma conscientização do professor acerca das potencialidades existentes tanto no computador quanto nele próprio. O professor deve compreender que, no contexto pedagógico, o computador apenas se tornará uma ferramenta realmente útil quando puder interagir com sua mente criativa. Babin e Kouloumdjian (1989) argumentam que " *a dupla homem-máquina torna-se inteligente não por causa da máquina, mas por causa do homem*", no que Lèvy (1993) denomina de 'ecologia cognitiva'. De acordo com o último autor, a convivência entre homem e

máquina apresenta-se como necessária ao fortalecimento de suas próprias relações, pois as tecnologias intelectuais sempre sinalizam uma função cognitiva do homem e contribuem para a contínua reconstrução do seu saber, sendo o diálogo uma constante entre os dois.

"*Entretanto, o que se vê é justamente uma ênfase maior sobre a máquina e em seu poder, surgindo o medo e a insegurança como conseqüências desta possível aproximação*" — afirma Nascimento (2001). Rosen e Weil (1995) concluem que uma das principais causas do medo da tecnologia demonstrado pelo professor é sua preocupação excessiva em *aprender a usar* computadores e softwares. Neste ponto, percebe-se que a baixa qualidade dos softwares educativos para formação de professores tem contribuído para o aumento dessa resistência por parte destes últimos. Tais sistemas são, geralmente, desenvolvidos segundo modelos e processos cujo foco não são os docentes e suas necessidades como usuários de software educacionais.

Lage, Zubenko e Cataldi (2001) enfatizam que "*o ponto inicial* [no design de *learnware*, *i.e.* sistemas cujo propósito maior é ensinar algo] *é detectar a necessidade educacional para o desenvolvimento do programa*". Os autores esclarecem ainda que o conceito de 'qualidade de *learnwares*' está relacionado ao fato de que este tipo de programa possui características bastante particulares, de acordo com seus objetivos curriculares e as necessidades específicas de seu público-alvo. É justamente por isso — afirmam os mesmos autores — que auferir a qualidade de softwares educativos é um processo que consiste em determinar seu grau de adaptação a um contexto específico, para o qual converge uma série de variáveis. Além dos atributos de currículo e audiência, outros exemplos de tais características seriam estilo de ensino e faixa etária, que requerem uma análise própria.

Ward (1994) já enfatizava a necessidade de haver estudos que demonstrassem que testes com usuários durante o processo de design forneceriam informações que resultariam em um melhor design. O autor sugere ainda que tais testes estariam dentro dos recursos e capacidades de designers industriais. Castro e Aguiar (1999) constatam que as aplicações têm incorporado recursos das mais variadas mídias de comunicação e ampliado suas áreas de utilização com grande rapidez. Tal fato tem gerado um lapso na esfera da Engenharia de Software, pois esta não consegue elaborar procedimentos que acompanhem com a mesma velocidade e aplicabilidade a utilização desses recursos. Sendo assim, neste trabalho, propomos uma metodologia centrada no usuário para o desenvolvimento de *learnware* (em particular, aplicações multimídia para a formação de professores), investigando os principais benefícios que a técnica de prototipação rápida pode trazer ao design iterativo de tais sistemas.

Este material está organizado da seguinte forma: no capítulo 2, trataremos sobre o conceito de design centrado no usuário — em que as necessidades, desejos e limitações dos usuários finais de um sistema são extensamente con siderados em cada estágio do processo de design. Discutiremos ainda as três dimensões do design centrado no usuário: design participativo, testes com usuários e design iterativo. Já no capítulo 3, apresentaremos o método de prototipação rápida, discutindo as vantagens de se elaborar protótipos com níveis crescentes de fidelidade, algumas estratégias de design para o ciclo de prototipação e a técnica que utiliza protótipos de papel. No capítulo 4, revisaremos alguns processos para o desenvolvimento de softwa re educativo e proporemos uma metodologia baseada em análise de mercado, entrevistas com especialistas e análise de competidores, cujo elemento -chave é um ciclo de prototipação rápida com testes de usabilidade junto a usuários reais. Descreveremos, no capí tulo 5, o projeto *Mídias Educativas*, que tem utilizado e refinado a metodologia proposta e representa o estudo de caso para a técnica de prototipação rápida. Por fim, no capítulo 6, concluiremos o presente trabalho, relatando os resultados obtidos e aponta ndo a necessidade e relevância de futuros trabalhos.

CAPÍTULO 2

Design centrado no usuário

> "We face a fascinating future, with much exciting new technology, many new information appliances. We should not have to know how they work. We should not need to know anything about their technology. All we have to know is our job and what we are tying to accomplish. The appliances simply work: they provide the information we need when we need it, effortlessly, without any effort on our part. Smart things, cyborgs, and emotional things: the future will indeed be different."
> — DONALD A. NORMAN, *Appliances of the future*, 2002.

Ao longo da história da Engenharia de Software, pode -se perceber com clareza uma preocupação crescente no que se refere ao fator q ualidade, tanto no âmbito acadêmico quanto no industrial. Acreditava-se que o termo 'qualidade de software' dizia respeito apenas àquelas características que poderiam ser encontradas no produto final, ou seja, naquilo que era efetivamente entregue ao cliente. Entretanto, devido à demanda por sistemas mais complexos e ao conseqüente acirramento da concorrência nesse segmento de mercado, as empresas notaram que aquela definição de qualidade já não era mais suficiente para garantir seu sucesso. Tornou -se, assim, imprescindível buscar maturidade cada vez maior não só para o produto gerado, mas também para os processos, modelos e normas que conduziriam o desenvolvimento de software.

Sabe-se hoje que os esforços de desenvolvimento de sistemas computacionais devem estar concentrados em suas fases iniciais de concepção e projeto, de forma a reduzir ao máximo a probabilidade de que eventuais problemas apareçam nas etapas seguintes, principalmente aqueles causados em decorrência de mudanças não previstas nos requisitos iniciais. Hall (2001) afirma haver evidências de que um projeto de baixa qualidade pode trazer outros custos adicionais a uma empresa, seja por meio do atendimento aos usuários devido a reclamações ou solicitação de serviços, seja através da devolução de bens. Segundo o autor, um design de baixa qualidade causaria também redução nas vendas de tal empresa devido à baixa aceitação do consumidor, uma imagem deficiente do produto e os efeitos associados da percepção do cliente sobre a companhia em si.

Embora atualmente possa ser constatada, por parte das empresas, uma ênfase maior nos processos de desenvolvimento, Kujala, Kauppinen e Rekola (2001) observam que muitos sistemas simplesmente não atendem às expectativas básicas de seus usuários, sendo rapidamente abandonados por eles. Muito se tem divagado sobre as possíveis razões para essa situação, entre elas: técnicas de desenvolvimento que não suportam escalabilidade, interdependências não controladas dentro dos programas e sistemas operacionais defeituosos. En tretanto, vários autores como Constantine e Lockwood (2001) têm demonstrado que o principal problema não é meramente tecnológico, mas sim conceitual, *i.e.* está justamente naquilo que aquelas metodologias consideram ser o objeto central de seu estudo.

Para eles, a Engenharia de Software tradicional tem construído sistemas cujo foco primário são suas *funcionalidades* e mesmo suas estruturas internas, ou seja, processamento lógico correto, desempenho adequado, entre outras. Para demonstração, pode -se citar o RUP (*Rational Unified Process*), certamente o processo de desenvolvimento de software mais difundido hoje nos meios científico e comercial, que — nas palavras do próprio Kruchten (2000), um de seus mais conhecidos defensores — é um processo "*centrado na arquitetura*" (capítulo 5), ou ainda "*dirigido a casos-de-uso*" (capítulo 6). Conforme conhecimento geral da comunidade de software, casos -de-uso constituem-se em uma visão de projeto centrada *no sistema*. Kruchten ainda os define como sendo nada mais que uma seqü ência de ações (fluxo de eventos através do sistema), que geram um resultado de valor observável a um ator particular. Vale recordar aqui que, em UML (*Unified Modeling Language*, a linguagem de modelagem e especificação utilizada no RUP), o que caracteriza como realmente importante o conceito de 'ator' é seu papel *no sistema*, não os aspectos do usuário que ele representa. Torna -se, assim, claro que tais processos são centrados na funcionalidade dos sistemas que se propõem a

desenvolver e tudo mais gira em torno dos casos-de-uso. Pouca ou mesmo nenhuma atenção é dada aos usuários finais do sistema, o que se comprova na confusão cotidiana que se faz entre os termos 'usuário' (quem, de fato, utilizará o software) e 'cliente' (*stakeholder* contratante).

Quase sempre, e principalmente por questão de necessidade, os usuários são obrigados a moldar-se aos sistemas computacionais que utilizam, o que pode ser melhor observado em países onde não há uma legislação estabelecida que ampare o consumidor de software. Saffo (1996) lembra que somos uma espécie 'assustadoramente' adaptável e, isto considerando, questiona a real qualidade daqueles sistemas cujo desenvolvimento não leva na devida conta seus usuários finais. Para ele, boas ferramentas deveriam *ajustar-se ao usuário*, mas reconhece que elas ainda são muito escassas hoje em dia. Não obstante, o autor constata que os usuários tendem a abandonar tais ferramentas a partir do momento que o esforço requerido para utilizá-las excede seu 'limite de indignação' — o máximo de esforço que se está disposto a empreender no sentido de completar uma tarefa com o uso de determinado sistema.

Sobre a usabilidade de sistemas, consideremos ainda um fato bastante curioso que tem sido gerado pelo crescente apelo à convergência digital: o desenvolvimento de aparelhos cada vez menores, concentrando um número de funções até pouco tempo inimaginável. São os chamados 'produtos inteligentes', cujo exemplo mais popular hoje certamente são os telefones celulares, que já contam com acesso à internet, câmeras de vídeo embutidas, jogos em três dimensões, download e reprodução de músicas no formato MP3 etc. Hall (2001) sumariza que a redução do tamanho impõe restrições físicas ao design de interfaces dos tais dispositivos — botões pequenos alternando vários comandos, *feedback* visual limitado etc. Não obstante, Adams e Hall (1992) argumentem que mesmo as tecnologias com interfaces físicas suficientemente grandes — como em alguns transportes públicos e máquinas de venda automática — também não estão imunes a problemas de usabilidade. A enorme quantidade de funções (muitas delas totalmente desnecessárias conforme as necessidades reais do usuário comum) que tentam agrupar-se em um único dispositivo com limitações físicas torna-se um entrave à sua facilidade de utilização. Norman (1998) sugere que a complexidade de uso aumenta segundo o quadrado da funcionalidade, fenômeno que ele denomina por *creeping featurism*. Norman (1990) ainda esclarece que os sistemas atuais são guiados pela tecnologia, não pelo que as pessoas realmente precisam ou desejam. Isso pode ser constatado com uma simples análise empírica, por exemplo, sobre qual a porcentagem de idosos na população de consumidores que, mesmo adquirindo um daqueles celulares supra-citados, utiliza todas as suas funções regularmente.

Sendo assim, todo o desenvolvimento de produtos tecnológicos deveria ter seu foco em quem, afinal, utilizará essas tecnologias. Norman e Draper (1986), Norman (1988, 1998) e Shneiderman (1998) mostram que, para se evitar problemas de usa bilidade, designers deveriam empregar uma abordagem ergonômica — ou centrada no usuário — em seu trabalho de projeto. Em termos gerais, define-se o design centrado no usuário (do inglês *user-centered design*) como uma filosofia ou metodologia de projeto por meio da qual as necessidades, desejos e limitações dos usuários finais de um sistema são extensamente considerados em cada estágio do processo de design. Uma abordagem centrada no usuário, conforme esclarece Hall (2001), envolve conhecer quem são os usuários do sistema, quais são suas qualificações, necessidades e expectativas, quais são seus objetivos e as tarefas necessárias para os alcançar, além de conhecer os ambientes físicos e sociais em que tais usuários executam seu trabalho.

Eason (1992) descreve três formas de design centrado no usuário, dependendo do tipo de design em desenvolvimento:

1. **Genérico**, isto é, design feito *para* o usuário, onde o papel do mesmo é como sujeito nos testes de design (*e.g.* produtos de consumo ou software em geral);
2. **Específico**, ou seja, design feito *pelo* usuário, onde o papel do mesmo é como participante no processo de design (*e.g.* uma aplicação de software para atender necessidades específicas de uma companhia);
3. **Adaptável**, em outras palavras, design *adaptado pelo* usuário, onde o papel do mesmo é controlar como a interface se apresenta e/ou opera para ele (*e.g.* individualizar a interface para sistemas operacionais em computadores pessoais).

O mesmo autor também descreve uma forma genérica de design centrado no usuário:

- Primeiramente, auxilia-se os usuários a articular suas necessidades como parte dos *requisitos* gerais do produto, que são função entre o contexto em que o produto será utilizado e as tarefas particulares a serem executadas;
- Segundo, formulam-se várias *opções de design*, que devem considerar tanto restrições técnicas como sociais;
- Em terceiro lugar, avalia-se o grau em que as *opções* atendem aos *requisitos*.

Eason (1992) esclarece que especialistas externos podem ser necessários para ajudar a identificar opções de design, mas que os requisitos *pertencem* exclusivamente aos usuários. Hall (1997) estende o modelo de Eason, sugerindo que — utilizando-se primeiramente *mockups* de baixa fidelidade das opções iniciais de design — os resultados das avaliações com usuários poderiam realimentar uma re-inspeção das necessidades destes. Assim, baseando-se em qualquer revisão dos requisitos, fidelidade extra seria ser agregada às (novas) opções de design. Essas, então, poderiam ser avaliadas por usuários para fornecer *feedback* adicional para uma segunda revisão dos requisitos e, se necessário, uma terceira iteração do processo de design.

Pode-se argumenta, no entanto, que isso já ocorre nos processos clássicos da Engenharia de Software: o que acrescentaria, então, a observação de um a abordagem iterativa centrada no usuário? Hall (1997) responde a tal questionamento lembrando que, na prática, os *requisitos* do usuário raramente são alterados, pois já foram acordados entre o *stakeholder* cliente e a empresa de software e, portanto, não são dignos de re-exame, mesmo sob a luz dos resultados dos testes com usuários. Pelo contrário, testes com usuários na Engenharia de Software tradicional envolvem determinar quais entre as várias alternativas de design melhor atendem aos requisitos já acordados. No próximo capítulo, trataremos especificamente sobre o ciclo de prototipação com níveis de fidelidade.

De acordo com Nielsen (1993), o design centrado no usuário está organizado em etapas de análise de requisitos dos usuários, especificação, prototipação e avaliação:

- **Análise** — O objetivo desta fase é identificar os usuários finais da interface, especificando e entendendo o processo pelo qual realizam sua tarefa e buscando extrair as suas necessidades, que devem orientar todas as etapas seguintes. A fase de análise, conforme afirma Nielsen (1993), tem grande importância no início do desenvolvimento, pois se caracteriza como a principal fonte de informação para a concepção do sistema.
- **Especificação** — Após a fase de análise, que envolve o entendimento do processo e da atividade realizada pelos usuários, acontece a especificação do sistema. A especificação deve permitir que o usuário atinja suas metas, realizando tarefas de maneira interativa e precisa. Uma especificação descreve um modelo de interação onde o usuário utiliza funções providas pelo software, visualiza informações ou fornece dados para o próprio sistema.

- **Prototipação** — Na fase de prototipação, é realizada a construção de uma interface primária, na qual procura-se obter uma representação dos conceitos desenvolvidos nas fases anteriores. A avaliação do protótipo é realizada com base em contextos de utilização e definida utilizando-se critérios determinados pela equipe de projeto.
- **Avaliação** — A avaliação da interface é um importante passo do processo de design, pois é através dela que se consegue estimar o sucesso ou fracasso das hipóteses do designer sobre a solução que ele está propondo, tanto em termos de funcionalidade, quanto de interação. Hartson (1998) argumenta que, ainda que o designer se baseie em uma abordagem teórica e conte com a ajuda de diretrizes (guidelines) e princípios de design, é necessário que ele avalie o resultado final obtido.

Muito embora a sistematização acima possa, à primeira vista, causar algum tipo de receio por parte de empresas que já possuem suas próprias metodologias de desenvolvimento de software bem definidas, o design centrado no usuário apresenta-se na realidade como um conjunto de técnicas baseadas em fatores humanos que podem e devem ser integradas àquelas metodologias. Para Gould e Lewis (1985), Shackel (1986) e Gould (1995), uma abordagem centrada no usuário envolve basicamente processos de design participativo, testes com usuários e design iterativo. A seguir, apresentaremos uma sucinta revisão da literatura a respeito de cada uma dessas dimensões do design centrado no usuário.

2.1 Design participativo

Muitas abordagens de design, principalmente aquelas utilizadas em estudos e projetos que envolvem ergonomia, pressupõem que o projetista detém conhecimento suficiente para agir em nome do usuário, aplicando conceitos e princípios gerais desenvolvidos e validados por meio de pesquisas. Tais abordagens caracterizam-se como sendo *para* o usuário, estando sob o encargo do designer decidir o que, afinal, seria melhor para aquele. Entretanto, justamente como crítica ao que foi descrito anteriormente, o design participativo surgiu como uma abordagem de projeto que busca envolver ativamente usuários finais como colaboradores no processo de design. Rosson e Carroll (2002) afirmam que, "*no design cooperativo (algumas vezes também chamado*

de design participativo), um sistema interativo emerge através de uma colaboração direta entre designers e usuários em potencial".

Kuhn e Winograd (1996) testemunham que as maiores comp anhias de desenvolvimento de software já consideravam comum o conhecimento de que é importante projetar *com* o usuário, ao invés de projetar *para* ele. Bødker (1991) demonstra o quanto é essencial aos usuários experimentar manualmente *mockups* e protótipos, participando ativamente do design de aplicações. Consideram-se atentamente questões inerentes ao ser humano, como desejos, ambições, crenças, emoções etc. O objetivo dessa prática é garantir que o produto, da forma como foi projetado, atenda às necessidades de seus usuários, tornando-se, assim, usável. Design participativo implica em envolver usuários como participantes no time (equipe) de design. Hackos e Redish (1998) defendem que "*incluir usuários no time de projeto é geralmente uma grande idéia que, definitivamente, deveria tornar-se parte de qualquer processo sofisticado centrado no usuário*". Na Figura 2.1 abaixo, pode-se ver um dos laboratórios de usabilidade da Oracle, especialmente preparado para sessões de design participativo:

Figura 2.1 — Laboratório de usabilidade da Oracle, especialmente preparado para sessões de design participativo (*ui.us.oracle.com*).

O envolvimento entre usuários, clientes e desenvolvedores para projetar uma dada solução se dá, normalmente, através de workshops. Esses encon tros são importantes por uma série de razões, dentre as quais passamos a citar:

- Usuários ganham voz no processo de design, o que aumenta a probabilidade de se gerar um design com grande usabilidade;
- Permite-se que colaboradores técnicos e não-técnicos possam participar igualmente;
- Viabiliza-se um fórum para tratar sobre tópicos de interesse geral;
- Utilizam-se técnicas que podem ser facilmente aprendidas e aplicadas em atividades futuras dentro da empresa.

Como parte essencial no processo design centrado n o usuário, o design participativo deve ser aplicado desde as fases de concepção e projeto de software, notadamente nas etapas inicias de elicitação de requisitos. Kuhn e Winograd (1996) apontam que as pesquisas sobre design participativo têm desenvolvido u ma variedade de técnicas no sentido de facilitar a comunicação das possibilidades de novas tecnologias aos usuários finais. Robertson (2001) explica que o sucesso em tal comunicação representa o fator-chave para uma elicitação consistente com usuários e, sob o ponto de vista destes, a autora classifica os requisitos em:

- **Conscientes** — É o tipo de requisito com maior probabilidade de ser mencionado pelo usuário. É algo para o que ele está particular e conscientemente atento (*e.g.* "*eu quero que a nova câmera caiba no meu bolso, tire fotografias digitais e possua uma bateria com autonomia maior*").
- **Inconscientes** — É um requisito normalmente não comunicado pelo usuário, pois este não sabe que precisa dele. Uma razão para tal situação poderia ser o usuário estar tão acostumado em ter a funcionalidade satisfeita que ele não mais imagina que ela seja um requisito (*e.g.* o usuário pode não mencionar à equipe de projeto seu desejo de que a nova câmera rebobine o filme quando a última fotografia for tirada). Outros motivos seriam: o usuário pressupõe que todos os demais na equipe possuem o mesmo conhecimento sobre o domínio que ele; por receio de subestimar o designer, muitas vezes é difícil ao usuário explicar absolutamente todos os detalhes sobre algo que ele conhece bem etc.
- **Insonháveis** — São os requisitos que o usuário não comunica por acreditar que eles são impossíveis de serem desenvolvidos ou por não fazerem idéia do que seria ter o produto em mãos e experimentar essa nova tecnologia (*e.g.* o usuário pode pensar

"*não vou sequer mencionar que eu gostaria que a nova câmera fosse à prova de água, pois sei que isso não é possível de ser feito*").

Robertson (2001) argumenta que, embora a entrevista seja sem dúvidas a técnica mais utilizada em sessões de design participativo e útil em muitas situações, ela não consegue trazer à tona todos os tipos de requisitos do usuário. A autora ainda explica a melhor utilização, em cada tipo de cenário, das técnicas consagradas na literatura para elicitação de requisitos, dentre elas:

- Abstração;
- Aprendizagem;
- Brainstorming;
- Mapas mentais;
- Modelos de simulação;
- Arqueologia de sistemas;
- Gravações com vídeo;
- Pontos de vista.

Nielsen (1993) lista o design participativo como o quarto estágio em seu modelo de ciclo-de-vida da Engenharia de Usabilidade e enfatiza a necessidade crucial que os projetistas têm de ter acesso a uma quantidade adequada de usuários representativos, pessoas que realmente estarão utilizando o sistema. O autor cita que mesmo gerentes bem-intencionados não substituem os usuários no design participativo, pois eles geralmente desconhecem os tópicos exatos com que os usuários se deparam em seu trabalho cotidiano, além de terem características diferentes — em muitos aspectos — daquelas presentes nos usuários reais. Freqüentemente, usuários que colaboram em workshops de design participativo levantam questões às quais o time de desenvolvimento permanecia completamente alheio, o que pode ser especialmente comprovado no que diz respeito a mal-entendidos em potencial entre as reais tarefas realizadas pelos usuários e o modelo que os desenvolvedores têm das mesmas — acrescenta Nielsen (1993). Usuários participantes dentro de um processo de design são por vezes denominados de 'especialistas no domínio' (*subject matter experts*, ou SMEs).

A fim de se obter o máximo de benefícios com o envolvimento do usuário nesse processo, devem ser apresentadas a ele sugestões de design do sistema de uma forma que consiga

entender. Não é responsabilidade dos usuários inventar opções de design, e sim rea gir a elas conforme as desaprove ou percebam que não funcionarão na prática — tarefa na qual, saliente-se, são bons como ninguém. Pondo de lado os numerosos volumes com especificações de sistemas, devem-se empregar representações concretas e visíveis de de sign para essa proposta, preferencialmente na forma de protótipos de baixa fidelidade (que serão discutidos em maiores detalhes no capítulo 3). Não obstante, Nielsen (1993) ainda nos alerta — tratando especificamente de projetos de desenvolvimento maiores e mais demorados — sobre o perigo de confiar demais nas informações obtidas a partir de um pequeno grupo de usuários que nunca muda. O autor recomenda que, em tais projetos, deva haver periodicamente uma mudança dos usuários que colaboram no processo de de sign participativo, pois, à medida que aumenta seu envolvimento no desenvolvimento do sistema, eles se tornam menos representativos com relação à população média de usuários. Novos usuários participando no processo de design terão maior probabilidade de qu estionar potenciais problemas, visto que não conhecerão a história daquele projeto; entretanto, certamente há um *trade-off* a ser considerado entre o número de mudanças e o tempo requerido para explicar o projeto a novas pessoas colaborando com o time. Acre scenta-se que a participação de usuários no processo de design pode ser medida segundo quatro dimensões:

- Liberdade de interação com os designers;
- Nível de envolvimento no processo;
- Escopo da participação no sistema total sendo projetado;
- Grau de controle sobre decisões de projeto.

Mas alguém poderia, como faz Furnival (1995) de forma retórica, questionar: "*por que uma empresa escolheria uma metodologia com a participação dos usuários, quando já foi comprovado que essas metodologias de design, como concentram os esforços nas fases do começo do ciclo-de-vida* [isto é, a determinação dos requisitos] *são mais caras e prolongam o processo de design?*" (p. 8). Segundo Mantei e Teorey (1988), os custos extras decorridos da participação de usuários representativos n o processo de design são recuperados posteriormente na aceitação e conseqüente uso efetivo do sistema pelos usuários, e isto particularmente ocorre quando o sistema será utilizado por cem ou mais usuários.

Furnival (1995) ressalta que, quanto ao desenvolvi mento de novos sistemas, a ignorância da possibilidade de usuários participarem e contribuírem com este processo muito provavelmente

resultará em usuários insatisfeitos e, por conseqüência, em baixa produtividade e qualidade de trabalho. A autora defende ainda que, recordando-nos do fato de que todo sistema é uma entidade sociotécnica — não apenas com seu lado técnico, mas também com seu lado social (humano), o design participativo apresenta-se como o caminho mais natural e holístico no desenvolvimento de novas tecnologias. Cooley (1987) justifica a prática do design participativo com a seguinte colocação: "*as decisões que tomamos com respeito ao desenvolvimento tecnológico durante os próximos cinco ou dez anos terão profundos efeitos no modo como nossa sociedade se desenvolve, como os seres humanos se relacionam com as máquinas e uns com os outros, na relação entre seres humanos, no ambiente que eles construírem e na própria natureza*" (p. 38).

2.2 Testes com usuários

Muitos métodos para avaliação de usabilidade em interfaces em que a participação do usuário não é necessará têm sido desenvolvidos e aplicados no mercado de software, como é o dos checklists de usabilidade baseados em guidelines e da avaliação heurística por especialistas em usabilidade. Segundo Nielsen (1993), os problemas de usabilidade detectados por esses métodos seriam melhor estimados sob, pelo menos, um pequeno número de observações de usuários reais. Para a filosofia de design centrado no usuário, os testes com usuários são simplesmente insubstituíveis, revelando-se como uma das práticas mais recomendadas e estudas pela Engenharia de Usabilidade, pois eles fornecem informação direta sobre como as pessoas utilizam o computador e quais os problemas exatos com que elas se deparam nas interfaces sob teste.

De forma bastante simples, a Engenharia de Usabilidade define testes envolvendo usuários como uma família de métodos que avaliam interfaces gráficas através da coleta de dados realizada a partir da observação de pessoas no uso real dos sistemas. Rubin (1994) afirma que o teste de usabilidade é um processo no qual participantes representativos avaliam o grau em que um produto se encontra em relação a critérios específicos de usabilidade. Testes com usuários envolvem os mesmos como sujeitos na avaliação de conceitos de design e *mockups* desde o início do processo de design.

Nielsen (1993) adverte que testes com usuários devem prezar por dois pontos principais: confiabilidade e validade. O primeiro deles está relacinado à questão de que, caso o teste seja repetido, os resultados obtidos serão os mesmos — o que se torna um problema crucial, visto

que encontramos enormes diferenças individuais entre usuários. O autor demonstra, contudo, que *algum* dado sobre o comportamento dos usuários é sempre melh or do que *nenhum*, enquanto Landauer (1988a, 1988b) comprova que intervalos de confiança — que podem ser estimados através de testes estatísticos padrão — são importantes se a escolha entre duas opções de design não está baseada apenas em qual é a melhor, m as também *o quão* melhor ela é.

O segundo ponto a ser considerado em um teste com usuários é sua validade, ou seja, provar metodologicamente que ele mede algo relevante para a usabilidade de produtos reais em uso real fora do laboratório. Conforme aponta Ni elsen (1993), problemas típicos de validade envolvem testes com usuários não-representativos, tarefas erradas, não incluir restrições de tempo e influência social.

O autor classifica as avaliações de interfaces em dois grupos: formativas ou somativas, tendo em vista que a proposta do teste deve estar clara. As formativas são aquelas feitas durante o processo de design, permitindo que se identifique e seja solucionado um problema de interação antes que o desenvolvimento de uma certa aplicação esteja concluíd o, ou até mesmo antes de ser implementada. As avaliações somativas, por sua vez, analisam o produto de software já terminado. Nielsen (1993) lista uma série de tópicos que devem ser considerados no planejamento de um teste com usuário (*e.g.* quais os objetivos do teste, quanto tempo espera-se que dure cada sessão, que aparato tecnológico será necessário, quais tarefas os usuários serão solicitados a executar etc.), apresenta variáveis que influenciam o orçamento dos testes e discorre sobre a importância de s e avaliar testes-piloto antecipadamente. Na Figura 2.2 abaixo, podemos ver um exemplo de laboratório para testes com usuários (Nielsen Norman Group):

Figura 2.2 — Laboratório de usabilidade do Nielsen Norman Group, especialmente preparado para testes com usuários (*www.useit.com*).

Uma etapa de suma importância em um teste de avaliação de usabilidade é identificar e recrutar usuários representativos do sistema. Ferreira (2002) afirma que determinar o perfil do usuário típico — alguém que represente uma média dos usuários finais — auxilia a equipe de desenvolvimento a ter a representação de um usuário para o qual o sistema está sendo projetado. Segundo a autora, as informações necessárias para se determinar o perfil do participante dos testes podem ser encontradas na especificação funcional, em análises e estudos realizados no mercado, junto ao gerente do produto, análise de grupo e padrões competitivos. Algumas vezes, os indivíduos exatos que utilizarão o sistema podem ser identificados, por exemplo, no caso de softwares que são desenvolvidos internamente em uma companhia para uso em um dado departamento daquela empresa. A figura 2.3 abaixo retrata um teste de usabilidade com estudantes no IDEA Lab, da Universidade de Melbourne:

Figura 2.3 — Teste de usabilidade no IDEA Lab com estudantes da Universidade de Melbourne (*idealab.dis.unimelb.edu.au*).

Quanto à experiência no uso da tecnologia, Nielsen (1993) afirma que a grande maioria das interfaces gráficas precisam ser testadas com usuário novatos, ao passo que testes de muitos outros sistemas devem também recrutar usuários mais experientes. A determinação do número de participantes é influenciada pelo grau de confiança que se deseja atingir, pela quantidade de recursos disponíveis para estabelecer e conduzir o teste, por uma análise do tipo de participantes

e pelo tempo estimado de duração da sessão de teste. Nielsen e Landauer (1993) argumentam que, a partir dos testes com o quinto usuário, os observadores começarão a constatar repetidamente os mesmos problemas de usabilidade e não aprenderão muitas coisas novas. Entretanto, se o objetivo do teste é validar resultados estatísticos, Ferreira (2002) afirma que será necessária uma quantidade maior de participantes, a fim de se obterem propriedades e generalizações sobre uma população específica. Testes com usuários possuem geralmente quatro estágios típicos:

1. **Prepação** — Deve-se estar assegurado que a sala (ou laboratório) de testes esteja preparada para o experimento, o computador esteja no estado inicial que foi especificado no plano de testes e que todo o restante do material de testes, instruções e questionários estejam disponíveis. Para se evitar disconforto e confusão nos usuários, esta preparação deve ser completada antes da chegada dos mesmos.
2. **Introdução** — Neste estágio, o condutor do teste saúda o usuário e lhe fornece uma breve explicação sobre o propósito do teste, bem como instruções sobre como utilizar o equipamento, caso não seja familiar a ele. Nielsen (1993) aponta alguns pontos essenciais que devem ser explicados ao usuário nesta etapa, dentre eles:
   - O propósito do teste é avaliar o software, não o usuário.
   - Não houve envolvimento direto do condutor do teste na construção do sistema a ser avaliado, permitindo que o usuário possa falar livremente e se m se sentir constrangido. Caso o condutor tenha projetado o sistema, recomenda -se que este fato não seja mencionado no sentido de se evitar o efeito oposto.
   - Os resultados do teste serão utilizados para melhorar o software, tal que o sistema a ser entregue eventualmente será diferente do que está sob teste.
   - O teste é voluntário e o usuário poderá desistir dele a qualquer momento.
   - Os resultados dos testes serão mantidos em sigilo e não serão de uma maneira que se possa identificar testes individuais por quem quer que seja.
   - Explicações mais detalhadas deverão ser dadas aos usuários sobre gravações de vídeo ou áudio, caso haja utilização de tais recursos.
   - Os usuários estão livres para qualquer questionamentos, dado que se deseja descobrir aquilo que não está claro na interface, mas que o condutor nem sempre poderá responder suas perguntas durante o teste em si, pois o objetivo

do mesmo é avaliar inclusive se o sistema poderá ser utilizado sem a necessidade de ajuda externa a ele.

O autor continua citando algumas recomendações, tais como a de que formulários de consentimento só deverão ser apresentados aos usuários quando forem realmente necessários, pois isto pode elevar seu nível de ansiedade. Após a introdução, devem-se entregar ao usuário as instruções escritas para o teste (caso haja alguma), incluindo a primeira tarefa, solicitá-los a que as leiam, perguntando-lhes se tudo está claro.

3. **O teste em si** — Durante o teste, o condutor deve policiar-se a fim de interagir o mínimo possível com o usuário, tomando cuida do para não expressar qualquer tipo de opinião pessoal ou indicar se o usuário está procedendo conforme o esperado. O condutor deve evitar ajudar o usuário, ainda que este se encontre dificuldades graves. A única exceção a tal regra ocorre quando o usuário encontra-se completamente desorientado, demonstrando insatisfação com essa situação. Caso haja vários observadores no teste, é recomendável que se eleja um deles para assumir o papel de condutor, tal que seja a única pessoa a fornecer instruções e falar durante o experimento. Os demais devem permanecer em silêncio mas, caso seja absolutamente necessário fazer algum comentário, este deve ser passado ao condutor na forma de uma nota ou comunicado durante os intervalos.
4. **Relato** — Após o teste, solicita-se aos usuários que preencham questionários subjetivos para avaliar sua satisfação no uso do sistema. Para se evitar algum tipo de comentário induzido pelas perguntas dos observadores, aqueles questionários devem ser respondidos antes de qualquer outra discussão. Embora os usuários possam dar recomendações completamente contraditórias para melhorar a qualidade do sistema, ainda assim essas sugestões podem servir como uma rica fonte de idéias adicionais a serem consideradas durante a etapa seguinte de redesign. No relato, os usuários também podem ser solicitados a comentar eventos ocorridos durante o teste, apontando aquilo que se revelou de difícil compreensão, bem como clarificando alguns de seus pressupostos e objetivos. Nielsen (1993) recomenda que, logo após a saída do usuário do teste, os observadores devem conferir todos os resultados obtidos, condensando-os num breve relatório tão logo seja possível. O autor comenta outros métodos que podem ser utilizados para se obter *feedback* dos testes com usuários além de questionários, como é o caso do protocolo *thinking-aloud* — o usuário verbaliza seus pensamentos enquanto utiliza o sistema, permitindo aos

observadores entender como ele visualiza o sistema e identificar quais são suas concepções erradas sobre o sistema.

Designers normalmente focam seus esforços em criar sistemas que se mostrem agradáveis e com uma rica experiência visual, enquanto desenvolvedores preocupam-se apenas em garantir uma funcionalidade correta. Esse modelo tende a comprometer seriamente a usabilidade dos sistemas, pois a equipe muitas vezes se sente pressionada pelas expectativas dos gerentes e se esquece das necessidades dos reais usuários. Se os testes revelarem problemas — tais como dificuldade de compreensão das instruções, manipulação confusa de partes do sistema ou interpretação errada de *feedback* (*e.g.* mensagens de erro no software) — então o time de projeto deverá melhorar o design e testá-lo novamente num ciclo de design iterativo, tema do qual passaremos a tratar em seguida.

2.3 Design iterativo

A partir dos problemas de usabilidade encontrados nos testes com usuários, bem como nas oportunidades de melhoria do sistema visualizadas pela equipe de desenvolvimento, deve-se produzir uma nova versão da interface. Define-se design iterativo como sendo uma metodologia de projeto baseada em um processo cíclico de design, prototipação, testes e redesign, de acordo com os passos que seguem:

1. **Design** — geração e análise de idéias, bem como criação de soluções;
2. **Prototipação** — materialização do design através de uma instanciação do mesmo;
3. **Testes** — avaliação o protótipo, descobrindo-se problemas de usabilidade;
4. **Redesign** — refinamento do projeto, voltando-se ao passo 1.

Kuniavsky (2003) explica que o design iterativo satisfaz a necessidade de um modo sistemático de integrar o processo de se encontrar problemas e criar soluções, focando-se em elementos individuais sem perder a visão do todo. O autor afirma que o desenvolvimento iterativo baseia-se na idéia de refinamento contínuo, através de tentativa e falha. Bailey (1993) adverte que revisões do projeto podem introduzir novos problemas de usabilidade. Essa é uma das razões por que Nielsen (1993) recomenda que o design iterativo seja sempre combinado com

avaliações. O autor afirma que, em alguns casos, resolver um problema pode tornar a interface pior para aqueles usuários que não experimentaram o tal problema. Um *trade-off* seria necessário, portanto, para se determinar pela permanência ou mudança da interface. Isso se basearia numa análise de freqüência de quantos usuários deparar-se-iam com o problema, comparando-se com o número de usuários que seriam prejudicados devido à solução proposta. O autor cita que o tempo e os custos que o conserto de um problema particular demandariam é, obviamente, um fator crítico na determinação de prioridades.

Sobre o design iterativo, é comum ouvirmos designers questionarem-se sobre quando se deve encerrar o ciclo de iterações. Hackos e Redish (1998) defendem que, se a meta em questão é assegurar usabilidade para o produto de forma que os usuários o considerem agradável e produtivo no uso, então certamente serão necessárias algumas iterações. Eles salientam que definir um fim para o processo de design iterativo requer que seja definido um conjunto de objetivos a ser alcançados antes que se encerre o design. Os autores afirmam que os limites de design para o esforço de desenvolvimento são determinados comumente pelo orçamento e cronograma disponíveis, ou por decisões de mercado.

O design iterativo é uma grande fonte de inspiração para novos produtos, com potencial para redirecionar as capacitações tecnológicas de uma dada empresa na direção projetos inovadores. A pesquisa com usuários reais permite compreender, comprovar e predizer o comportamento humano com uma certa tecnologia inovadora. O design iterativo também provê as ferramentas necessárias para medir quão bem as pessoas percebem, entendem, recordam e aprendem a utilizar novos produtos desde seu processo de design. Säde (1999) afirma que "*modelagem e avaliação iterativas são uma boa ferramenta para assegurar usabilidade e desejabilidade*", enquanto Gould e Lewis (1985) vão além, argumentando: "*no estado atual de entendimento sobre design de interfaces, ele* [o design iterativo] *é a única maneira para assegurar sistemas excelentes*". Nesse contexto, prototipação é a atividade-chave para companhias que buscam garantir a usabilidade de suas aplicações, mesmo daquelas inovadoras, provendo o maior *feedback* de seus usuários pelo menor custo. É sobre esse tema que trataremos no próximo capítulo.

CAPÍTULO 3

Prototipação rápida

> "Of course, structured methods of understanding the activities of daily life have become almost as common as the key step in creating great user experiences. [...] Executives are using design in the early stages of their processes and in solving types of problems that traditionally have not involved designers. [...] Executives are doing this because designers are creating a new body of knowledge, a set of frameworks and methods that can be used by companies for understanding the relationship between the business context and the user experience; finding patterns in seemingly chaotic situations; designing systems of solutions that simultaneously create user value and economic value; and visualizing and prototyping opportunities early in the development process. Furthermore, these frameworks and methods are not only being used to create new communications, products and services; executives are using them to reform the ways their organizations work."
>
> — PATRICK WITHNEY, *Perspectives on design strategy*, 2005.

Em uma abordagem iterativa de projeto, principalmente naquelas que são centradas nos usuários finais de um sistema, prototipação revela-se como um elemento essencial. Ela permite a criação, avaliação e refinamento de opções de design, até que se alcance a usabilidade desejada. Conforme

Buchenau e Suri (2000) explicam, protótipos são representações de um sistema, construídas antes que os artefatos finais venham a existir de fato. A prática de prototipação está relacionada, assim, à elaboração de representações incompletas da um sistema-alvo com o propósito de se executarem testes, como uma maneira eficaz de se compreender as dificuldades de desenvolvimento, bem como a escala de problemas correspondentes ao mesmo. Rosson e Carroll (2002) acrescentam que um ou mais protótipos podem também ser desenvolvidos para se comparar soluções alternativas para tópicos de usabilidade que não podem ser resolvidos simplesmente por discussão ou análise.

Como se pode constatar na literatura, o uso de prototipação não é recente. Shackel (1959), por exemplo, investigou o problema de design de se "*posicionar vinte e quatro potenciômetros e vinte e quatro switches associados num painel de tamanho mínimo, consistentemente com fácil localização e controle pelo operador*", utilizando para isso vários layouts (*i.e.* designs) desenhados em tamanho natural em papel. Seminara e Gerrie (1966) investigaram com sucesso *mockups* alternativos de painéis para visualização e controle, utilizando elementos magnetizados, além de haverem testado representações tridimensionais de estações de trabalho a serem incorporadas dentro de um laboratório lunar móvel. Verifica-se, portanto, que inicialmente o uso de protótipos estava concentrado nos aspectos físicos do design (*e.g.* layouts e dimensionamento de estações de trabalho).

Entretanto, à medida que a tecnologia avançava, os protótipos passaram a ser rápida e facilmente desenvolvidos em telas de computador, como constata Hall (2001). Segundo o autor, isso facilitou a habilidade de se conduzir a prototipação de sistemas de uma maneira mais eficiente, no que passou a ser denominado de 'prototipação rápida': a geração de *mockups* em um curto período de tempo. Ferramentas RAD (acrônimo para *rapid application development*) de prototipação permitem que se exibam elementos da interface de modo prático, bem como que se associe a eles algum tipo de comportamento, a fim de se obter uma idéia do que será a interação com o software e as capacidades do sistema completamente implementado. Sinha, Klemmer e Landay (2002) apresentam uma ferramenta de prototipação rápida para interfaces baseadas em fala, denominada de SUED. Outros métodos — tais como prototipação em papel (que será discutido em sessão posterior deste capítulo) — podem ser também utilizados para se obter *feedback* dos usuários a cada protótipo testado. Tais protótipos mostram-se bastante úteis para se determinar o alvo do desenvolvimento, antecipando-se muitos problemas e complexidades desde a etapa de projeto do sistema. Vale parafrasear aqui a definição de Ward (1994) sobre o que seria

um 'problema' de usabilidade — qualquer característica (ou omissão dela) no produto ou em suas instruções que seja identificada como contribuinte para:

1. Um **erro**, ou seja, qualquer ação que produza uma resposta do sistema diferente daquela que foi intencionada;
2. **Confusão**, que ocorre quando o usuário não consegue prosseguir na utilização do sistema sem assistência, mesmo após ter consultado as instruções;
3. **Irritação** ou **desagrado**, observados quando o usuário comenta que uma característica em particular revelou-se de difícil uso ou não de uma maneira que ele teria preferido — mesmo que isso não tenha realmente resultado em um erro.

Não obstante à utilidade dessas ferramentas de prototipação, Buchenau e Suri (2000) constatam que muitos designers de interação têm verificado a necessidade de avançar o limite inerente àquelas ferramentas, passando a explorar e comunicar — na prática — *como* será interagir com aquilo que eles projetam. Considerando a crescente complexidade de interação homem-computador provocada pela convergência de tecnologias, espaços e serviços, os mesmos autores expõem a emergência e o valor daquilo que eles chamam de 'prototipação de experiências'. Segundo eles, o termo denominaria qualquer tipo de representação *integrada* — em quaisquer meios — elaborada para entender, explorar ou comunicar como deverá ser (*i.e.* quais as sensações experimentadas) lidar com aquilo que se está projetando. Em outras palavras, prototipar uma experiência significaria "*enfatizar o aspecto experimental de qualquer representação de que necessitássemos para, com êxito, (re)vivenciar ou conduzir uma experiência com um produto, espaço ou sistema*" (p. 424). Eles argumentam que a maioria das ferramentas para prototipação apenas *maquiam* a experiência real dos usuários, considerando-os como uma audiência passiva. Sendo assim, os autores discutem alguns métodos e técnicas que suportariam participação ativa dos usuários e forneceriam uma experiência subjetiva realmente relevante. Para tanto, a prototipação de experiências apresenta-se composta de três diferentes tipos de atividades no contexto do processo de projeto e desenvolvimento:

- Entender a experiência existente do usuário e seu contexto;
- Explorar e avaliar idéias de projeto;
- Comunicar as idéias para uma dada audiência (equipe e *stakeholders*).

Buchenau e Suri (2000) detalham possíveis participantes dessa audiência: clientes, designers de interação, designers industriais, especialistas em fatores humanos, e quipe de desenvolvimento etc. Os autores ressaltam a necessidade nos projetos atuais de equipes multidisciplinares, na qual cada uma das várias especialidades traz uma compreensão única sobre os tópicos abordados pelo time, bem como utiliza uma abordagem i ndividual para resolver os problemas. Portanto, faz-se necessário que existam "*ferramentas e técnicas que criem uma experiência compartilhada, fornecendo uma base para um ponto de vista comum*" (p. 425).

Os autores concluem, através de suas análise e exempl os, que prototipar experiências tem contribuído em projetos reais de design de três modos principais:

1. Ajudando a desenvolver um entendimento sobre a essência de uma experiência já existente. Prototipar experiências simularia aspectos importantes das relaç ões inteiras — ou de parte delas — entre pessoas, lugares e objetos, à medida que elas se sucedem no tempo.
2. Contribuindo na exploração de idéias. Tal abordagem de prototipação forneceria inspiração, confirmação ou rejeição de idéias, baseando-se na qualidade da experiência que elas simulam. Isso produziria respostas e *feedback* aos questionamentos dos projetistas a respeito das soluções propostas em termos de "*como seria se...?*".
3. Facilitando a comunicação de tópicos e idéias. Prototipar experiências permite que usuários e diferentes tipos de profissionais participem ativamente de uma nova experiência proposta, fornecendo ainda uma estrutura comum para que se estabeleça um ponto de vista compartilhado por todos.

Outro aspecto muito importante, relacionado à p rática de prototipação, são os testes de viabilidade e aceitação de novas tecnologias por parte dos usuários. Prototipação — como prática de projeto — vem sendo atualmente promovida dentro da comunidade de negócios como um elemento-chave no que se refere a o fator inovação, conforme atestam Leonard e Rayport (1997) e Scharge (1999). Mesmo as melhores e mais sofisticadas idéias inovadoras precisam de constatações empíricas que lhes assegurem sucesso no mercado, através de um processo iterativo de avaliações junto aos usuários; preferencialmente a um baixo custo. Na seção que segue, discutiremos os vários níveis de fidelidade de protótipos e suas melhores aplicações.

3.1 Níveis de fidelidade

Encontramos em Hall (2001) a definição de fidelidade em protótipos como sendo o parâmetro que os distingue de acordo com a precisão com a qual reproduzem as características encontradas no produto concluído. Para melhor ilustrar esse conceito, apresentaremos a idéia de protótipos horizontais e verticais proposta por Nielsen (1993) e encontrada em seu *ciclo de vida da Engenharia de Usabilidade*. Segundo o autor, o primeiro deles simularia todas as características do sistema, não se preocupando, contudo, em aprofundar-se nas funcionalidades. Já um protótipo vertical forneceria funcionalidade completa para algumas das características (ou módulos) do software. O sistema inteiro poderia, então, ser definido como tendo todas as características presentes, bem como suas respectivas funcionalidades completas. O grau de fidelidade, assim, representaria alguma combinação de atributos daqueles protótipos horizontais e verticais, definidos por aquele autor. A Figura 3.1 abaixo mostra protótipos de diferentes níveis de fidelidade para dois *websites*:

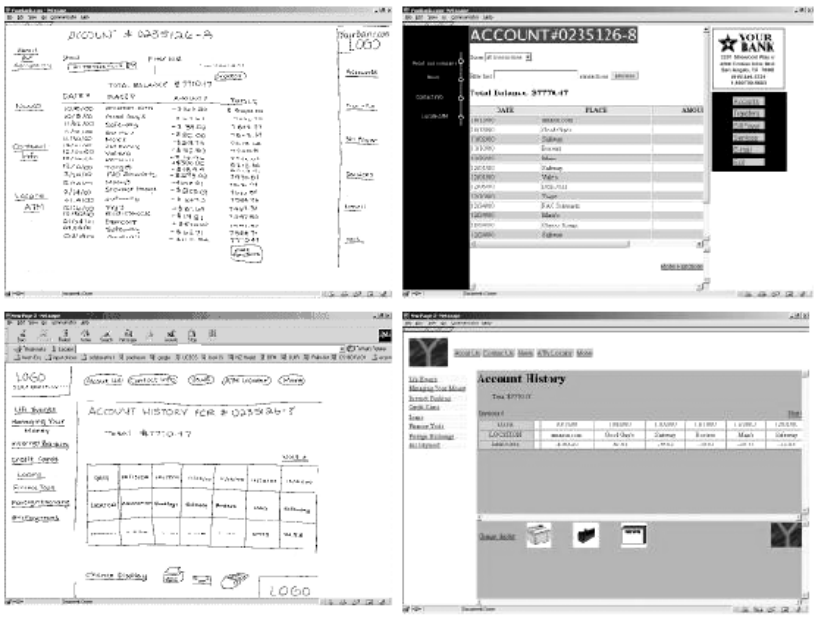

Figura 3.1 — Protótipos de dois *websites*: à esquerda, as versões de baixa fidelidade; à direita; as versões de fidelidade maior.

Prototipação pode assumir várias formas, sendo conduzida segundo muitos propósitos. Jordan (1998) afirma que "*há um número de diferentes opções de prototipação, com diferentes graus de realismo e sofisticação, que podem ser utilizadas no ciclo de design-avaliação*". O

autor nos fornece vários exemplos disso, em um nível crescente de fidelidade dos protótipos, conforme segue:

1. Descrições verbais ou escritas da forma e funcionalidade do produto proposto;
2. Protótipos visuais, como desenhos em papel;
3. Representações ou modelos físicos do produto;
4. Protótipos interativos baseados em telas;
5. Protótipos completamente funcionais.

Analisando a adequação de métodos ergonômicos para melhorar o projeto de produto, Stanton e Young (1999) estabelecem uma distinção entre os tipos de protótipos de acordo com os estágios principais do processo de design, como segue:

- Conceito;
- Design;
- Protótipo analítico;
- Protótipo estrutural;
- Protótipo operacional.

Segundo tais autores, um protótipo analítico corresponde apenas a um design auxiliado por computador, enquanto que um protótipo estrutural é aquele parcialmente implementado por alguma linguagem de programação. Já um protótipo operacional refere-se ao design final, pronto para ser completa e efetivamente implementado. Tal visão é similar àquela sobre o processo de design defendido por Meister (1990), onde se refere a testes com *mockups* sendo executados durante os estágios de *planejamento*, *projeto preliminar* e *projeto detalhado*, e testes operacionais de sistemas prototipados durante a fase de *produção e distribuição*.

Como expusemos anteriormente, testes de usabilidade representam o núcleo da Engenharia de Usabilidade e, segundo Rosson e Carroll (2002), sua boa prática está intimamente relacionada à prototipação: usuários representativos são solicitados a interagir com protótipos, sendo seu comportamento e reações subjetivas devidamente estudados. Os autores argumentam que uma versão parcial e inicial do sistema seria o protótipo mais conveniente para testes de usabilidade, mas completam que — infelizmente — aguardar uma versão executável do sistema pode postergar os testes de usabilidade para a fase de desenvolvimento. Eles concluem que um

protótipo *descartável* seria uma excelente opção no caso de os profissionais em usabilidade já possuírem bom conhecimento em alguma ferramenta de prototipação.

Prototipação rápida está relacionada com a confecção e testes de protótipos de baixa fidelidade segundo um processo iterativo de design, em período de tempo suficientemente curto. Rosson e Carroll (2002) constata que o aspecto rústico em protótipos de baixa fidelidade (como é o caso da prototipação em papel, que discutiremos a seguir) encoraj a questionamento e discussão, pois os observadores são motivados a *preencher* os detalhes ausentes. Estudos de Ward (1994) convincentemente demonstram que protótipos de baixa fidelidade apresentem grande eficácia nos testes de design para layouts de control e e exibição. Eles se mostram bastante valorosos para avaliar os aspectos cognitivos do design. Entretanto, protótipos de baixa fidelidade falham na avaliação dos atributos físicos da interface, como *feedback* tátil, auditivo e também, em certa medida, visual. Na prática, diferentes protótipos e seu nível associado de fidelidade podem fornecer tipos diversos de informação de design. Rosson e Carroll (2002) listam algumas das vantagens e desvantagens em se utilizar protótipos de baixa fidelidade. As vantagens estão relacionadas ao fato de que tais protótipos:

- São mais rápidos de serem criados e iterados;
- Apresentam menor custo no planejamento de recursos;
- Requerem menor investimento e minimizam a resistência a mudanças por parte da equipe de desenvolvimento;
- Demandam poucas habilidades especiais e aumentam a participação da equipe;
- Possuem formato flexível e podem ser adaptados a cada situação;
- Podem ser construídos em qualquer ponto do processo.

Os autores observam também algumas desvantagens, pois protótipo s de baixa fidelidade, entre outras:

- Não conseguem cobrir testes de performance;
- Não avaliam detalhes estéticos;
- Não apresentam boa escalabilidade para problemas mais complexos.

Os autores afirmam que gráficos e animações de alta qualidade podem ser utilizados para se criar protótipos convincentes; entretanto, isso pode conduzir a um engajamento (comprometimento) prematuro. Hall (2001) também demonstra que protótipos de maior fidelidade revelam mais problemas, além de fornecer *feedback* mais completo sobre os problemas de usabilidade no design original. Contudo, isso ocorre a um custo mais elevado, devido, por exemplo, à criação de interfaces sensíveis ao toque ou algum outro recurso mais caro. O autor, portanto, conclui que é provável haver um *trade-off* entre o tipo e qualidade da informação coletada e os recursos investidos para se obter aquela informação. A partir dos resultados obtidos em seus estudos de avaliação de protótipos, com diferentes níveis de fidelidade e critérios de usabilidade apropriados, Hall (2001) constata que o tipo de protótipo a ser utilizado em um contexto de design em particular dependerá de cinco fatores críticos, como segue:

1. **Que produto está sendo projetado?** Se o produto apresenta apenas uma função e possui interface simples, então ele pode ser testado adequadamente utilizando-se um protótipo de baixa fidelidade, como, por exemplo, um simples *mockup* em papel. Contudo, se o produto sendo projetado é multifuncional e apresenta uma interface mais complexa, podem ser necessários testes com vários tipos diferentes de protótipos. Neste caso, a abordagem supra-citada de protótipos horizontais e verticais proposta por Nielsen (1993) pode ser utilizada para se examinar tópicos de design.
2. **Que informação de projeto estão sendo solicitadas?** Se a informação procurada diz respeito, por exemplo, à otimização de layouts de controle, então alguns *mockups* de baixa fidelidade podem ser suficientes. Já se a informação requerida for sobre a melhor experiência sensorial (*e.g. feedback* tátil) de, por exemplo, chaves, botões, switches e telas, então recomenda-se um maior nível de fidelidade em testes que deveriam, preferencialmente, seguir-se àqueles para se determinar o layout ótimo.
3. **Qual o estágio do processo de design?** De forma geral, quanto mais avançado o estágio no processo de projeto em que nos encontrarmos, mais apropriado será utilizar protótipos de maior fidelidade. Entretanto, pode haver situações onde ainda seja adequado testar *mockups* de baixa fidelidade mesmo nas últimas etapas do processo de design.

4. **Que recursos estão disponíveis?** Provavelmente está dentro das capacidades e recursos da maioria das empresas construir protótipos de baixa fidelidade, tais como *mockups* de papel. Se a companhia for de maior porte e empregar pessoal com expertise em tecnologia, a criação de protótipos sensíveis ao toque, por exemplo, poderá apresentar uma boa relação custo-benefício. O mesmo poderia ser dito com respeito ao desenvolvimento de protótipos sólidos melhor elaborados.
5. **Quais seriam os custos de se desenvolver um projeto de baixa qualidade?** No caso de produtos para os quais haja altos custos de produção e/ou implantação em larga escala, a compensação econômica de se elaborar protótipos de maior fidelidade poderia ser facilmente comprovada. No entanto, Hall (2001) argumenta que deve haver primeiramente testes com protótipos de baixa fidelidade, para se deduzir o design ótimo a partir de uma perspectiva engonômico-cognitiva, para — apenas posteriormente — proceder-se aos testes de aspectos ergonômico-físicos do design.

Ainda em seus estudos, o autor conclui, sobre o tópico mais abrangente de prototipação rápida para desenvolvimento de novas tecnologias, que:

- É possível obter-se boa informação de design a partir de testes com usuários, utilizando-se protótipos de baixa fidelidade mais cedo no processo de design;
- A fidelidade do protótipo determina o tipo de informação que pode ser obtida a partir dos testes com usuários;
- O valor da informação de design depende da fidelidade do protótipo avaliado;
- Em sistemas multimídia, informações cognitivas do design devem ser coletadas e analisadas antes daquelas de natureza física, tais como *feedback* tátil ou auditivo;
- O retorno econômico da prototipação está relacionado aos recursos investidos na coleta de informação de design, relativos ao custo potencial de se produzir um design de baixa qualidade.

Escolhido o grau de fidelidade adequado para avaliar os protótipos em cada etapa do projeto, faz-se necessário planejar e adaptar a melhor forma de condução de um estudo de usabilidade com protótipos. A seguir, apresentaremos algumas estratégias de design para a aplicação de testes de prototipação rápida com usuários.

3.2 Estratégias de design

Nielsen e Mack (1994) têm fortemente argumentado em favor de uma combinação de avaliações heurísticas e testes de protótipos com usuários. Eles recomendam que, primeiramente, sejam feitas avaliações heurísticas sobre o protótipo para "*limpar a interface e remover tantos problemas óbvios de usabilidade quanto possível*" para, então, proceder-se aos testes com usuários sobre o design já revisado. Os autores, entretanto, não mencionam exatamente quantos avaliadores deveriam ser utilizados nesse processo de design iterativo, mas concluem que — severamente falando — "*avaliações de um único especialista em usabilidade seriam ainda pouco creditáveis de confiança* [...] *mas que avaliações de três ou quatro profissionais seriam satisfatórias a muitos propósitos práticos*".

Embora Hall (2001) concorde com essa visão, o autor constata ser extremamente difícil construir uma estratégia unificada que englobe todas as situações de design. Sendo assim, ele sugere que uma estratégia de design apropriada, baseada em testes com usuários, seria:

1. Selecionar o mais baixo nível de fidelidade, convenientemente apropriado ao produto que está sendo projetado;
2. Selecionar uma pequena amostra representativa de usuários-alvo;
3. Selecionar tarefas relevantes para serem executadas;
4. Selecionar critérios apropriados de usabilidade para medição;
5. Executar testes em sessões com usuários de uma maneira científica: controlar a tendência dos usuários a dar as respostas que ele considera que agradariam os avaliadores, enfatizar que o produto — e não o usuário — está sendo testado e utilizar a técnica *thinking-aloud* com entrevistas aos usuários após o teste;
6. Re-projetar o protótipo (baseando-se nos problemas encontrados) em um nível maior de fidelidade;
7. Repetir o processo caso seja necessário.

Hall (2001) afirma que, enquanto sua estratégia está apoiada em testes com usuários como ferramenta de avaliação, avaliações heurísticas podem mostrar-se mais efetivas na relação de custo-benefício dos que testes com usuários, em estágios apropriados ou com certos tipos de protótipos. O autor reconhece que mesmo um especialista em usabilidade, ou em interação

homem-computador, com bom conhecimento de causa sobre o problema específico de design seria capaz de fornecer direção à avaliação durante o processo de projeto.

A estratégia de design proposta por Snyder (2003) vai mais adiante, preocupando -se inclusive com as pessoas, as atividades e o tempo necessário em cada etapa do processo proposto. A autora resume os passos essenciais do que seria um estudo de usabilidade com prototipação rápida (*i.e.* uma série de testes conduzidos com uma dada interface e u m conjunto de tarefas), conforme o processo a seguir:

- Determinar os objetivos dos testes — o que se pretende aprender com eles?
- Quem são os usuários? Após sua definição, segue -se o recrutamento dos mesmos;
- Elaborar tarefas relacionadas às atividades reali zadas pelos usuários;
- Criar as peças do protótipo necessárias para se executar aquelas tarefas;
- Realizar *walkthroughs* internos de preparação para os testes;
- Conduzir vários testes de usabilidade, refinando o protótipo após cada um deles;
- Estabelecer priori dade para os problemas observados;
- Planejar alterações adicionais na interface a curto prazo e/ou mantê -las em uma relação, tal que possam ser posteriormente resolvidas;
- Comunicar as descobertas àqueles que não estiveram diretamente envolvidos.

Snyder (2003) baseia o processo descrito em sua experiência à frente de dezenas de projetos executados no contexto de mercado. A autora descreve a visão geral do que seria uma agenda para estudos de usabilidade com protótipos, definindo o que acontece em cada ativid ade, quem são os responsáveis e o tempo previsto em cada atividade:

1. **Reunião inicial:**
   Todos os *stakeholders* devem participar deste encontro, quando se discutem objetivos, riscos e preocupações. O tempo previsto é de *três horas*.
2. **Recrutamento de usuários:**
   Deve-se procurar por pessoas que se encaixem no perfil de usuário e agendar os testes com ela. Caso o projeto em desenvolvimento seja para uma empresa ou instituição em particular, duas pessoas do time são suficientes para realizar o recrutamento. A autora c onstata que muitas companhias, entretanto, terceirizam esta

atividade. O tempo para concluir o recrutamento varia normalmente *entre duas e três semanas*.

3. **Design de tarefas:**

   Nesta etapa, especificam-se as tarefas que serão executadas nos testes de usabilidad e. Deve participar desta atividade o núcleo do time de desenvolvimento, auxiliado por alguém que tenha um bom conhecimento sobre o domínio daquilo que deve ser testado. O design de tarefas deve durar *entre três e cinco horas*.

4. **Criação do protótipo e walkthroughs:**

   O núcleo do time de desenvolvimento deve listar elementos da interface necessários à realização das tarefas, dividir o trabalho, executar *walkthroughs* periódicos e formalizar a seqüência de ações (*script*) sem usuários reais antes dos testes de usabilidade. O tempo previsto é de *meio a cinco dias*.

5. **Testes de usabilidade e refinamento iterativo:**

   Deve-se executar os testes de usabilidade (a maioria dos quais dura entre uma e duas horas), listar os problemas encontrados após cada teste e revisar o protóti po antes do próximo teste. Todos os *stakeholders* devem participar desta atividade, cujo tempo total previsto é de *dois dias*.

6. **Priorização de problemas e plano de ação:**

   Nesta atividade, problemas que não foram resolvidos são priorizados, aqueles mais graves são discutidos (bem como possíveis soluções), um plano de ação para resolver os problemas encontrados é elaborado e os mesmos são listados. Todas as pessoas da equipe que contribuíram em um ou mais testes devem participar. O tempo médio desta atividade é de *três horas*.

7. **Comunicação de resultados**

   Um ou dois membros da equipe (geralmente, mas não necessariamente, os especialistas em usabilidade) devem escrever um sumário dos dez maiores problemas descobertos, disponibilizar os resultados na intranet da empresa , redigir um relatório, apresentar os resultados obtidos, editar um vídeo com o *walkthrough* observado e criar especificações para a implementação da interface.

Essa estratégia é aplicada em particular nos processos de prototipação em papel, técnica sobre a qual trataremos a seguir. Ela é conhecida pelo seu baixo custo, alto *feedback* e por ser de grande utilidade e facilidade de uso por não-especialistas. Apresentaremos a técnica de

prototipação em papel e suas vantagens principais, planejamento, projeto de tarefas, preparação de protótipos, papéis e responsabilidades na condução dos testes com usuários, bem como o tratamento de dados.

3.3 Prototipação em papel

Num sentido mais amplo, pode-se considerar a prototipação em papel como um método de *brainstorming*, projeto, criação, teste e comunicação de interfaces gráficas com usuários. Segundo a definição de Snyder (2003), "*prototipação em papel é uma variação de testes de usabilidade onde usuários representativos executam tarefas reais através da interação com uma versão em papel da interface manipulada por uma pessoa 'atuando como um computador', que não explica como a interface foi projetada para funcionar*". A autora salienta que esta técnica é independente de plataforma, podendo ser utilizada para websites , aplicações para internet, software, dispositivos *handheld* e mesmo hardware — tudo que possui uma interface homem-computador pode ser objeto da prototipação em papel. A Figura 3.2 abaixo apresenta um exemplo de protótipo em papel de uma caixa de diálogo:

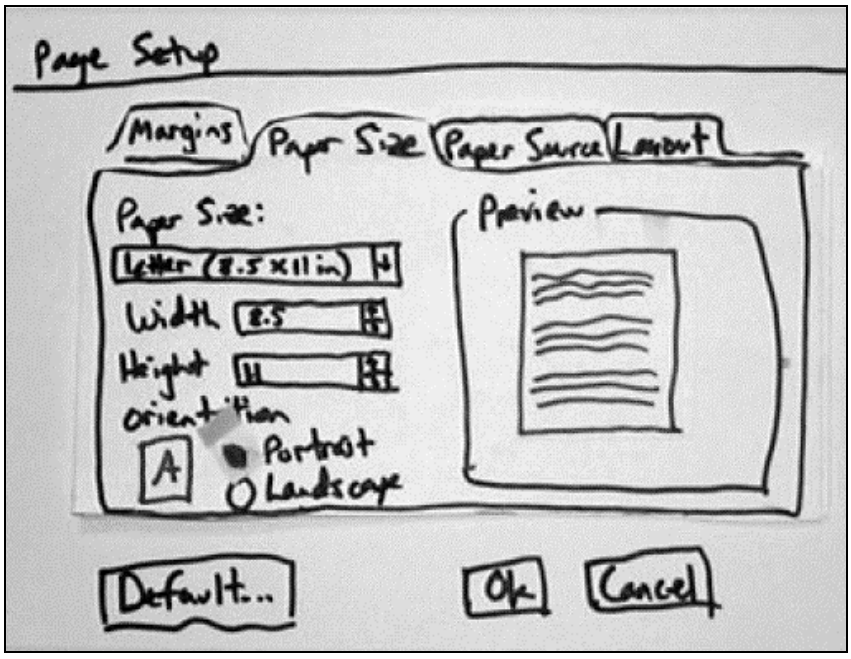

Figura 3.2 — Protótipo de uma caixa de diálogo, com todos os elementos da interface prototipados em papel.

Rosson e Carroll (2002) avaliam que protótipos em papel tornam-se ainda mais importantes em projetos que utilizam métodos de design participativo. Geralmente, usuários-alvo de um sistema ou outros *stakeholders* não possuem tanto conhecimento em computação quanto o time de desenvolvimento. Os autores argumentam que trabalhar sobre protótipos elaborados com papel e caneta minimiza diferenças de experiência com computadores e, pelo contrário, enfatiza a importância do próprio domínio. Visto que tais protótipos são criados com materiais comuns de escritório, uma grande faixa de indivíduos podem contribuir diretamente com idéias de design, o que aumenta o envolvimento e engajamento de *stakeholders* no projeto. Snyder (2003) lista outros benefícios da prototipação em papel:

- Fornece *feedback* substancial dos usuários nas primeiras etapas do processo de desenvolvimento, ou seja, antes que esforços de implementação sejam investidos;
- Promove um rápido desenvolvimento iterativo, experimentando-se várias idéias;
- Facilita a comunicação dentro do time de desenvolvimento, bem como entre este e seus clientes;
- Não requer habilidades técnicas, permitindo que mesmo um time multidisciplinar trabalhe junto;
- É uma técnica de baixíssimo custo;
- Encoraja criatividade no processo de desenvolvimento do produto.

Hackos e Redish (1998), por outro lado, listam algumas desvantagens desta técnica:

- Os protótipos geralmente reproduzem apenas algumas das funcionalidades finais;
- Aquilo que é representado pode não estar presente na tecnologia final;
- Eles requerem que um facilitador humano simule o que o computador faria;
- Esta técnica pode não ser levada suficientemente a sério pelos usuários;
- Protótipos em papel são mais fáceis de se perder.

A Figura 3.3 abaixo mostra como organizar peças de um protótipo em um envelope:

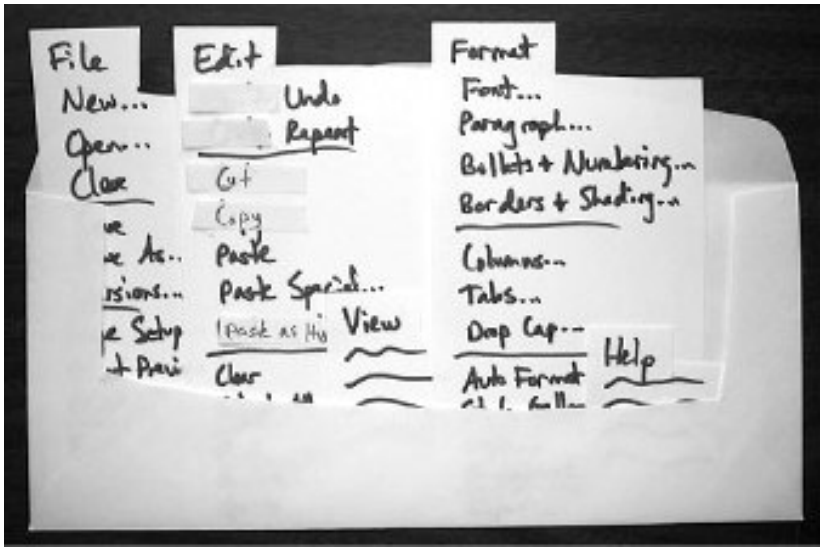

Figura 3.3 — Envelope contendo as peças prototipadas.

Contrariamente a esta última suposição, Snyder (2003) argumenta que protótipos em papel — além de se revelarem particularmente agradáveis àquelas pessoas que não se sentem seguras no uso de computadores — deixam explicitamente claro àqueles usuários com boa experiência em tecnologia (*e.g.* administradores de rede e desenvolvedores de software) que seu objetivo é obter *feedback* antes que o projeto esteja fechado. A autora constata ainda que a técnica de prototipação em papel produz efeitos psicológicos interessantes em termos da maneira em que os usuários respondem a um design não terminado que lhes é apresentado:

- **Mais familiar que um computador:**

  Muitas pessoas, especialmente as mais jovens, sentem-se bastante confortáveis com o uso de computadores; mas ainda há uma porcentagem substancial de adultos que não se sentem da mesma forma. Quando se deparam com alguma tecnologia pouco familiar, há um agravamento no risco de os usuários sentirem-se constrangidos quando não conseguem entender algo, por exemplo, em um teste de usabilidade. Por outro lado, com protótipos em papel não há a presença de computadores, apenas seres humanos. Deve-se ter em mente que a audiência-alvo de um sistema pode incluir pessoas que não se sentem confiantes com o uso de computadores.

- **Feedback mais criativo:**

  A autora mostra haver evidências de que um design inacabado encoraja uma reposta mais criativa por parte dos usuários, além de exercer um efeito dramático sobre como os *stakeholders* se tornam mais envolvidos com o projeto. Ao invés de

assumir uma atitude passiva, uma pessoa a que se apresenta um design não concluído torna-se mais criativamente comprometida em refletir sobre conceitos e funcionalidades delimitados no protótipo.

- **Aspectos estéticos postergados:**
  A menos que se esteja nas etapas finais de desenvolvimento ou especificamente buscando *feedback* sobre o design visual, não é de grande utilidade colher comentários sobre, por exemplo, formas e cores contidas em um protótipo . Protótipos em papel desenhados a mão-livre evitam *feedback* visual, pois se torna óbvio que a essa técnica não especifica aspectos estéticos, o que induz os usuários a manterem foco nos conceitos e funcionalidades do sistema prototipado.

A seguir, apresentaremos os principais pontos que devem ser observados ao se planejar um estudo de usabilidade com prototipação em papel, tais como as pessoas que devem ser envolvidas no processo, as atividades necessárias e o tempo sugerido para cada uma destas.

3.3.1 Planejamento do estudo

Na seção 3.2, apresentaram-se algumas estratégias de design para o processo de prototipação, dentre elas a de Snyder (2003), que tem sua utilidade voltada particularmente ao planejamento de estudos de usabilidade com protótipos em pa pel. Em tal estágio, não há ainda a preocupação em se elaborar qualquer protótipo: primeiramente, os objetivos de alto nível devem ser acordados e tarefas para verificação de usabilidade devem ser detalhadas, para somente então se desenvolver um protótipo em papel e testá-lo.

Prototipação em papel e testes de usabilidade devem envolver todos os membros da equipe de projeto, não apenas os profissionais de usabilidade e design. A autora analisa que até mesmo decisões de fatores técnicos, como o projeto de ban co de dados, pode causar impacto sobre a interface gráfica e vice-e-versa. Em outras palavras, o número de decisões no dia-a-dia de um projeto que afetam o usuário é demasiadamente grande para ser gerenciado por apenas uma pessoa ou mesmo um departamento. Em empresas cuja equipe é composta por muitas pessoas, recomenda-se que seja designado um time-núcelo para as atividades e estudos de usabilidade, pois não é provável (nem desejável) que toda a equipe de desenvolvimento esteja envolvida em *cada* aspecto da preparação e teste do protótipo. Tipicamente, o time-núcleo deveria incluir os designers e desenvolvedores responsáveis pela interface gráfica, bem como um especialista em

usabilidade — caso haja algum na equipe. Snyder (2003) adverte ser importante possuir no time-núcleo pelo menos uma pessoa com ponto de vista técnico — que conheça a arquitetura do sistema, os limites e restrições tecnológicos. Isso impediria a equipe de desenvolver um protótipo que fosse impossível ou extremamente difícil de ser implementado. No início de um dado projeto que envolva prototipação em papel, faz-se necessário haver discussões entre todos os membros da equipe sobre os objetivos e métodos que serão utilizados nos testes de usabilidade. Uma agenda típica para as reuniões iniciais, segundo Snyder (2003), seria:

- **Fornecer uma visão geral das atividades:**
  Nas primeiras vezes que se fazem prototipação com papel e testes de usabilidade, é natural que muitas pessoas na equipe tenham questões sobre essas atividades. Pode-se apresentar uma versão prototipada em papel de alguma aplicação já conhecida pela equipe, pois — embora seja complicado explicar o que é a técnica — as pessoas logo a reconhecem quando a vêem em prática. Os membros que não estiverem diretamente envolvidos no time-núcleo devem estar cientes sobre quais atividades requererão sua participação. A autora defende que cada pessoa participe, pelo menos, de uma reunião para definição de *walkthrough* (preferencialmente a última antes dos testes), dois testes de usabilidade (para se avaliar semelhanças e diferenças entre os problemas levantados em ambos) e da reunião de comunicação de resultados.
- **Discutir riscos e preocupações:**
  O valor real da prototipação em papel está na sua habilidade em apontar problemas nas primeiras etapas do processo de desenvolvimento, enquanto ainda há tempo de evitá-los. Portanto, os testes de usabilidade devem ser planejados para responder as principais dúvidas da equipe sobre os aspectos da experiência do usuário. Snyder (2003) defende que prototipação em papel é uma excelente ferramenta para gerenciamento de riscos porque auxilia a clarificar aquilo que não se conhece sobre o quão bem a interface gráfica trabalhará, o que contribui para que a equipe tome decisões importantes o mais cedo possível. Alguns questionamentos a serem feitos a cada membro da equipe durante este estágio seriam:
  – Sobre que pontos há mais insegurança?
  – Quais são os maiores riscos sob uma perspectiva de negócio?
  – Que decisões de projeto importantes ainda devem ser tomadas?

  - Que tarefas são mais críticas para os usuários, mesmo que sejam as menos freqüentes?
  - Que partes do sistema apresentam um caráter inovador para a equipe?

- **Criar um perfil de usuário:**

  A equipe deve estar ciente do tipo de usuário que será recrutado para os testes de usabilidade. A discussão deve ser iniciada tratando sobre que características os usuários-alvo do sistema possui (pode haver mais de um perfil). As seguintes questões sobre os usuários podem ser úteis nesta etapa:
  - Quais são suas responsabilidades no trabalho que desempenham?
  - Qual seu nível de familiaridade com a tecnologia e os conceitos utilizados pelo produto em desenvolvimento?
  - Que produtos similares eles compraram e utilizaram?

  A partir das características que forem mencionadas, deve -se elaborar uma lista com todas elas e identificar-se aquelas que capturam a essência dos usuários em questão.

- **Determinar o cronograma:**

  Há projetos de desenvolvimento em que as equipes podem encontrar muitas dificuldades para interagir com usuários reais. Sendo assim, Snyder (2003) sugere que o recrutamento de usuários e o agendamento dos testes de usabilidade sejam feitos antes mesmo da elaboração dos protótipos. A autora constata que a maioria esmagadora dos testes de prototipação com usuários duram entre uma e duas horas e que, segundo a teoria de Nielsen e Landauer (1993), testes com cinco a oito usuários já poderiam fornecer dados suficientes para a identificação dos padrões principais. Para tanto, é necessário que seja testado o mesmo conjunto de tarefas com usuários pertencentes a um mesmo perfil, como foi anteriormente definido. Snyder (2003) recomenda-se que seja previsto um período entre os testes para a revisão dos problemas encontrados e alteração do protótipo em papel.

3.3.2 Design de tarefas

Snyder (2003) afirmam que o design de tarefas é um dos aspectos mais importantes em testes de usabilidade: não basta observar os usuários interagirem como quiserem com a interface; é preciso definir tarefas específicas que revelem aquilo que se deseja aprender. Hackos e Redish (199 8) informam que, uma vez decidido que a técnica de prototipação deverá integrar o processo de

projeto, deve-se determinar como ela deve ser iniciada. Dependendo da situação, algumas preocupações primárias poderiam ser:

- O fluxo de telas para tarefas maiores;
- A metáfora geral e como ela será conduzida;
- O layout da tela principal de tarefas;
- O layout de todas as outras telas;
- Metáforas alternativas ou outras idéias de design.

Os autores explicam que a prática comum na área é de que inicie com aquelas tarefas mais críticas e freqüentes. Haveria uma estagnação na prototipação caso todas as variações e tarefas menores fossem discutidas, testadas e avaliadas desde o começo do processo. Uma boa tarefa fornece dados confiáveis sobre tópicos-chave, tal que se obtenha o máximo *feedback* dos testes com usuários. Por outro lado, tarefas mal escolhidas podem mascarar problemas ou mesmo transformar em um aquilo que não é um problema em si. Snyder (2003) aponta algumas das características que se deve considerar para a design de boas tarefas:

1. **A tarefa baseia-se em um objetivo relevante ao perfil de usuário escolhido:** Algumas questões sobre o real objetivo dos usuários no perfil escolhido devem estar sempre em mente na definição de tarefas, como: para que as pessoas utilizarão a interface? O que é importante para elas? Como o sistema auxilia os usuários a alcançar seus objetivos? A autora adverte que, quando a equipe está acostumada a raciocinar sobre a interface sempre em termos de funcionalidade, pode ser difícil passar a pensar sobre os objetivos do usuário. Nesses casos, deve-se procurar argumentar *por que a funcionalidade é interessante* até que as respostas indiquem um objetivo real dos usuários.
2. **Ela considera questões importantes para o sucesso do produto ou negócio:** Questões específicas são mais fáceis de se responder, ao passo que questões mais gerais são geralmente mais importantes para o sucesso do sistema. Uma maneira de priorizar questões que devem ser respondidas pelos testes de usabilidade é solicitar que cada pessoa na equipe escolha as três questões mais importantes sob sua perspectiva. Snyder (2003) ainda esclarece que questões mais gerais podem, às vezes,

ser esclarecidas através de padrões observados nos testes de várias questões mais específicas.

3. **Ela possui um escopo bem definido:**

   O escopo de uma tarefa deve ser abrangente o suficiente para que o usuário alcance um objetivo real e responda uma ou mais questões importantes. Tipicamente, tarefas com escopo bem definido requerem entre cinco e trinta minutos para s erem completadas em um teste com protótipos em papel.

4. **A tarefa deve possuir um conjunto finito e previsível de possíveis soluções:**

   Visto que protótipos em papel não são completamente funcionais (ou seja, não representam todas as funcionalidades pretendidas para um sistema), seria impossível confeccionar todas as peças necessárias para simular a interação do usuário com o sistema completo. Assim, deve-se introduzir algumas restrições nas tarefas com o propósito de limitar a quantidade de preparação necessári a para os testes. Por outro lado, Snyder (2003) recomenda que sempre deve haver pelo menos uma solução para cada tarefa. A autora explica que muitos especialistas em usabilidade desaconselham aquelas tarefas onde se solicita ao usuário que alcance algo que , sabe-se, é impossível.

5. **Ela possui um ponto de término que o usuário possa reconhecer:**

   A autora argumenta que tarefas sem um término definido devem ser evitadas, tais como "*explore este site*" ou "*descubra algo interessante sobre este módulo*". Os usuários não se sentiriam seguros se estão realmente fazendo a coisa correta e a tarefa seria interrompida em algum ponto arbitrário. O protótipo deve fornecer *feedback* suficiente para que o próprio usuário reconheça que concluiu a tarefa.

6. **Uma boa tarefa elicita ações, não apenas opiniões:**

   Uma tarefa bem projetada deve levar o usuário a interagir com o protótipo, não apenas observá-lo e dizer o que ele pensa. Snyder (2003) ressalta que — geralmente — aquilo que os usuários dizem gostar nem sempre é um indicativo con fiável de que eles podem realmente usar a interface sem problemas. A autora avalia que, durante um teste de usabilidade, os observadores devem estar atentos a maior parte do tempo em como os usuários trabalham com a interface, não preocupando -se em conversar com eles. Questionamentos sobre a opinião dos usuários devem ser feitos ao fim de cada sessão, conforme detalharemos mais adiante.

3.3.3 Preparação do protótipo

A criação de um protótipo em papel acontece de maneira iterativa com atividades de *walkthrough* — elaboram-se algumas peças do protótipo, executa-se um *walkthrough*, verifica-se o que está errado ou faltando e repete-se este processo. Snyder (2003) afirma que a melhor maneira de criar um protótipo vai depender do tamanho e composição do time e em que etapa no desenvolvimento o sistema se encontra. No caso de interfaces que ainda não existem de fato ou que estão sendo substancialmente modificadas, a melhor abordagem seria 'dividir para conquistar': cada parte do protótipo ficaria sob o encargo de alguma pessoa. A autora defende ainda o uso de design paralelo: uma técnica em que os membros da equipe trabalham a sós ou em pequenos grupos para gerar diferentes idéias de design para, então, revisar as outras idéias. A Figura 3.4 abaixo mostra duas idéias de design, prototipadas em papel, para um mesmo engenho de busca:

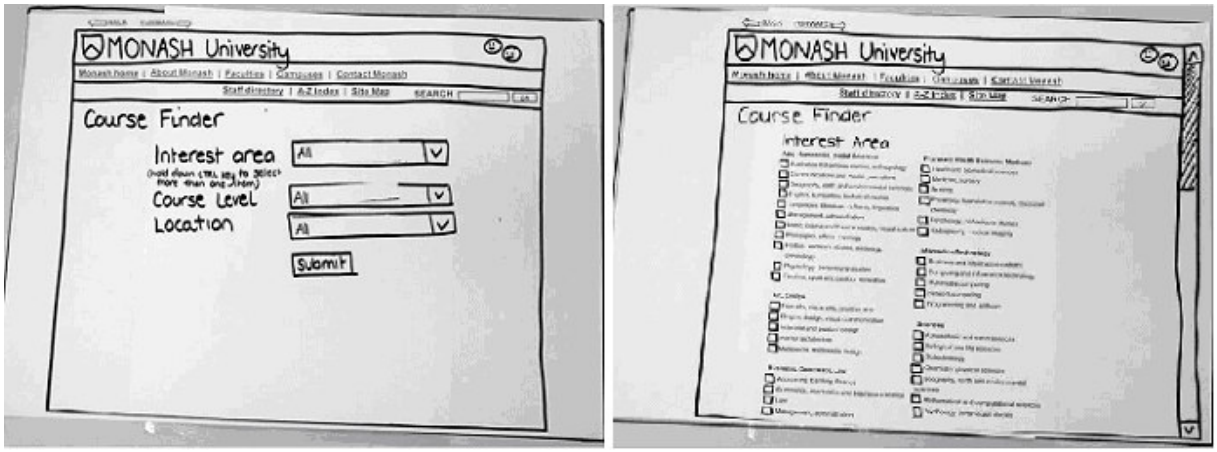

Figura 3.4 — Duas idéias de design, prototipadas em papel, para um engenho de busca.

Hackos e Redish (1998) detalham que um *kit* para sessões de prototipação em papel pode incluir papéis recortados na forma apropriada para representar a área da interface, canetas e marcadores coloridos, cartões indexados, notas *post-it* de cores e tamanhos diferentes, fita de fácil remoção, apagadores, tesouras, régua e peças pré-desenhadas que sejam relevantes ao projeto. A Figura 3.5 exibe um típico *kit* para prototipação em papel:

Figura 3.5 — *Kit* típico para a construção de protótipos em papel.

Snyder (2003) acredita que desenhos à mão-livre misturados à pedaços de tela impressos são úteis para transmitir as idéias de design aos usuários, embora se devesse optar por apenas um tipo de representação quando possível. Duas perguntas principais poderiam guiar o time na direção da escolha mais apropriada: quanto se espera que a interface seja alterada e qual representação é mais rápida?

Snyder (2003) recomenda que, antes dos testes com usuários, três atividades devem ser executadas pela equipe: *walkthroughs* internos, um *walkthrough* final e testes-piloto. *Walkthroughs* internos auxiliam a equipe a conduzir testes de usabilidade com prototipação em papel, pois são similares a estes últimos (há igualmente protótipos e tarefas), não havendo — contudo — a participação de usuários externos. Neles, os membros do time agem como 'usuários experientes', executando cada ação do modo esperado pela equipe. Esta atividade pode ser bastante útil à equipe, pois ela ajuda a:

- Identificar partes do protótipo que ainda precisam ser criadas;
- Preparar caminhos diferentes que o usuário pode racionalmente tomar (estejam aqueles corretos ou não);
- Perceber como se espera que a interface opere, o que é especialmente útil àqueles responsáveis pela documentação, treinamento ou os novatos no time de desenvolvimento;

- Identificar tópicos relacionados às características técnicas do produto.

O *walkthrough* final — também chamado de ensaio do teste de usabilidade — não se refere especificamente ao protótipo em papel em si , sendo um importante passo na preparação para qualquer tipo de testes com usuários. T al ensaio deve acontecer, idealmente, no dia anterior aos testes e envolve um grupo maior que aquele que trabalhou diretamente na elaboração do protótipo. O objetivo aqui é assegurar que todos estão preparados para o teste. O *walkthrough* final deve ser conduzido da mesma maneira que um teste de usabilidade, onde o papel do usuário é ainda exercido por alguém da equipe, mas cujos objetivos são:

- Familiarizar os observadores com o protótipo;
- Coletar questões da equipe;
- Estimar o tempo necessário para cada tar efa;
- Decidir quando se deve intervir nos testes;
- Criar um plano (roteiro) alternativo para os testes.

Já a finalidade de um teste-piloto é refinar as tarefas e a metodologia usada nos testes principais — diferentemente dos testes de usabilidade, onde o ob jetivo é refinar a interface. Nos testes-piloto, a participação de um usuário real é imprescindível. Snyder (2003) diferencia um teste-piloto de um teste comum de usabilidade, pois no primeiro o usuário deve expressar sua opinião sobre as tarefas e instruções em si, não apenas executá-las. Todas essas atividades (*walkthroughs* internos, *walkthrough* final e testes-piloto), incluindo o próprio teste de usabilidade, devem ser protagonizadas por alguns membros da equipe que assumem papéis especiais, e é sobre isso que trataremos a seguir.

3.3.4 Papéis e responsabilidades

Depois que o protótipo foi criado, os testes de usabilidade devem, então, ser conduzidos; eles envolvem pessoas que são representativas para o perfil de audiência anteriormente acordado pela equipe. Solicita-se ao usuário que execute algumas tarefas, interagindo diretamente com o protótipo — 'clicar', tocando-se nos botões ou *hyperlinks* do protótipo, ou 'digitar', escrevendo dados sobre o mesmo. Uma ou duas pessoas da equipe interpretam o ' *computador*', manipulado

as peças de papel para simular o comportamento da interface sem, contudo, explicar como ela foi projetada para funcionar. Um *facilitador* (geralmente alguém treinado em usabilidade) conduz a sessão, enquanto outros membros da equipe atuam como *observadores*, tomando notas.

Para que se obtenha o máximo de aproveitamento nas sessões de testes com usuário, é necessário que haja organização entre as atribuições de cada ator. Algumas das responsabilidades de cada papel (computador, facilitador e observador) são:

- **Computador:**

  Este membro da equipe deve executar apenas as operações e seqüências que uma máquina programada realizaria. Ele deve refletir, através da manipulação das peças do protótipo, as entradas dos usuários no sistema. Mesmo que o pr ocessamento interno do sistema venha (e deve vir) a ser completamente desconhecido pelo usuário, este precisa compreender as relações de causa -e-efeito entre suas entradas e as saídas do sistema. O 'computador' deve sempre aguardar pelas ações dos usuários antes de agir sobre o protótipo, evitando antecipar o que eles poderão vir a fazer. Este ator deve sempre evitar falar ou conversar com os usuários durante a execução do teste. Contudo, Snyder (2003) permite que tal seja feito em algumas situações: para diferenciar se o usuário está ordenando um comando ou apenas emitindo um comentário, para esclarecer o que o usuário veria no sistema final (*e.g.* "*isto em cima são abas na janela*") e para fornecer mensagens de erro orais.

- **Facilitador:**

  Eis uma visão geral das atividades tipicamente executadas pelo facilitador de testes de usabilidade com usuários:

  – os observadores qual deve ser seu comportamento, o que devem observar e como devem tomar notas.
  – Cumprimentar os usuários, conceder -lhes uma explicação sobre o exper imento, obter sua autorização (termo de consentimento) e efetuar seu pagamento, nos casos em que a participação de usuários é remunerada.
  – Conduzir o usuário ao local de testes e apresentar a ele o 'computador' e demais observadores presentes.
  – Solicitar aos usuários que se apresentem e descrevam seu perfil.
  – Explicar o protocolo de testes (como interagir com o protótipo, como expressar-se pela técnica de *thinking-aloud* etc.).

- Facilitar cada tarefa, interagindo com os usuários quando necessário.
- Gerenciar o tempo gasto em cada tarefa, induzindo às áreas que são de maior interesse aos observadores.
- Encerrar a sessão de testes no tempo previsto.
- Agradecer aos usuários e conduzi-los.
- Listar, com os observadores, os tópicos encontrados e modificar o protótipo para o próximo teste (isto pode ser feito pelo time-núcleo).

- **Observador:**

  Os observadores devem receber uma cópia das tarefas que os usuários executarão, o que os ajudará a lembrar qual a proposta de cada uma delas e a que pontos de observação devem estar atentos. Os observadores devem tomar nota de tudo aquilo que causar-lhes surpresa durante o teste, pois provavelmente isto indicaria que os usuários estão agindo de modo diferente daquele que era esperado. Snyder (2003) recomanda às equipes em que os observadores não participaram diretamente da construção do protótipo que cada um deles esteja responsável em observar um aspecto em particular no uso da interface pelos usuários. A Figura 3.6 abaixo mostra um exemplo de usuário interagindo com parte de um protótipo:

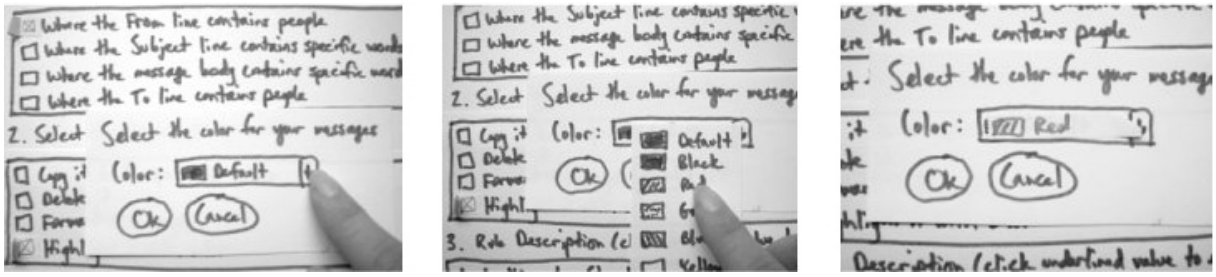

Figura 3.6 — Exemplo de listas *drop-down* prototipadas em papel.

A autora ainda apresenta uma lista de regras às quais os membros da equipe participantes de sessões com usuários deveriam estar atentos:

1. Eles (*computador*, facilitador e observadores) devem permanecer durante todo o teste;
2. Devem permanecer em silêncio enquanto os usuários estão trabalhando;
3. Devem estar conscientes de sua linguagem corporal (*e.g.* não parecerem desinteressados com a atuação do usuário);

4. Não devem revelar quantas tarefas serão avaliadas;
5. Não devem ajudar os usuário, a menos que estes estejam completamente sem direção;
6. Devem evitar questões de estética e focarem-se no problema em si;
7. Devem respeitar os participantes do teste e manter os dados dos usuários confidenciais.

Caso a equipe seja muito pequena, deverá haver a combinação de alguns desses papéis. Snyder (2003) ressalta que tal combinação deve basear-se nas habilidades de cada pessoa, mas que as abordagens mais comuns são:

- **Facilitador-observador:**
  É possível que um facilitador consiga tomar boas notas e ainda conduzir o restante dos testes, mas muitos especialistas em usabilidade argumentam que uma ou outra atividade poderá ser prejudicada.
- **Computador-facilitador:**
  Embora seja possível facilitar um teste de algo que a própria pessoa projetou, a autora considera realmente muito difícil manter-se objetivo e imune à tentação de se querer explicar ou justificar a interface do protótipo.
- **Computador-observador:**
  Esta combinação não é muito prática, pois o 'computador' estará com suas mãos ocupadas com a manipulação e organização das peças do protótipo, o que o impedirá de tomar notas relevantes com praticidade.
- **Computador-facilitador-observador:**
  Esta combinação é a mais incomum, pois impõe sérias restrições a todas as atividades, praticamente inviabilizando o estudo de usabilidade.

Hackos e Redish (1998) apresentam a possibilidade de se ter observadores em um recinto distinto daquele para sessões de prototipação, acompanhando o que está acontecendo durante o teste de avaliação de usabilidade através de monitores de vídeo. Os autores salientam que os dois pontos mais importantes que devem ser transmitidos aos observadores são:

1. Eles devem tentar capturar o comportamento do usuário e fatos interessantes sobre o usuário ou o protótipo. Uma nota do tipo "*o usuário selecionou o ícone para edição antes de selecionar o registro que ele queria editar*" é mais instrutiva do que uma outra como "*o usuário não entende como o sistema funciona*".
2. Os observadores devem manter sua concentração na observação; não devem redesenhar o protótipo imediatamente. Deve-se primeiramente verificar o que está acontecendo e identificar os problemas.

3.3.5 Tratamento de dados

Devido ao fato de toda a prototipação ser em papel, torna-se fácil modificá-la logo em seguida, ou mesmo durante, cada teste de usabilidade. Snyder (2003) constata que vários testes podem ser conduzidos em um mesmo dia e que não demora muito até que se percebam padrões emergindo do *feedback* dos usuários. Portanto, prototipação em papel permite iterar e aumentar a qualidade de interfaces rapidamente — baseando-se naquilo que é colhido a partir de usuários reais — e antes que qualquer linha de código tenha sito escrita.

Hackos e Redish (1998) apontam algumas dimensões que devem ser observadas sobre quão bem o protótipo funciona para os usuários, das quais citamos:

- Ele corresponde ao modelo mental dos usuários?
- Ele corresponde ao modo de trabalho dos usuários?
- Utiliza o vocabulário próprio dos usuários?
- O protótipo abrange as tarefas que os usuários esperam ser capazes de executar com o produto?
- Ele funciona bem em todos os cenários (*i.e.* situações) descritos pelos usuários?
- Divide bem a carga de trabalho entre o computador e os usuários?
- Provê mensagens onde e quando os usuários necessitam delas?
- Ele mantém consistência entre botões, ícones e outras ferramentas de navegação em todas as telas?
- O protótipo mantém consistência de vocabulário em todas as telas?

Snyder (2003) adverte que testes de usabilidade são uma fonte intensa de da dos e que, portanto, é importante reunir-se a equipe após os testes para os tópicos revelados por estes sejam discutidos. Essa é a reunião de comunicação de dados, cujos objetivos principais devem ser: priorizar as observações feitas durante os testes e el aborar um plano de execução das mesmas. A autora recomenda que seja utilizada a técnica conhecida por *diagrama de atividades* (a Figura 3.7 abaixo mostra a técnica de diagrama de afinidades, realizada durante a reunião de comunicação de dados):

1. Cada observador transcreve suas próprias observação de seu bloco de notas para cartões indexados;
2. Todos os cartões são dispostos sobre uma parede;
3. Cada pessoa lê todas as observações, suas e dos demais;
4. Os cartões são re-arrumados em grupos, que devem ser nomeados;
5. A equipe elege que grupos terão representam maior impacto para a próxima versão do protótipo.

Figura 3.7 — Em um diagrama de afinidades, a equipe ordena observações individuais em grupos relacionados.

Snyder (2003) esclarece que não é uma tarefa prática escrever-se um relatório detalhado sobre os resultados da prototipação em papel, mas considera que é importante comunicar tais resultados a pessoas que não participaram da técnica ou àquelas que utilizarão as informações nas

etapas seguintes do desenvolvimento do sistema. Algumas idéias para registro e comunicação dos resultados de estudos de usabilidade são:

- **Lista com os dez resultados mais importantes:**
  Esta lista é um breve resumo dos resultados já priorizados, com uma descrição de um ou dois parágrafos sobre cada um após a reunião de comunicação de dados. Para algumas empresas de pequeno porte, tal lista é suficiente.
- **Relatório metodológico:**
  Este documento único contém toda informação relacionada à metodologia utilizada no processo de prototipação (perfis dos usuários, descrição das tarefas, roteiros de testes etc.). Muitas das informações contidas neste relatório poderão ser facilmente reaproveitadas em outros projetos (*e.g.* alguns perfis de usuários).
- **Especificação da interface:**
  Em certo sentido, um protótipo em papel pode ser visto como uma especificação de interface, pois é uma coleção de telas. Entretanto, um documento mais completo registraria — além da seqüência de telas — explicações sobre o comportamento destas e algum processamento subjacente.
- **Registro em vídeo do walkthrough:**
  Um registro em vídeo do *walkthrough* pode ser um meio eficiente de sumarizar aquilo que foi aprendido a partir de um teste de usabilidade com prototipação em papel. Tal registro deve ser feito através de explicações e demonstrações de algum aspecto particular observado e é especialmente útil quando o time de desenvolvimento está espalhado geograficamente.

Os tópicos abordados neste capítulo dizem respeito às atividades executadas dentro do ciclo de design iterativo com prototipação rápida. Contudo, tal ciclo (concepção -prototipação-avaliação) é apenas uma das partes de uma metodologia mais abrangente e centrada no usuário. No capítulo seguinte, analisaremos algumas das metodologias para desenvolvimento de *learnware* e proporemos uma metodologia cujo foco está na descoberta de problemas de usabilidade antes da fase de implementação, bem como na qualidade do conteúdo pedagógico.

CAPÍTULO 4

Metodologias para o desenvolvimento de learnware

> "Given the critical importance of learning to the competitiveness of countries and the success of both individuals and companies, research on learning and learnware product development should be a top priority for nations around the world. The country that first learns how to exploit fully the potential of this new medium, and transforms this understanding into products, will have a significant competitive advantage over other nations in its capacity to develop human capital."
> — ADVISORY COMMITTEE FOR ONLINE LEARNING, *The e-learning e-volution in colleges and universities*, 2001.

Muitas metodologias concernentes a software do tipo educativo têm sido propostas. Pádua, Teixeira e Ribeiro (2003), por exemplo, apresentam uma proposta metodológica para avaliação qualitativa de software educacional infantil — MAQSEI (acrônimo para *Metodologia de Avaliação de Qualidade de Software Educacional Infantil*) — que abrange os aspectos técnico e pedagógico. O primeiro encentra-se na avaliação da usabilidade dos sistemas, na qualidade da interação usuário-computador proporcionada pela interface do software. Já o aspecto pedagógico concentra-se na avaliação da conveniência e da viabilidade de utilização do

software em situações educacionais. Oliveira, Gomes e Borges (2001) — discutindo a problemática da avaliação de software dentro do contexto do ensino e da aprendizagem de matemática — argumentam que "*um software não pode ser avaliado fora do contexto de uso, pois a adequação do software depende não apenas de suas características, mas também da forma como o software está inserido na atividade de ensino e aprendizagem*".

Entretanto, observamos que o objetivo de tais metodologias está na *avaliação* de softwares educativos, não em seu *processo de desenvolvimento*. Embora Pádua, Teixeira e Ribeiro (2003) sustentem que a MAQSEI pode ser utilizada em avaliações formativas, metodologias como esta colaboram apenas para que o desenvolvedor de software descubra defeitos e faça as modificações necessárias no programa *durante* sua produção, ou seja, já após a etapa de projeto. Essas metodologias visam atender às necessidades de instituições de ensino e professores de disciplinas específicas, que carecem de ferramentas teóricas efetivas e confiáveis para a seleção de *learnwares* a serem adotados para o apoio ao ensino.

A própria Engenharia de Software, mesmo com seus modelos e processos bem documentados, têm deixado a desejar no que se refere ao desenvolvimento de *learnware*, pois não considera com o devido cuidado a audiência de tais sistemas e suas necessidades. Lage, Zubenko e Cataldi (2001) explicam tal incapacidade "*programas educacionais possuem características bastante peculiares de acordo com os objetivos curriculares e as necessidades específicas do grupo-alvo*". O problema torna-se ainda mais drástico quando temos a incorporação de várias mídias em um mesmo software. Sob essa situação, Castro e Aguiar (1999) afirmam que "*a rapidez com que as aplicações têm incorporado recursos das mais variadas mídias de comunicação e ampliado suas áreas de utilização* [*e.g.* software educacional] *tem provocado um lapso na Engenharia de Software em elaborar procedimentos que acompanhem com a mesma velocidade e aplicabilidade a utilização destes recursos*".

Com base nessas motivações e para tentar preencher tal lacuna, algumas metodologias específicas para o desenvolvimento de software educativo têm sido sugeridas. Considerando a integração de aspectos pedagógicos às atividades convencionais de desenvolvimento de software como a núcleo de sua proposta, Lage, Zubenko e Cataldi (2001) apresentam uma metodologia cujo ciclo de vida é baseado em protótipos evolutivos com sucessivos refinamentos, pelas seguintes razões:

- É importante que se saiba o mais rapidamente se a interpretação do produto pela equipe de desenvolvimento está de acordo com as necessidades e considerações dos clientes e usuários.
- Em muitos casos, os usuários não conseguem passar uma idéia detalhada do que eles desejam. Portanto, desenvolvedores podem não saber exatamente o que criar, o que torna cada protótipo uma revisão e refinamento de requisitos do produto final.

Lage, Zubenko e Cataldi (2001) definem os seguintes estágios no ciclo de vida incremental:

1. Estudo de viabilidade sistema;
2. Definição de requisitos do sistema;
3. Especificação de requisitos do protótipo;
4. Design do protótipo;
5. Design detalhado do protótipo;
6. Desenvolvimento (codificação) do protótipo;
7. Implementação e testes do protótipo com refinamento iterativo das especificações daquele, aumentando-se seu objetivo e/ou escopo. Se o objetivo e escopo desejados foram alcançados, pode-se retornar ao passo 2;
8. Design do sistema final;
9. Implementação do sistema final;
10. Operação e manutenção;
11. Revisão para próximas versões.

Percebe-se que o quarto passo da metodologia não considera níveis crescentes de fidelidade em sua prototipação: nos protótipos, as funcionalidades são efetivamente implementadas em alguma linguagem de programação, de maneira incremental. Com isso, praticamente nenhum trabalho é empreendido na etapa 9, visto que o esforço está excessivamente concentrado em se *codificar* protótipos. Entretanto, podemos apontar vários pontos positivos nesta metodologia:

- Ela incorpora os instrumentos clássicos fornecidos pela abordagem cognitivo-construtivista na etapa de projeto.

- Sua proposta metodológica considera a construção de *learnware* sob um ponto de vista integrado, atentando aos aspectos pedagógicos presentes no ciclo de vida.
- Há uma preocupação especial na configuração dos perfis de diferentes profissionais que formam o time de desenvolvimento (professores e especialistas em pedagogia, analistas, programadores, engenheiros de software, designers gráficos, especialistas em som e vídeo etc.), no design do programa e no processo de documentação.

Castro e Aguiar (1999) propõem um ciclo de vida para o desenvolvimento de aplicações educacionais com grande carga de interatividade baseado, igualmente, em prototipação evolutiva. Nesta metodologia, o desenvolvimento teria início com atividades preliminares de preparação do ambiente, tais como a formação da equipe de trabalho e a escolha dos modelos a serem utilizados durante o processo de construção do software. Já a prototipação evolutiva englobaria atividades típicas de elicitação e implementação de requisitos — a cada versão do protótipo, novas funcionalidades seriam acrescentadas à aplicação. As atividades de implantação, no entanto, estariam fora do ciclo iterativo da prototipação, caracterizando uma prontificação para o uso. Os autores definem um quadro geral de atividades em cada etapa de desenvolvimento com seus respectivos autores, como segue:

1. **Preparação** (analista, aluno e educador)
   - Formação da equipe de trabalho
   - Escolha dos modelos de desenvolvimento
   - Projeto instrucional
2. **Análise** (analista, aluno, roteirista, educador, programador e programador visual)
   - Lista de atores candidatos
   - Identificação dos requisitos
   - Identificação das restrições do desenvolvimento de software
   - Representação dos requisitos
   - Validação dos requisitos
3. **Projeto** (analista, programador visual, programador, roteirista, educador e aluno)
   - Escolha da metáfora
   - Projeto de interfaces
   - Projeto de objetos
   - Modelagem de requisitos não-funcionais

4. **Implementação** (analista, programador e programador visual)
5. **Testes** (analista, programador, educador e aluno)
6. **Avaliação** (analista, educador e aluno)
7. **Implantação** (analista, programador e programador visual)
   - Manutenção e *up-grade*
   - Controle de versões
   - Distribuição
   - Rotinas de configuração
   - Suporte técnico

Vale salientar que também esta metodologia não considera níveis de fidelidade, sendo que cada protótipo seria efetivamente implementado em alguma linguagem de programação (os autores prevêem orientação a objetos), seu código seria testado e cada versão, avaliada — este ciclo iterativo compreenderia os passos 4 a 6. Uma boa contribuição da metodologia de Castro e Aguiar (1999) é que ela descreve a função de cada ator no desenvolvimento de software educativo:

- **Analistas:**
  Desenvolvedores responsáveis pela modelagem lógica do software.
- **Educadores:**
  Professores que utilizarão o sistema como suporte às suas aulas.
- **Alunos:**
  Futuros utilizadores do software educativo.
- **Programadores:**
  Desenvolvedores especializados na ferramenta de autoria adotada.
- **Programadores visuais:**
  Especialistas na manipulação dos recursos multimídia.
- **Roteiristas:**
  Profissionais responsáveis em seqüenciar os eventos levantados, tornando mais fácil a navegação entre eles e contribuindo para o aprendizado.

Campos, Campos e Rocha (1996) estabelecem dez recomendações, em forma de etapas, para o desenvolvimento de um produto hipermídia educacional:

1. Definição do ambiente de aprendizagem;
2. Análise de viabilidade;
3. Seleção do tipo de documento;
4. Seleção do método para autoria;
5. Planejamento da interface;
6. Planejamento do documento;
7. Seleção do sistema de autoria e das ferramentas;
8. Implementação;
9. Avaliação;
10. Validação.

As autoras constatam que "*a experiência tem mostrado que o processo de desenvolvimento de software adequado à hipermídia educacional deve ser composto do modelo de ciclo de vida de prototipagem evolutiva, acrescido da etapa inicial da escolha do ambiente educacional e avaliação por parte de professores e alunos*". Campos, Campos e Rocha (1996) afirmam ainda que, na etapa de validação do sistema, pode-se trabalhar com grupos representativos da população alvo do software através de, basicamente, duas técnicas de coleta de dados: observação direta da interação usuário-hipermídia e resposta dos usuários a um questionário.

4.1 Problemas com as metodologias tradicionais

Podemos observar que as metodologias aqui resumidas possuem a preocupação em seguir um modelo de design iterativo baseado em prototipação. Contudo, nenhuma delas considera a questão dos vários níveis de fidelidade que os protótipos podem assumir. Além disso, elas não envolvem prototipação rápida, pois seus protótipos são literalmente codificados em alguma linguagem de programação.

O problema surge quando, sem agilidade no modelo de prototipação, tais metodologias recomendam que sejam executadas apenas algumas poucas iterações em seus ciclos evolutivos. Isso tende a comprometer a qualidade dos sistemas, pois aquelas metodologias não conseguem

desconsiderar os cronogramas rígidos e apertados que a concorrência de mercado impõe aos projetos.

4.2 Metodologia proposta

Sendo assim, propomos um modelo metodológico para o desenvolvimento de softwares educativos que prima pela qualidade dos mesmos, tanto em termos de conteúdo e aprendizado quanto de aspectos de usabilidade. Para tanto, a metodologia proposta está baseada principalmente na participação de especialistas em educação (pedagogos com *expertise* em uma dada área) no time de projeto, na multidisciplinaridade da equipe, bem como num ciclo iterativo de prototipação rápida com testes de usabilidade envolvendo usuários representativos do sistema.

Assim, as métricas para validação desta metodologia são: relevância educacional do conteúdo pedagógico e o número de problemas de usabilidade encontrados antes da fase de implementação do sistema. Tal metodologia é fruto de extensa experiência acadêmica e profissional de uma equipe composta por várias pessoas com formações variadas, e vem sendo continuamente testada e refinada através de um projeto de desenvolvimento de mídias educativas para a formação de professores (o próximo capítulo descreverá este estudo de caso). A Figura 4.1 exibe as principais atividades e os documentos gerados em nossa metodologia:

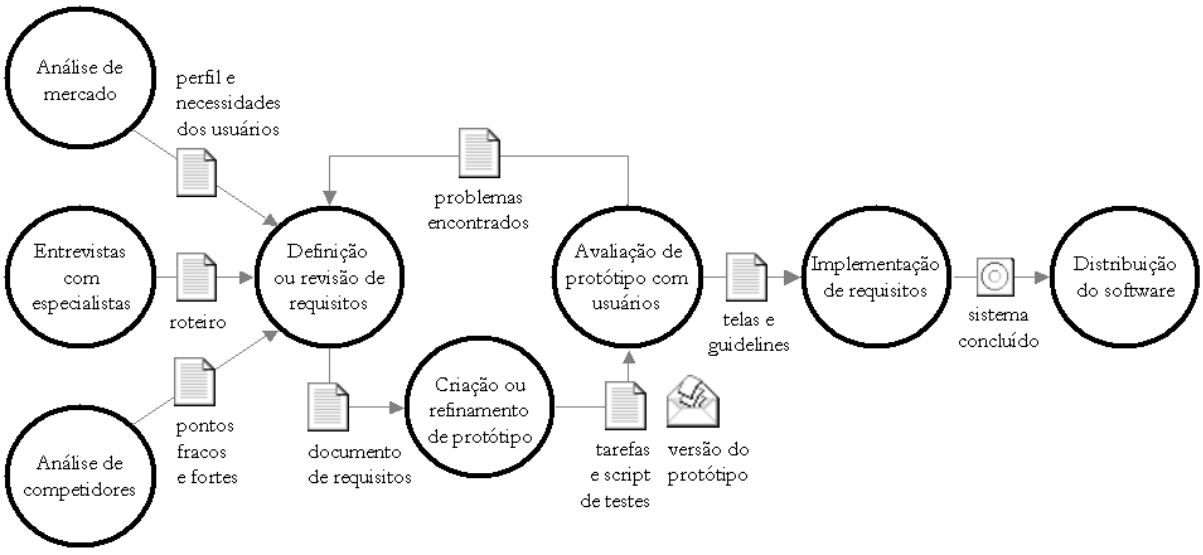

Figura 4.1 — Principais atividades e documentos gerados na metodologia proposta

A seguir, descreveremos as características de cada atividade. Salientamos que as etapas comuns à implementação foram abstraídas, pois o enfoque da metodologia está na concepção e planejamento do sistema, centrados no usuário.

- **Análise de mercado**

  Nesta etapa, procura-se sondar oportunidades de negócios na área de software educativo, buscando-se projetos que atendam a uma determinada demanda de mercado, avaliando-se as características, desejos e necessidades do consumidor (usuário) e traçando-se estratégias de marketing para a divulgação e distribuição do produto final no mercado, se for o caso.

- **Entrevistas com especialistas**

  Entendendo que o desenvolvimento de software educativo possui características bastante peculiares no que diz respeito, por exemplo, ao conteúdo educacional e modelo de aprendizado (construtivista, comportamental etc.), nossa metodologia entende ser necessário contar com a experiência de especialistas em educação, na área pedagógica de que trata o software, durante o planejamento do mesmo. Através de reuniões e entrevistas aos especialistas, os roteiristas estarão aptos a desenvolver uma seqüência de situações (roteiro) que auxiliará os usuários no aprendizado dos conceitos a que o software se propõe ensinar, o que confere qualidade de conteúdo pedagógico à metodologia proposta.

- **Análise de competidores**

  Nesta atividade, procura-se analisar os pontos fortes e fracos de sistemas concorrentes. Tais sistemas (softwares, websites, DVDs etc.) são aqueles que possuem características que o classificam na mesma categoria do produto em desenvolvimento. Através dessa análise, e equipe saberá exatamente que problemas potenciais deverão ser evitados no planejamento do produto, bem como estará apta a melhorar aquilo que já funcionou em outros sistemas.

- **Definição/revisão de requisitos**

  A partir dos resultados gerados pelas três atividades anteriores (perfil e necessidades dos usuários, roteiro e pontos fortes e fracos de sistemas competidores), pode-se proceder a uma definição parcial dos requisitos. Nesta atividade, a equipe traça algumas funcionalidades e características que se acredita serem essenciais ao produto. Ao final de cada ciclo de prototipação e testes, o documento de requisitos será

revisado de acordo com os problemas encontrados na avaliação dos protótipos junto aos usuários.

- **Criação/refinamento de protótipo**

  A definição inicial de requisitos permite à equipe de prototipação elaborar representações do sistema que serão avaliadas com os usuários. Cada protótipo criado ou refinado em um ciclo de design iterativo deve testar um ou dois aspectos do futuro sistema, ou seja, as sessões de prototipação devem possuir objetivos específicos (*e.g.* avaliar navegação e controle, conceitos da metáfora utilizada, algum módulo particular etc.). Níveis crescentes de fidelidade podem ser observados em cada iteração, dependendo do aspecto que se deseja observar (*e.g.* recursos de som e vídeo podem requerer protótipos elaborados em alguma ferramenta de autoria).

- **Avaliação de protótipo com usuários**

  Juntamente com a criação ou refinamento dos protótipos, a equipe deve determinar algumas tarefas críticas sobre o sistema prototipado que auxiliarão na observação de possíveis problemas. Um documento de uso interno à equipe — contendo um passo-a-passo de cada tarefa (*script*) e os papéis e atribuições de cada membro do time durante os testes — deverá guiar as sessões de avaliação do protótipo com usuários. Esta atividade pressupõe um agendamento prévio dos testes e deve ser realizada, preferencialmente, no ambiente real de trabalho dos usuários. Caso sejam encontrados problemas no uso de alguma funcionalidade ou módulo do sistema, os observadores deverão gerar um documento apontando as principais falhas descobertas no protótipo e uma nova versão deve solucionar tais problemas.

- **Implementação de requisitos**

  Nesta etapa acontece a codificação daqueles requisitos que foram prototipados e validados junto aos usuários (*i.e.* sua avaliação de usabilidade não revelou problemas maiores). A prototipação de requisitos validados pelos usuários pode ocorrer em paralelo à prototipação e avaliação daqueles ainda em aberto.

- **Distribuição do software**

  Após todas as funcionalidades haverem sido validadas e implementadas, o produto final estará pronto para ser implantado ou lançado no mercado de acordo com alguma estratégia de *marketing*.

As atividades de análise de mercado, entrevistas com especialistas e análise de competidores podem ser realizadas em paralelo. Elas são pré-requisitos para a definição de requisitos, que — num primeiro momento — produz uma descrição parcial das funcionalidades do futuro sistema. A partir deste documento, é elaborada uma versão de baixa fidelidade (em papel) do software que deverá ser avaliada com usuários reais em sessões de prototipação rápida. Os problemas conceituais e de usabilidade encontrados realimentarão o ciclo iterativo: os requisitos serão revisados, uma nova versão do protótipo (possivelmente com fidelidade extra) será criada e novos testes com usuários deverão ser conduzidos.

À medida que os requisitos vão sendo testados e validados junto aos usuários, pode-se proceder à implementação das funcionalidades associadas, de forma a minimizar o risco de haver atrasos no cronograma inicial do projeto. O ciclo de prototipação dos requisitos não testados continuaria a ser executado, agora em paralelo à implementação daqueles já validados. Quando todos os módulos e funcionalidades tivessem sido prototipados, testados e validados junto aos usuários e implementados pela equipe de desenvolvimento, o sistema estaria pronto para ser implantado ou distribuído.

No capítulo a seguir, descreveremos o projeto *Mídias Educativas*, uma parceria entre academia e mercado, que vem utilizando e refinando a metodologia proposta no desenvolvimento de aplicações multimídia para a formação de professores.

CAPÍTULO 5

Estudo de caso

> "Usability cost-benefit data shows that including usability in product development actually cuts the time to market and increases sales because usability and ease of use build quality into products and catch many expensive problems early on in the cycle when they can be addressed at lower cost. Finally, working with users from the beginning of a product cycle ensures that the product is being designed so that users will be satisfied."
>
> — CLAIRE MARIE KARAT, *A business case approach to usability cost justifications*, 1994.

O projeto *Mídias educativas* é fruto da parceria entre o Centro de Informática da Universidade Federal de Pernambuco e a empresa Casullo Comunicação e Design Ltda. Esse projeto visa o desenvolvimento de uma aplicação multimídia a ser utilizada por professores do ensino fundamental. Sua motivação são os baixos resultados das últimas avaliações do ensino fundamental e médio quanto à aprendizagem de matemática. Considerando a importância desse conteúdo para a formação básica e superior de recursos humanos em nosso país e suas conseqüências sobre o desenvolvimento do mesmo, o projeto busca contribuir integrando, em um único software, elementos que possam significativamente transformar esse quadro atual do ensino em nosso país.

5.1 Formação da equipe e responsabilidades

Lage, Zubenko e Cataldi (2001) atestam a necessidade de se ter vários tipos de profissionais envolvidos no time de trabalho. Eles afirmam que "*a criação de programas educativos é uma tarefa que envolve muitas áreas diferentes. A formação* [...] *de times de desenvolvimento é fundamental neste tipo de projeto educacional*". Sendo assim, a equipe do projeto está composta por cerca de uma dezena de profissionais, consultores, professores e estudantes nas seguintes áreas: gerência, pedagogia, design gráfico, engenharia de usabilidade, jornalismo, psicologia, desenvolvimento. Seguindo a metodologia proposta no capítulo anterior, e devido ao reduzido tamanho do time, os membros da equipe possuem atribuições que se entrelaçam em alguns momentos do processo. Os grupos de trabalho que estão colaborando atualmente em cada atividade são:

- **Análise de mercado:**
  Gerentes e pedagogos.
- **Entrevistas com especialistas:**
  Pedagogos e jornalistas.
- **Análise de competidores:**
  Engenheiros de usabilidade e psicólogos.
- **Definição/revisão de requisitos:**
  Toda a equipe.
- **Criação/refinamento de protótipo:**
  Engenheiros de usabilidade, designers gráficos, desenvolvedores.
- **Avaliação de protótipo com usuários:**
  Engenheiros de usabilidade, designers gráficos, desenvolvedores e psicólogos.
- **Implementação de requisitos:**
  Desenvolvedores e designers gráficos.
- **Distribuição do software:**
  Gerentes, pedagogos e jornalistas.

5.2 Atividades iniciais

A atividade inicial de análise de mercado definiu quem serão os futuros usuários do sistema e suas principais necessidades. Através de um levantamento realizado com os professores participantes de um congresso de educação em Recife, a equipe descobriu seu o público potencial, montando uma imagem mental do usuário típico do sistema. Muitas características do usuário foram melhor compreendidas, entre elas: o estilo de vida do professor, qual sua familiaridade com a tecnologia, a maneira como procura manter-se em constante de capacitação e as dificuldades que encontra neste processo. Utilizou-se para tal a técnica conhecida como *Persona* — uma descrição de alguém específica que representa os usuários-alvo do sistema sendo projetado, detalhando suas necessidades, preferências, informações geográficas, biográficas etc.

As entrevistas com professores especialistas na área matemática permitiu à equipe, de fato, compreender o problema educacional da má formação dos professores do ensino básico naquela matéria. Algumas reuniões com os especialistas expuseram essa problemática, o que possibilitou à equipe enxergar a real condição em que se encontram os usuários em potencial do software, sob o prisma de sua necessidade de capacitação. O resultado das entrevistas foi a elaboração de uma seqüência de situações-problema (roteiro) que fazem uso dos mais diversos recursos — como diálogos, animações, narrações e jogos — para representar de forma bastante clara, didática e lúdica a transmissão do conhecimento daqueles especialistas.

A atividade de análise de competidores compreendeu a avaliação de um exaustivo número de aplicações educativas cuja concepção fosse similar àquela do nosso produto, procurando analisar seus pontos fortes e fracos. Para isso, utilizou-se a técnica conhecida por TICESE (acrônimo para Técnica de Inspeção de Conformidade Ergonômica de Software Educacional), que orienta o avaliador para a realização de inspeção de conformidade ergonômica do software. A técnica considerada tantos os aspectos pedagógicos como os aspectos referentes à interface dos sistemas (*e.g.* usabilidade e acessibilidade). Gamez (1998) afirma que "*melhores resultados da aplicação desta técnica serão conseguidos se a mesma for executada por uma equipe multidisciplinar. Sugere-se a presença de um profissional com conhecimentos em ergonomia da interação homem-computador na equipe de avaliação, visto que a técnica tem um forte enfoque sobre as questões de usabilidade de dispositivos interativos*" (p. 3).

5.3 Ciclo de design iterativo

A primeira versão do protótipo destinou-se a avaliar as propriedades de navegabilidade e controle sobre telas e componentes gráficos que o sistema oferecia aos usuários. As tarefas que auxiliaram nessa observação foram: (1) acessar uma aula específica dentro do conteúdo educacional do produto e (2) encontrar a definição de um dado termo no glossário do sistema prototipado. Os testes de usabilidade foram executados com os alunos da disciplina *Interface Usuário-Máquina* — oferecida pelo Centro de Informática da Universidade Federal de Pernambuco — sendo úteis como uma inspeção heurística sobre o protótipo, antes que o mesmo fosse avaliado com usuários reais (*i.e.* os professores), e como testes-piloto, para se corrigirem eventuais ajustes sobre o método de condução dos testes.

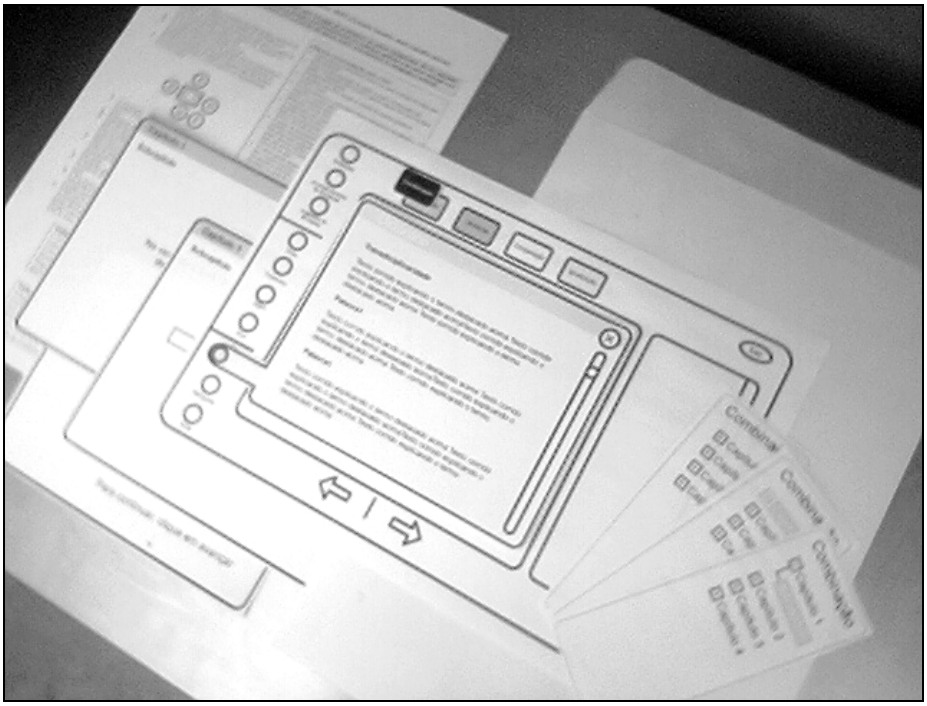

Figura 6.1 — Envelope contendo uma cópia da primeira versão do protótipo.

Os alunos formaram quinze equipes de cinco membros, onde cada um destes desempenhou um papel particular durante testes: um usuário, um facilitador, um 'computador' e dois observadores. Cada equipe recebeu um envelope contendo uma cópia do protótipo (Figura 6.1 acima) e um *script* de testes (Apêndice), que incluiu um passo-a-passo detalhado para cada tarefa e cartões com a descrição destas a serem entregues aos usuários. Os alunos representando os usuários mantiveram-se fora da sala-de-aula (local de testes) enquanto os facilitadores,

observadores e computadores familiarizavam-se com o protótipo por meio de um *walkthrough* interno. Quando os primeiros retornaram ao ambiente de testes, procedeu-se à avaliação dos protótipos pela observação da sua interação com aqueles através da realização das tarefas prescritas. Os principais problemas encontrados pelas equipes foram discutidos, classificados e priorizados por toda a classe em uma lista única. A partir desse levantamento, a equipe de desenvolvimento esteve apta a refinar não apenas o protótipo, mas também o *script* de condução dos testes.

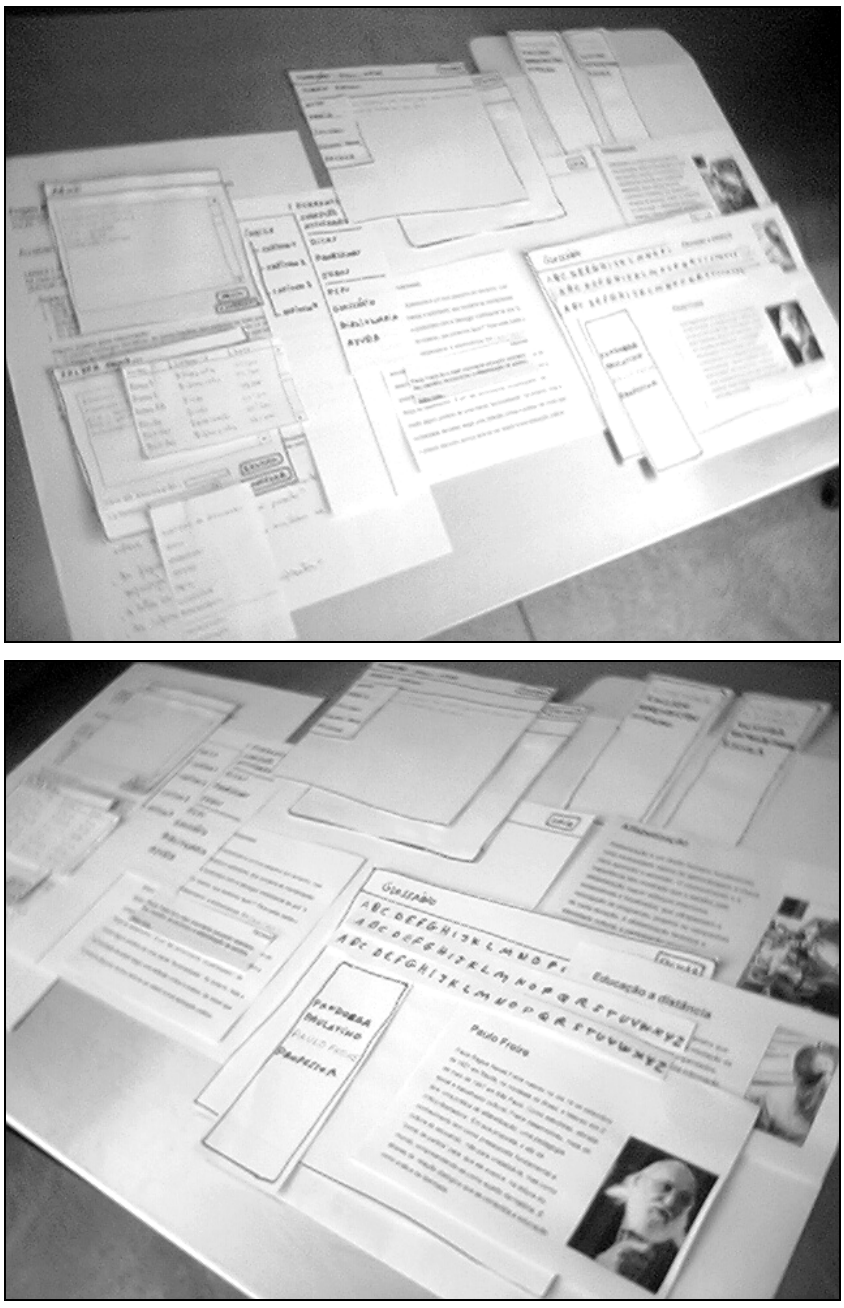

Figura 6.2 — Peças da segunda versão do protótipo.

A nova versão do protótipo (Figura 6.2) solucionou os problemas revelados nos primeiros testes e contemplou outros requisitos que não haviam sido avaliados. As novas tarefas, escolhidas pela equipe de desenvolvimento para avaliar tais requisitos, foram: (1) acessar o glossário do software a partir de uma palavra que ocorre no texto de uma aula, (2) encontrar a definição no glossário de uma palavra que não ocorre no texto de uma aula, (3) escrever uma anotação no sistema e (4) encontrar e reler uma anotação já feita.

Analisando o trabalho de cada grupo na equipe, percebemos que algumas atividades iniciais no processo proposto (análise de mercado, entrevistas com especialistas e análise de competidores) vinham sendo realizadas, de certa forma, em paralelo com a atividade de prototipação rápida. Percebemos que alguns tópicos — *e.g.* adequação da metáfora, navegabilidade e controle etc. — poderiam já ser avaliados com usuários nesta etapa, enquanto outras (principalmente aquelas relacionadas a módulos ou funcionalidades mais específicas do sistema) só poderiam ser eficazmente testadas após uma definição inicial, mesmo que parcial, dos requisitos do sistema. Assim, por motivos didáticos, consideramos ser mais prudente iniciar o ciclo de prototipação após aquelas primeiras atividades.

A nova versão do protótipo foi testada, então, em sessões com cinco professores do ensino fundamental de uma escola pública, no bairro da Várzea, em Recife. Dois observadores da equipe de projeto tomaram notas e indagaram a respeito da interação dos usuários com o protótipo, estando o autor deste trabalho responsável por facilitar as sessões, atuando ainda como 'computador' na manipulação das peças em papel. Cada sessão durou em média vinte minutos, entre as quais ajustes no protótipo eram feitos e avaliados com os usuários seguintes. Os testes revelaram novos problemas, incluindo questões sobre a própria concepção do produto, e seus resultados permitiram novo refinamento do protótipo.

No próximo e último capítulo, concluiremos o presente estudo, discutindo os principais resultados obtidos e descrevendo brevemente as atividades e trabalhos futuros.

CAPÍTULO 6

Conclusão

"Man hath his daily work of body or mind Appointed."

— JOHN MILTON, *Paradise Lost*, 1667.

Por meio do projeto *Mídias Educativas*, que visa o desenvolvimento de uma aplicação multimídia a ser utilizada por professores do ensino fundamental, a metodologia proposta vem sendo utilizada e refinada numa situação real de mercado. A metodologia — que prima pela qualidade tanto do conteúdo pedagógico quanto da usabilidade do produto — tem-se mostrado adequada para conduzir o processo de desenvolvimento de software educativo no projeto *Mídias Educativas*.

A técnica de prototipação rápida no ciclo de design iterativo (*i.e.* concepção, prototipação e avaliação) da metodologia proposta comprovou-se bastante eficiente. Em apenas duas iterações, a avaliação das primeiras versões do protótipo revelou 56 problemas de usabilidade e 24 novos requisitos que o sistema deverá contemplar.

A utilização da prototipação em papel unificou a visão da equipe sobre as principais características do software em desenvolvimento à medida que apontava lacunas de compreensão a respeito dos conceitos envolvidos. Apesar da grande qualidade do *feedback* gerado a partir das avaliações, a prototipação em papel demandou um custo muito baixo, contabilizadas as horas de trabalho do time-núcleo e os recursos materiais empregados.

6.1 Trabalhos futuros

Como próximas atividades no projeto mídias edcativas, continuaremos realizando sessões de avaliação da nova versão do protótipo com usuários reais (professores de matemática) em escolas da Região Metropolitana do Recife. Para avaliar a adequação da nomenclatura dos usados no sistema, utilizaremos a técnica de *card-sorting* com os professores, que consistirá em averiguar como estes relacionam os conceitos presentes no software.

Colhemos evidências de que, em um design centrado no usuário, avaliações de protótipos em um ciclo de design iterativo reduziriam a necessidade de se realizar testes de aceitação de software educativo ao final do processo de desenvolvimento. Entretanto, a demonstração de tal hipótese exige uma metodologia quantitativa que avalie o grau de aceitação do produto já nos primeiros testes com protótipos. Para tanto, questionários de aceitação deverão ser aplicados desde as iterações iniciais até a avaliação do produto final, constatando se há ou não variações consideráveis entre os índices de aceitação coletados junto aos usuários ao longo do processo de desenvolvimento.

Referências bibliográficas

ABRANCHES, S.P. (2003). *Modernidade e formação de professores: a prática dos multiplicadores dos núcleos de tecnologia educacional do Nordeste e a informática na educação.* Tese de doutorado, Faculdade de Educação da Universidade de São Paulo (USP).

ADAM, A. & HALL, R.R. (1992). *Design of visual displays in ticket vending machines.* Livro de resumos da Third International Scientific Conference on Work With Display Units, E24-25, Berlim, Setembro de 1992.

BABIN, P. & KOULOUMDJIAN, M.F. (1989). *Os novos modos de compreender — a geração do audiovisual e do computador.* Tradução de M.C.O. MARQUES, São Paulo: Paulinas.

BAECKER, R.M. & BUXTON, W.A.S. (1987). *Readings in human-computer interaction: a multi-disciplinary approach.* San Mateo CA: Morgan Kaufmann.

BAILEY, G. (1993). *Iterative methodology and designer training in human-computer interface design.* Em Proceedings of ACM INTERCHI'93 Conference, 198-205, Amsterdam, 24 a 29 de Abril de 1993.

BØDKER, S. (1991). *Through the interface — a human activity approach to user interface design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

BØDKER, S., NIELSEN, C. & PETERSEN, M.G. (2000). *Creativity, cooperation and interactive design.* Em Conference Proceedings on Designing Interactive Systems DIS'00: Processes, Practices, Methods and Techniques, 252-261, 17 a 19 de Agosto, Nova Iorque: ACM Press.

BUCHENAU, M. & SURI, J.F. (2000). *Experience prototyping.* Simpósio sobre projeto de sistemas interativos em Conference Proceedings on Designing Interactive Systems: Processes, Practices and Techniques, 424-433, Nova Iorque: ACM Press.

CAMPOS, F., CAMPOS, G. & ROCHA, A..R. (1996). *Dez etapas para o desenvolvimento de software educacional do tipo hipermídia.* Em Anais do III Congresso Ibero-americano de Informática na Educação (RIBIE), Barranquilla, Colômbia.

CASTRO, G.M.C & AGUIAR, T.C. (1999). *Engenharia de software no desenvolvimento de software educacional hipermídia.* XXV Conferência Latinoamericana de Informática, Assunção, Paraguai.

CONSTANTINE, L. & LOCKWOOD, L. (2001). *Process Agility and Software Usability Toward Lightweight Usage-Centered Design.* The Management Forum, Software Development, Vol. 9, N.º 6, Junho de 2001.

COOLEY, M. (1987). *Human-centred systems: an urgent problem for systems designers.* Em AI & Society, vol.1, 37-46.

CUSHMAN, W.H. & ROSENBERG, D.J. (1991). *Human factors in product design.* Amsterdam: Elsevier.

DIMENSTEIN, G. (1997). *Aprendiz do futuro — cidadania hoje e amanhã.* São Paulo: Ática.

EASON, K.D. (1992). *The development of a user-centred design process: a case study in multi- disciplinary research.* Palestra inaugural, 14 de Outubro de 1992, Loughborough UK: Lough-borough University of Technology.

FERREIRA, K.G. (2002). *Teste de usabilidade.* Monografia de conclusão do curso de especialização em Informática, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais.

FURNIVAL, A.C. (1995). *A participação dos usuários no desenvolvimento de sistemas de informação.* Em Ciência da Informação (Artigos), vol. 25, num. 2.

GAMEZ, L. (1998). *TICESE: Técnica de Inspeção de Conformidade Ergonômica de Software Educacional.* Dissertação de mestrado, Universidade de Minho. Guimarães, Portugal.

GOULD, J.D. (1995). *How to design usable systems.* Em R.M. BAECKER, J. GRUDIN, W.A.S. BUXTON & G.S. GREENBERG, Eds. Readings in Human-Computer Interaction, 2.ª ed., 93-121. San Francisco: Morgan Kaufmann Publishers.

GOULD, J.D. & LEWIS, C. (1985). *Designing for usability: key principles and what designers think.* Communications of the ACM, 28, 300-311.

HACKOS, J.T. & REDISH, J.C. (1998). *User and task analysis for interface design.* John Wiley & Sons, Inc. Nova Iorque.

HALL, R.R. (1997). *Ergonomics, design and new technology. The Cumming Memorial Lecture.* Productivity, Ergonomics and Safety, Proceedings of the 33.rd Annual conference of the Ergo-nomics Society of Australia, 53-62, Gold Coast, Queensland, Novembro.

HALL, R.R. (2001). *Prototyping for usability of new technology.* Int. J. Human-Computer Studies 55, 485-501.

HARTSON, H.R. (1998). *Human-computer interaction: interdisciplinary roots and trends.* Extraído do Journal of Systems and Software.

JORDAN, P. (1998). *An introduction to usability.* Londres: Taylor & Francis.

KRUCHTEN, P. (2000). *The rational unified process 2nd edition: an introduction.* Reading, MA: Addison Wesley Longman, Inc.

KUHN, S. & WINOGRAD, T. (1996). *Profile: participatory design.* Em Bringing Design to Software, editado por TERRY WINOGRAD. Nova Iorque: ACM Press.

KUJALA, S., KAUPPINEN, M. & REKOLA, S. (2001). *Bridging the gap between user needs and user requirements.* Em Advances in Human-Computer Interaction I (Proceedings of the Panhellenic Conference with International Participation in Human-Computer Interaction PC-HCI 2001), editado por AVOURIS, N. & FAKOTAKIS, N., Typorama Publications, 45-50.

KUNIAVSKY, M. (2003). *Observing the user experience: a practitioner's guide to user research.* Morgan Kaufmann, São Francisco.

LAGE, F., ZUBENKO, Y. & CATALDI, Z. (2001). *An extended methodology for educational software design: some critical points.* FIE 2001. 31.th ASEE/IEEE Frontiers in Education Conference Paper 1238. Session T2G, 10 a 13 de Outubro de 2001, Reno, Nevada.

LANDAUER, T.K. (1988a). *Relations beetween cognitive psychology and computer systems design.* Em Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, 1-25, editado por CARROLL, J.M.. MIT Press, Cambridge, MA.

LANDAUER, T.K. (1988b). *Research methods in human-computer interaction.* Em Handbook of Human-Computer Interaction, 905-928, editado por HELANDER, M. North-Holland, Amsterdam.

LEONARD, D. & RAYPORT, J.F. (1997). *Spark innovation through empathic design.* Em Harvard Business Review, Novembro e Dezembro, 102-113.

LÈVY, P. (1993). *As tecnologias da inteligência — o futuro do pensamento na era da informática.* Tradução de COSTA, C.I., Rio de Janeiro: Ed. 34.

MANTEI, M. & TEOREY, T.J. (1988). *Cost-benefit analysis for incorporating human factors in the software lifecycle.* Communications of the ACM., vol.31, num.4, 428-439, Abril.

MOOIJ, T. & SMEETS, E. (2001). *Modeling and supporting ICT implementation in secondary schools.* Em Computers & Education, 36, 265 –281.

NASCIMENTO, G. (2001*). O professor e as tecnologias intelectuais: uma parceria que pode dar certo.* Educação e Cibercultura, organizado por L.R.G. ALVES & J.B. SILVA, 51-63, Salvador: Edufba.

NIELSEN, J. (1993). *Usability engineering*. Londres, Reino Unido: Academic Press.

NIELSEN, J. & LANDAUER, T.K. (1993). *A mathematical model of the finding of usability problems.* Em Proceedings of ACM INTERCHI'93 Conference, 206-213, Amsterdam, 24 a 29 de Abril de 1993.

NIELSEN, J. & MACK, R.L. (1994). *Usability inspection methods.* Nova Iorque: John Wiley.

NORMAN, D. (1988). *The psychology of everyday things*. New York: Basic Books.

NORMAN, D. (1990). *Why interfaces don't work*. Em The art of human-computer interface design, editado por LAUREL, B., 209-218, Massachusetts: Addison Wesley Publishing Company, Inc.

NORMAN, D. (1998). *The invisible computer*. Cambridge, MA: MIT Press.

NORMAN, D. & DRAPER, S.W. (1986). *User-centered system design*. Hillsdale NJ: Lawrence Erlbaum.

OLIVEIRA, S.S., GOMES, A.S. & BORGES, H. (2001). *Avaliação de software educativo para o ensino de matemática — o caso das estruturas aditivas.* Em Encontro de Pesquisa Educacional do Nordeste: Educação, Desenvolvimento Humano e Cidadania, São Luís.

PÁDUA, C.I.P.S., TEIXEIRA, A.B.M & RIBEIRO, A.P. (2003). *MAQSEI — uma metodologia de avaliação da qualidade de software educacional infantil.* Em Anais do XIV Simpósio Brasileiro de Informática na Educação, Rio de Janeiro.

PREECE, J., ROGERS, Y., SHARP, H., BENYON, D., HOLLAND, S. & CAREY, T. (1994). *Human-computer interaction.* Workingham, Reino Unido: Addison-Wesley.

ROBERTSON, S. (2001). *Requirements trawling: techniques for discovering requirements.* Em International Journal of Human-Computer Studies, 55 (4), 405-421.

ROSEN, L.D. & WEIL, M. (1995). *Computer availability, computer experience and technophobia among public school teachers.* Em Computers in Human Behavior, 11(1), 09-31.

ROSSON, M.B. & CARROLL, J.M. (2002). *Usability engineering: scenario-based development of human-computer interaction.* São Francisco: Morgan Kaufmann.

RUBIN, J. (1994*). Handbook of usability testing: how to plan, design and conduct effective tests.* Nova Iorque: John Wiley & Sons, Inc.

SAFFO, P. (1996). *The consumer spectrum*. Em Bringing Design to Software, 87-99, editado por WINOGRAD, T. Nova Iorque: ACM Press.

SCHARGE, M. (1999). *Serious play: how the world's best companies simulate to innovate.* Harvard Business School Press, Boston.

SEMINARA, J.L. & GERRIE, J.K. (1966). *Effective mockup utilisation by the industrial design-human factors team.* Em Human Factors, 8, 347-359.

SHACKEL, B. (1959). *A note on panel layout for numbers of identical items.* Em Ergonomics, 2, 247-253.

SHACKEL, B. (1986). *Ergonomics in design for usability*. Em M.D. HARRISON & A.F. MONK, Eds. People & Computers: Designing for Usability, 44-64. Cambridge: Cambridge University Press.

SHNEIDERMAN, B. (1998). *Designing the user interface: strategies for effective human-computer interaction*, 3.ª ed. Reading MA: Addison Wesley.

SINHA, A.K., KLEMMER, S.R. & LANDAY, J.A. (2002). *Embarking on spoken-language NL interface design.* Em The International Journal of Speech Technology, Maio, 5(2), 159-169.

SNYDER, C. (2003). *Paper prototyping: the fast and easy way to design and refine user interfaces.* Morgan Kaufmann Publishers, Londres.

WARD, S. (1994). *Getting feedback from users early in the design process: a case study*. Em N. ADAMS, N. COLEMAN & M. STEVENSON, Eds. Ergonomics: the Fundamental Design Science, Proceedings of the 30.th Annual Conference of the Ergonomics Society of Australia, 22-29, 4 a 7 de Dezembro. Canberra: The Ergonomics Society of Australia.

ZHAO, Y. & CZIKO, G.A. (2001). *Teacher adoption of technology: perceptual control theory perspective.* Em Journal of Technology and Teacher Education, 9(1), 05-30.