

O desenvolvedor-designer e o design da interação*

Gabriela Trindade Perry

ABSTRACT

The central character of this article is the developer-designer, a person who has responsibility both on the implementation of the application and on interaction design. The point of this paper is that, even if these “devigners” have formal education and experience in interaction design, the fact of being responsible for the computational aspects of the application is enough to declare them unable to carry out activities of interaction design. The reason is that, to develop the computational part of an application, it is necessary to use a mental framework that focuses on the needs of the system, not on the user’s and on the task’s needs. Following this line of reasoning, it is recommended that projects should have an interaction designer.

RESUMO

O personagem central deste artigo é o desenvolvedor-designer, uma pessoa que tem responsabilidade tanto pela implementação da aplicação quanto pelo design da interação. O ponto deste artigo é que mesmo se estes “devigners” têm educação formal e experiência em design da interação, o fato de serem responsáveis pelo lado computacional da aplicação é suficiente para declará-los incapazes de conduzir atividades de design de interação. A razão é que, para desenvolver o lado computacional de uma aplicação, é necessário usar um framework mental que focaliza as necessidades do sistema, não as da tarefa ou do usuário. Seguindo esta linha de raciocínio, recomenda-se que os projetos devem ter um designer de interação.

Author Keywords

Devigner, interaction design, mental model.

1. INTRODUÇÃO

O personagem central deste artigo é a figura do desenvolvedor-designer, uma pessoa que tem responsabilidade tanto sobre a implementação da aplicação como sobre o design da interação. Argumenta-se que, mesmo que tenham tido educação formal e experiência em design de interação, o fato de serem responsáveis pelo lado computacional da aplicação (por exemplo: modelagem e análise, codificação e testes) é suficiente para declará-las impossibilitadas de exercer as atividades de especificação de interação. O motivo é que, para desenvolver a parte computacional de uma aplicação, utiliza-se um framework

mental que focaliza as necessidades do sistema, e não as do usuário e da tarefa.

Contudo, antes de expor a argumentação, será feita uma breve ilustração do cenário de desenvolvimento de software e design de interação. Em seguida, será apresentado o sujeito deste artigo: o desenvolvedor-designer (“*devigner*”).

2. O PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Até o início dos anos 80, quando popularizaram-se os estudos em interface homem-computador [6], os profissionais das ciências da computação eram os únicos envolvidos nos processos de desenvolvimento de software. Até porque, antes desta data, havia poucos exemplos de aplicações que tivessem interfaces visuais. As primeiras tentativas de organizar o ciclo de vida destes produtos é anterior à década de 80, vide o modelo Waterfall, proposto em 1970 [13]. O modelo Espiral [2], que inspira as metodologias baseadas em prototipagem rápida de hoje em dia (por exemplo: XP¹ e RUP²), é de 1988.

O aumento das capacidades dos computadores pessoais acabou por popularizá-los, dando forças à uma explosão de pesquisas sobre interação homem-computador (doravante IHC), com trabalhos em áreas como: manipulação direta [14] utilização de metáforas [8] e modelagem da performance do usuário [3], por exemplo. Desta forma, mobilizados por uma grande demanda por softwares que apoiassem tarefas de forma efetiva e sem causar constrangimentos ao usuário, um novo grupo de profissionais começou a fazer parte do cenário do desenvolvimento. Dele fazem parte os responsáveis por inserir e implementar práticas de IHC durante o ciclo de vida, sendo uma delas o design de interação.

Nas seções seguintes, serão apresentados alguns conceitos-chave utilizados neste artigo: design de interação, designers-desenvolvedores e desenvolvedores-designers..

*Este artigo está publicado apenas no blog.

¹ eXtreme Programming [1].

² Rational Unified Process [9].

3. DESIGN DE INTERAÇÃO

Para apresentar o conceito de design de interação serão utilizadas definições apresentadas por dois autores da área de IHC, [5] e [7]. No primeiro, encontra-se a seguinte definição de interação:

Por interação nos referimos a qualquer comunicação entre o usuário e o computador, seja direta ou indireta. Interação direta envolve um diálogo com feedback e controle durante a realização de uma tarefa. Interação indireta pode envolver processamento de fundo ou em lotes. O importante é que o usuário está interagindo com o computador com algum objetivo.

O designer de interação é responsável por desenvolver o conteúdo, comportamento e aparência do design de interação [7]. Pessoas neste papel são diretamente responsáveis por assegurar usabilidade, incluindo performance e satisfação do usuário. Eles estão preocupados com elementos críticos de design como funcionalidade, sequenciamento, conteúdo e acesso à informação, bem como detalhes como aparência de menus, formatação de formulários, e como assegurar consistência através da interface. Uma grande parte do trabalho designer está relacionada com definir indicadores de usabilidade mensuráveis, avaliar designs de interação com usuários e fazer redesigns baseados na avaliação dos usuários.

As atividades do designer de interação transcendem a aparência da interface e definição de guias de estilo, pois é dele a responsabilidade de definir como a aplicação se comunica com o usuário [7]. Isso implica não apenas em escolher componentes de interação de acordo com cada ocasião - por exemplo: menus, listas, campos de texto, alertas - ou o tipo da interface - por exemplo: se é modal ou não, se tem janelas ou tabs - mas definir a forma como estes componentes orientam a ação dos usuários através da aplicação. Um designer de interação pode, por exemplo, utilizar as heurísticas como as propostas por Norman [12] para guiar o usuário, fazendo com que: i) a memória de trabalho não seja sobrecarregada, disponibilizando informações de forma eficiente; ii) simplificando as estruturas da tarefa; iii) fazendo as coisas visíveis; iv) mapeando os dados com a representação de forma correta; v) explorando as restrições; vi) projetar prevendo erros e vii) se tudo o mais não se aplicar, padronizar.

Com esta exposição, procura-se evidenciar que a atividade de design de interação está essencialmente ligada à compreender o usuário realizando uma tarefa, e que demanda uma natureza de conhecimento diferente à computacional, de perfil mais ligado à lógica. Também se deseja mostrar que esta atividade transcende o caráter estético/simbólico, que apenas se preocupa com a aparência das interfaces. Na próxima seção será apresentado o personagem-central deste artigo, o desenvolvedor-designer.

3. DEVIGNER: O DESENVOLVEDOR-DESIGNER

Conforme visto na seção 2, os desenvolvedores estão há mais tempo envolvidos com a produção de software que os designers de interação. Também pode-se constatar, através da análise da evolução dos métodos de controle do ciclo de vida (como os citados na seção 2), que o grau de especialização foi aumentado com o passar do tempo, criando novas atividades, tais como: analistas; responsáveis pela arquitetura da aplicação, programadores; responsáveis pela implementação da arquitetura em uma determinada linguagem e testadores; responsáveis por escrever testes automatizados para cada uma das partes da aplicação. Sendo assim, acredita-se que não seja surpresa o fato de um desenvolvedor acumular a função de designer de interação, mesmo que esta atividade não esteja especificada no modelo que está sendo seguido. Este profissional é chamado de devigner [15], uma mistura de developer e designer. Este termo foi criado por membros de comunidades de desenvolvedores de RIA - *Rich Internet Applications* - que detectaram a falta de designers visuais capacitados a se integrar ao fluxo de trabalho de projeto de interfaces.

Na próxima seção serão apresentados os argumentos que levam à conclusão da necessidade de haver um responsável pelo design da interação que não esteja também responsável por tarefas de caráter computacional de um projeto.

4. PORQUE UM DESENVOLVEDOR NÃO DEVE ACUMULAR A TAREFA DE DESIGN DE INTERAÇÃO EM UM MESMO PROJETO

Em princípio, cita-se Theodor Nelson [12], que sustenta que desenvolvedores não estão capacitados para fazer design de interação. Segundo ele,

Aprender a programar tem tanto a ver com design quanto aprender a digitar tem a ver com escrever poesias. O design da interatividade é raramente ensinado em escolas de programação. O que precisamos em software é o que as pessoas aprendem em escolas de cinema, ao menos até onde isso possa ser ensinado.

Em relação à afirmação de Nelson de que “o design da interatividade é raramente ensinado em escolas de programação”, cita-se que a ACM³ e IEEE⁴ recomendaram que o ensino de IHC fosse introduzido nos currículos dos cursos superiores de computação apenas no final dos anos oitenta [4]. Ressalta-se que, neste artigo, não se pretende afirmar que um profissional da área da computação não esteja preparado para desempenhar a atividade de designer de interação, apenas que não deve acumulá-la com atividades computacionais em um mesmo projeto.

Seguindo a linha de que as atividades de design de interação e desenvolvimento computacional devem ser feitas por diferentes profissionais, [10] faz uma analogia

³ Association for Computer Machinery.

⁴ Institute of Electrical and Electronics Engineers

com arquitetos e engenheiros. Diz o autor que, se as casas fossem feitas apenas por engenheiros, elas seriam certamente mais fáceis de construir, mas não necessariamente melhor para se viver. Como o designer, o arquiteto, além de ser preparado para perceber o que funciona ou não, tem um papel de advogado do cliente (no caso do designer, o usuário).

A idéia de modelo conceitual se constitui em um framework que pode ser utilizado para compreender o motivo desta incompatibilidade [12]. Nele estão representados os modelos que o designer e o usuário têm do sistema (imagem do sistema). Se estes modelos forem

diferentes, o resultado é que, para o usuário, o sistema parecerá inconsistente e difícil de usar. Norman propõe a metodologia Design Centrado no Usuário como resposta à esta questão. Seguindo esta linha, assume-se, que o designer (de interação, no caso deste artigo) é o profissional responsável por projetar o sistema considerando as necessidades do usuário. No entanto, para descrever de forma mais precisa o cenário do desenvolvimento, é preciso considerar os indivíduos responsáveis pela implementação do sistema, chamados aqui, de forma genérica, de desenvolvedores. A forma como estas relações se dão, de acordo com a visão deste artigo, está ilustrada na figura 1.

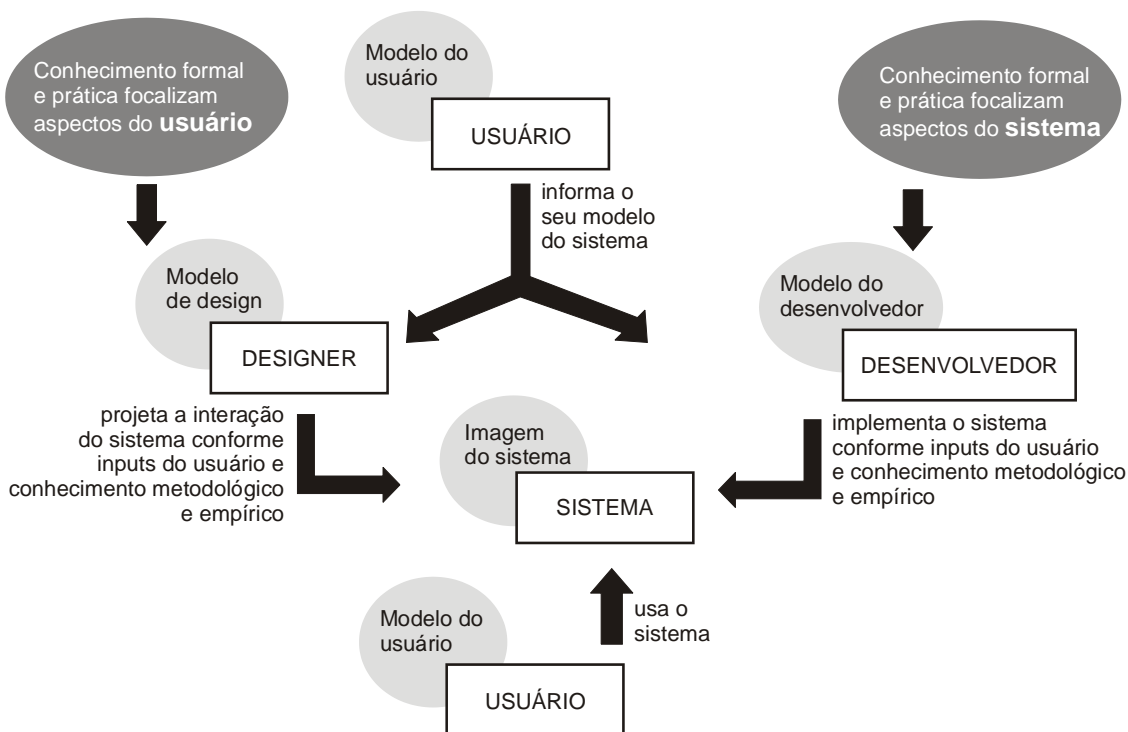


Figura 1. Relações simplificadas entre usuário, designer, desenvolvedor e sistema em um cenário de desenvolvimento de software.

Na figura 1, percebe-se que o usuário informa tanto o desenvolvedor quanto o designer sobre seu modelo do sistema. De fato, ambos profissionais são capacitados, durante o curso de graduação, a ouvi-lo e extrair destas entrevistas um conjunto de requisitos para a aplicação.

As diferenças estão na forma como estes encontros com o usuário acontecem, na forma que esta informação é utilizada e no resultado deste tratamento. Enquanto o designer procura ouvir o usuário e compreender as necessidades que a aplicação deve satisfazer (através de métodos como análise da tarefa e entrevistas, por exemplo), o desenvolvedor procura ouvir o usuário para extrair o conjunto de requisitos da aplicação, o que permite que seja executado o ponto de entrada em sua metodologia de desenvolvimento: a geração de casos de uso. O designer e o desenvolvedor, à medida que projetam, são informados por

conhecimentos formais e por saberes empíricos, que podem misturar experiências práticas com orientações individuais. A diferença fundamental - e este é o argumento central deste artigo - é que o modelo do designer, por princípio, é centrado no usuário, enquanto o do desenvolvedor é centrado no sistema. Assim, enquanto o designer projeta, ele tem em mente o usuário, enquanto o desenvolvedor tem o sistema.

Por este motivo, mesmo que um desenvolvedor acumule a função de designer de interação, ele estará utilizando, em função das necessidades das atividades relacionadas à implementação, um modelo centrado no sistema. Mesmo que este desenvolvedor seja uma pessoa educada formalmente e com saber empírico consolidado em design de interação, ele não consegue trocar de modelo a todo tempo. À medida que o desenvolvimento da aplicação

avança, em alguns momentos as necessidades do sistema consomem as atenções, seja por críticas para o desenrolar do ciclo, seja por difíceis e delicadas. Nestes momentos, o desenvolvedor não consegue se portar como se tivesse “dupla personalidade”, ora utilizando um “modelo de designer”, ora um “modelo de desenvolvedor”. Após estes momentos passarem, e as questões terem sido resolvidas, as necessidades do usuário foram deixadas em segundo plano, e retomá-las neste ponto pode ocasionar um nova entrada em ciclos de turbulência no desenvolvimento. Uma outra possibilidade é que estes momentos de atenção extremamente focada nas necessidades do sistema se sucedam durante uma boa parte do ciclo, sem períodos de calmarias – quando, hipoteticamente, o desenvolvedor poderia usar seu “modelo de designer”.

Para ilustrar estas idéias com exemplos, a seção seguinte traz o caminho percorrido por um devigner para implementar um caso de uso em uma aplicação educacional, e uma pergunta feita em uma comunidade de desenvolvimento de *Rich Internet Applications*.

5. DEVIGNERS EM AÇÃO

Esta seção inicia com uma mensagem retirada de uma lista de discussão. Nela, o desenvolvedor tem questões que pertencem ao domínio do design de interação, mas trata-as como uma questão do sistema.

seguinte galera, preciso criar uma aplicacaum gigante em flex... estou a milhoes de anos programando em delphi e agora estou engatinhando em flex
alguem poderia me dar uma maum?

minha ideia eh fazer um menu principal q crie dinamicamente paineis com um moduleloader dentro q ajam como se fossem janelas com minimizar maximizar mover dimensionar precisaria tb.. q qdo fosse clicado no menu para chamar essas janelas, verificasse se elas jah existem.

existe alguma coisa pronta pra isso? essa eh a melhor forma?

como vcz fariam pra fazer uma aplicacaum com uns 50 cadastros e uns 800 relatorios?

se possivel mandem exemplos... fontes... q naum consegui nem criar dinamicamente os paineis

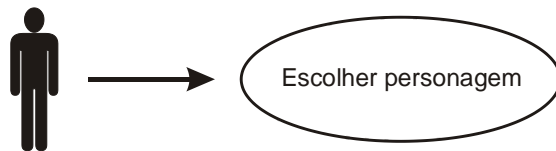
Figura 2. Pergunta feita à lista flexdev. Em

http://groups.google.com.br/group/flexdev/browse_thread/thread/134475aacc2010547/b8aea1bff8865830

O usuário desta lista demonstra estar utilizando um modelo de desenvolvedor ao fazer estes questionamentos. Ele se preocupa com a implementação, descrevendo os componentes que está pensando em utilizar, informa o tamanho do sistema (esta é uma variável importante, que pode ser um fator de decisão entre diferentes abordagens à implementação), pergunta se é a melhor forma, pede

exemplos... Em listas de discussão este tipo de mensagem não chega a ser incomum.

O próximo e último exemplo mostra o caminho percorrido para implementar um caso de uso. Esta implementação foi feita pela autora do artigo. A figura 3 ilustra o caso de uso e seu respectivo cenário.



Cenário:

O usuário deve escolher um personagem, um desenho que será associado a um nome, uma atividade (por exemplo, prefeito) e a uma fala.

Quando a aplicação for lançada, haverá cerca de 100 desenhos, além dos que os usuários poderão enviar, a fim de compor uma coleção de desenhos.

Para facilitar a tarefa de escolha, deve ser disponibilizado um filtro.

Figura 3. Estudo de caso e cenário associado.

Como o caso foi vivenciado pela autora deste artigo, pode-se mostrar a sequência de raciocínio e ações que foi, mais ou menos, a seguinte:

1. Como mostrar uma grande quantidade de desenhos dispostos em linhas e colunas?
2. Como fazer com que os desenhos continuem sendo mostrados em linhas e colunas à medida que as escolhas no filtro são feitas? Os desenhos podem ter suas posições recalculadas dinamicamente? Ou eles ficarão invisíveis se não passarem pela filtragem (nesse caso a interface ficaria com “buracos”)?
3. Algum componente faz isso? Existe um exemplo?
4. Teste de implementação do componente. Verificação de propriedades, métodos e eventos. Estudo da documentação. Estudo de mais exemplos.
5. Como fazer o filtro? Como relacionar o filtro ao componente?
6. Estudo da documentação.
7. Teste de implementação da solução para filtragem e apresentação dos desenhos em linhas e colunas.
8. Modelagem.
9. Implementação.
10. Testes.
11. Percepção de erro de concepção da interação.

Como se pode perceber, os esforços foram todos direcionados à resolução de questões e necessidades da implementação. Apenas depois que estes foram atendidos é que se perceberam erros na concepção da interação. Para

ilustrar este caso, a figura 4 traz a primeira interface gerada (passo 9), a figura 5 uma tentativa de solucionar o problema

(passo 11) e a figura 6 mostra uma consequência da nova solução adotada.

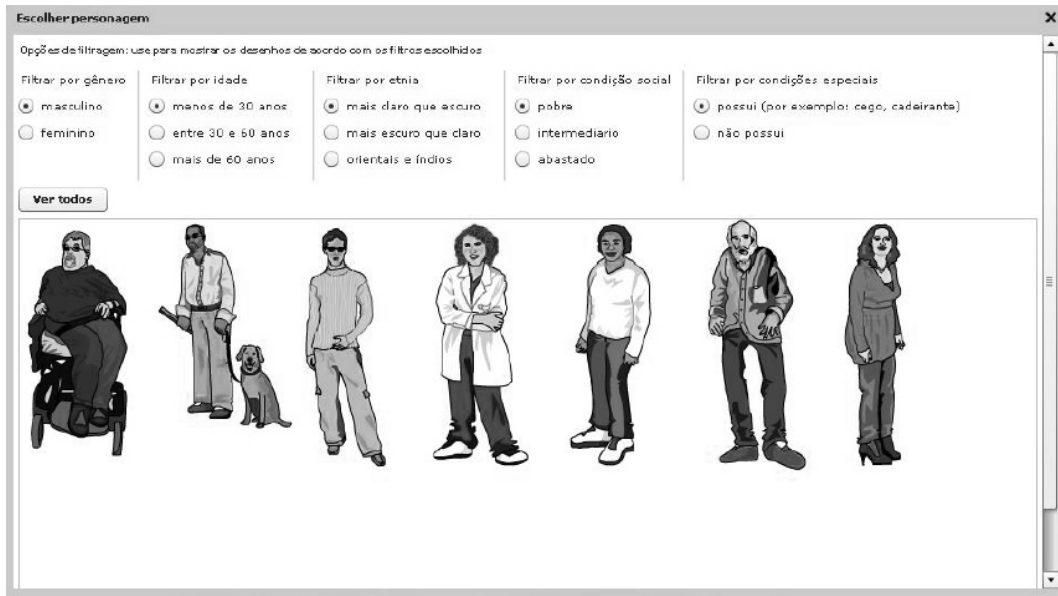


Figura 4. Primeira solução.

Esta interface não permitia que o usuário escolhesse mais de uma alternativa por vez (por exemplo, ver homens e mulheres com mais de 60 anos).

Esta é uma restrição do componente utilizado (conhecido como botão de rádio). Este componente foi escolhido sem

que fosse feita uma reflexão sobre sua adequação para este caso, porque a principal preocupação era mostrar os desenhos dispostos em linhas e colunas e o funcionamento do filtro.

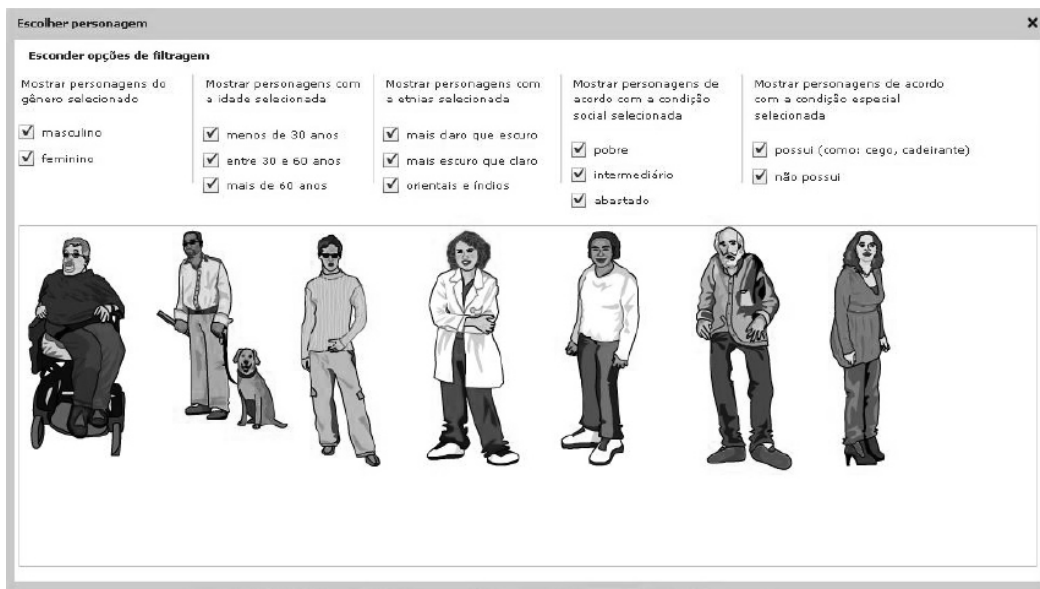


Figura 5. Segunda solução (após perceber erro de projeto de interação).

Nesta solução os botões de rádio foram substituídos por checkboxes, que permitem mais de uma escolha. A questão

passou a ser como a seleção dos checkboxes afetaria a visão da interface. A primeira opção era fazer os itens estarem

todos marcados no início, e a ação de desmarcar removeria os desenhos que não se encaixam na seleção. A segunda era trazer os itens desmarcados e, à medida que fossem marcados, esconderiam os que não deveriam ser mostrados. Esta escolha leva em consideração se é mais natural para o

usuário escolher o que ele quer ver ou o que ele não quer ver.

De qualquer forma, neste ponto, o desenvolvedor estava usando seu “modelo de designer”.

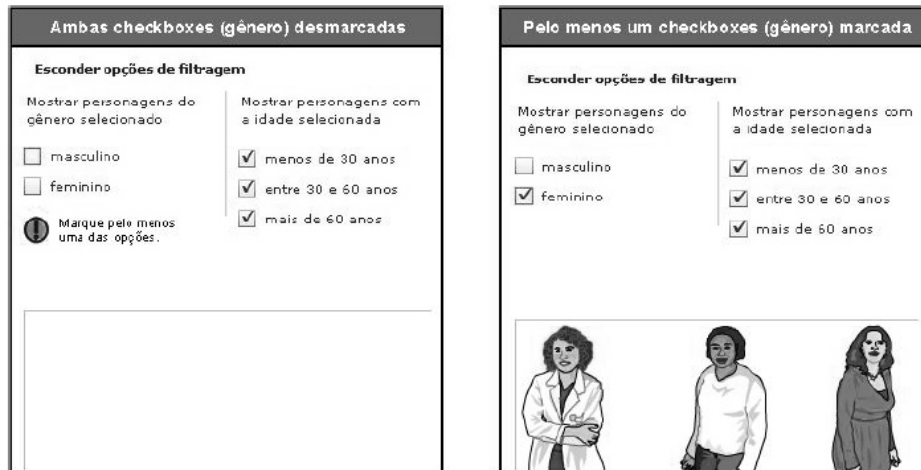


Figura 6. Consequência da nova solução.

Este comportamento da interface segue uma das heurísticas propostas por Norman [12] projetar para o erro. Caso o usuário desmarcasse todas as opções de qualquer um dos filtros, não haveria desenhos para serem mostrados. Nesse caso ainda havia escolhas a fazer. A primeira era colocar um aviso, como ilustrado na figura. A segunda solução seria impedir que todos os itens fossem desmarcados. Provavelmente esta solução não foi adotada porque, além de interferir na liberdade de ação do usuário e ter potencial de provocar confusão (por exemplo: “porque essa alternativa não pode ser desmarcada e as outras podem?”), é mais difícil de implementar que a primeira. Nesse caso, o desenvolvedor abandona o “modelo de designer” e volta a fazer escolhas informado pelo “modelo do desenvolvedor”.

6. CONCLUSÃO

A conclusão que se faz a partir destes dados, é que, em todos os projetos, deve haver uma pessoa responsável apenas pela especificação do design de interação. O motivo é que, para realizar esta tarefa, é preciso utilizar um modelo de pensamento que priorize as necessidades e características do usuário e da tarefa. Se, como visto nos exemplos da seção anterior, o profissional acumula esta atividade com algum aspecto da implementação do sistema, o resultado é que o modelo de raciocínio utilizado para tomar decisões que prevalece é o do desenvolvedor.

Isto pode se dar por vários motivos, entre os quais cita-se: as necessidades de implementação são mais urgentes por representarem um empecilho direto à geração de protótipos funcionais (um requisito de metodologias ágeis, como XP).

Também se pode especular que os desenvolvedores, por causa das características de sua formação sintam-se mais à vontade com atividades que possam ser projetadas, analisadas, construídas e testadas, ou seja, que tenham um caráter mais objetivo. As atividades relacionadas ao design de interação, por sua vez, mesmo que possam ser modeladas e testadas, pressupõe um conhecimento do usuário que está impregnado de aspectos subjetivos. Além disso, não se testa todas as possibilidades de interações: deve-se decidir entre algumas delas com base nos conhecimentos formais e empíricos. Isto pode ser uma característica da atividade que pode ser pouco atraente para os desenvolvedores.

De toda forma, parece não haver impedimentos para o desenvolvedor que deseje executar as especificações de interação. Apenas recomenda-se que ele não o faça se estiver responsável por atividades de implementação em um determinado projeto. Nesse caso, a colaboração entre os profissionais pode-se dar através de projeção em duplas, para que os dois portadores de modelos tão importantes para o sucesso de uma aplicação – os modelos de desenvolvedor e de designer – possam contribuir da melhor forma possível, sem negligenciar um aspecto em detrimento de outro.

7. REFERÊNCIAS

1. Beck, K. et al. (2001). Manifesto for agile software development. <http://agilemanifesto.org>
2. Boehm, B. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21, 5, (1988), 61-72.

3. Card, S. K.; Moran, T. P.; Newell, A. The Keystroke-Level Model for User Performance Time With Interactive Systems. *Communications of the ACM*, 23, 7, (1980), 396-410.
4. Denning, P., D. Comer, D. Gries, M. Mulder, A. Tucker, A. Turner, P. Young. Computing as a Discipline. *Communications of the ACM*, 32, 1, (1989), 9-23.
5. Dix, A., J. Finlay, G. Abowd, R. Beale. *Human-Computer Interaction*, segunda edição. New York: Prentice Hall, 1998.
6. Hendrick, H W. *Macroergonomics : an introduction to work system design*. Santa Monica, Calif.: Human Factors and Ergonomics Society, 2001.
7. Hix, D., H. R. Hartson. *Developing User Interfaces : ensuring usability through product e process*. Nova Iorque: John Wiley, 1993.
8. Kay, A. *Interface: A personal view*. In B. Laurel, B. (Ed). *The Art of Human-Computer Interface Design*. Addison-Wesley Publishing Company, 191-207, 1990.
9. Kruchten, P. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 1990.
10. Marion, C. (1999) What is Interaction Design and What Does It Mean to Information Designers? <http://mysite.verizon.net/resnx4g7/PCD/WhatIsInteractionDesign.html>.
11. Nelson, T. *The Right Way to Think About Software Design*. In B. Laurel, B. (Ed). *The Art of Human-Computer Interface Design*. Addison-Wesley Publishing Company, 191-207, 1990.
12. Norman, D. *The psychology of everyday things*. Nova Iorque : Basic Books, 1988.
13. Royce, W. Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON*, 1, 8, (1970), 328-338.
14. Schneiderman, B. *Designing the User Interface: Strategies for Effective Human Computer Interaction*. Addison-Wesley, 1980.
15. Stewart, R. (2006). One Reason Why the "Devigner" Workflow is Important - Designer Shortage. <http://blog.digitalbackcountry.com/?p=593>