Modelagem de interfaces ricas cientes de contexto na Web 2.0 para plataforma *Android*

Andre L. Bonfatti
Univ. Federal de São Carlos
Rdv João Leme dos Santos,
Km 110 - CEP 18052-780 –
Sorocaba – SP – Brasil
andrebnf@gmail.com

Diego H. Quintale
Univ. Federal de São Carlos
Rdv João Leme dos Santos,
Km 110 - CEP 18052-780 –
Sorocaba – SP – Brasil
diegohquintale@gmail.com

Luciana A. M. Zaina
Univ. Federal de São Carlos
Rdv João Leme dos Santos,
Km 110 - CEP 18052-780 –
Sorocaba – SP – Brasil
lzaina@ufscar.br

Fábio L. Verdi
Univ. Federal de São Carlos
Rdv João Leme dos Santos,
Km 110 - CEP 18052-780 –
Sorocaba – SP – Brasil
verdi@ufscar.br

RESUMO

A Computação Ubíqua tem trazido novos desafios para a Engenharia de Software. Um desses desafios tem sido o desenvolvimento para diversos dispositivos móveis. Dentro desse escopo a plataforma *Android* tem sido alvo de diferentes experiências exploratórias. Esse artigo apresenta uma ferramenta que permite modelar interfaces do usuário para páginas da web 2.0 para plataforma *Android* que sejam adaptáveis de acordo com os dispositivos e as preferências do usuário. Uma aplicação de *upload* e *download* de arquivos foi desenvolvida e testada em diferentes dispositivos por meio de simuladores.

Categorias

D.2.2 [**Design Tools and Techniques**]: user interfaces

Termos Gerais

Design

Palavras-chave

Adaptação de interfaces, *Android*, interface rica, Web 2.0.

1. INTRODUÇÃO

A área da Computação Ubíqua tem motivado a área de Engenharia de Software a produzir recursos que dêem suporte a diversos ramos da produção e manutenção de software. Um dos aspectos críticos é como desenvolver aplicações que se adéquem às mais díspares capacidades dos dispositivos existentes, as denominadas aplicações ubíquas[17].

Workshop de Trabalhos de Iniciação Cientifica- Webmedia 2012, 15 a 18 de Outubro, 2012, São Paulo, São Paulo, Brazil.

Um dos aspectos importantes dentro do desenvolvimento de aplicações ubíquas é a ciência de contexto. Nas aplicações ubíquas um contexto pode representar as restrições de um ambiente, já que muitos dispositivos podem ter poucos recursos disponíveis. Outro fator de contexto está relacionado ao perfil do usuário que acessa a aplicação, sua localização geográfica, a língua mais adequada para ele, entre outros. A ciência de contexto também está relacionada ao desenvolvimento de interface com o usuário. Um dos aspectos críticos é como desenvolver aplicações que sejam adequadas as diferentes capacidades dos dispositivos existentes e que ao mesmo tempo possam atender as preferências do usuário. Essa necessidade de conhecer os elementos que envolvem a interação e adaptar a aplicação segundo esses elementos é denominada de ciência de contexto. De maneira genérica, pode-se definir contexto como um conjunto de condições relevantes e influências que possibilitam a compreensão de uma situação[12, 16].

O objetivo deste artigo é apresentar uma ferramenta denominada *DroidRichWeb*, cujo objetivo é modelar interfaces ricas na Web para a plataforma Android, que sejam adaptáveis de acordo com o contexto da interação do usuário. Para validar a proposta foi desenvolvida uma aplicação de *upload* e *download* de arquivos, onde a interface dessa aplicação foi modelada através do *DroidRichWeb*. Foram utilizados simuladores de diferentes dispositivos para validação da aplicação e da adaptação. O restante do artigo está estruturado da seguinte forma: a seção 2 apresenta sucintamente uma discussão sobre adaptação em dispositivos móveis e um estudo de tecnologias para interface de desenvolvimento Web e de trabalhos relacionados. A ferramenta proposta é apresentada na seção 3. A seção 4 descreve a validação da proposta e a seção 5 realiza as considerações finais.

2. FUNDAMENTOS E TECNOLOGIAS

2.1 Conceitos

Há alguns anos são exploradas metodologias que provêm mecanismos de desenvolvimento de interfaces adaptáveis dependendo do usuário e seu dispositivo. Com o crescimento do uso de dispositivos móveis esta demanda tem sido cada

vez maior. O desenvolvimento de interfaces para dispositivos móveis, no entanto, apresenta muitos desafios [16]: utilização racional do espaço de tela, mecanismos de interação e desenvolvimento de interfaces genéricas. A necessidade de criar diferentes versões de interfaces que sejam aderentes aos diversos tipos de dispositivos móveis, acarreta um tempo de dedicação maior ao desenvolvimento. O mais adequado é que parte da interface seja projetada durante o processo de desenvolvimento e outra parte possa ser construída dinamicamente em tempo de execução da interface[9].

Estudos demonstram que a satisfação do usuário na utilização de interfaces em dispositivos com tela de tamanho pequeno está diretamente relacionada a capacidade de adaptação da interface. A adaptação deve procurar exibir uma interface que seja mais precisa considerando as limitações de tamanho da tela [14].

2.2 Tecnologias

Além do desenvolvimento de aplicações web na plataforma *Android*, outras tecnologias foram estudadas durante este trabalho. O navegador padrão do *Android* 2.2[15] foi testado a fim de indicar quais funcionalidades da Web 2.0 era capaz de executar. Ao final dos testes foi concluído que as *tags* de organização do HTML5[3] (como `header`, `nav` etc) são interpretadas corretamente além da maioria dos eventos e efeitos implementados pela biblioteca *Javascript JQuery*[6]. Com base nesse teste a ferramenta foi implementada dando as opções de inclusão, modificação e exclusão de elementos básicos HTML (imagem, link etc) além de outros recursos como efeitos implementados pelo *JQuery* e organização do HTML5. Também foi estudada a propriedade `pattern` do HTML 5. A sua função é bloquear o botão de submissão de um formulário quando algum campo de texto não está no formato exigido pelo desenvolvedor. Essa propriedade, porém, não é interpretada pelo navegador nativo do *Android* e, portanto, a validação de campos de texto foi feita utilizando a biblioteca *JQuery*. O *JQuery* é usado adicionando-se a biblioteca ao código HTML e fazendo a chamada de suas funções. Como a ferramenta utiliza-se de arquivo XML[2] foi adotada a biblioteca JDOM[4] para facilitar sua manipulação. Outro teste foi realizado e este procurava validar algumas funcionalidades CSS3[1] no navegador padrão do *Android*. Foram testadas as propriedades: `face-font`, `nth-child`, `rgba`, `text-shadow`, `border-radius`, `border-image`, `stroke`, `gradient`, `column`, `transformation`; bem como as funcionalidades de animação e caixas (`box`) flexíveis. Todos os casos de teste foram satisfatórios. Quanto à adaptabilidade, algumas técnicas de modelagem para formulários e campos de texto[20] foram estudadas a fim de que estes objetos fossem corretamente estruturados e, após sua adaptação, o resultado fosse satisfatório.

2.3 Trabalhos Relacionados

Alguns trabalhos já desenvolvidos dentro do escopo de desenvolvimento de interfaces adaptáveis foram estudados. O ambiente *XMobile* permite a geração de aplicações adaptativas baseadas em formulários para dispositivos móveis[19]. Outra ferramenta, a *Semantic Transformer*, permite realizar a transformação automática de páginas *Web* desenvolvidas para a plataforma *desktop* em páginas Web adequadas para dispositivos móveis[18]. Algumas outras ferramentas foram consultadas, como a *qooxdoo*[7], *Jo HTML5 Mobile App Framework*[5], entre outras; porém a maioria delas têm foco no desenvolvimento de interfaces que se adaptam tanto para ambientes *desktop* quanto para *mobile* e isso faz com que a adaptabilidade em função do perfil do usuário e/ou do dispositivo não seja tão eficaz. Uma ferramenta explorada foi a *DroidDraw*[13], que permite a modelagem de interfaces para aplicações para plataforma *Android*.

Billsus et al [10] reporta um trabalho que realiza a apresentação de interfaces adaptáveis para acesso ubíquo por meio da web. A adaptação da interface é baseada na aprendizagem das escolhas do usuário. Cada decisão que o usuário toma é essencial para a exibição das informações. São realizados testes com exibição de menus o conteúdo é adaptado, mantendo as informações das quais o usuário demonstra mais interesse acima das demais. Os exemplos realizados foram específicos para alguns domínios de aplicação. Além disso, a ferramenta não utiliza informações do dispositivo como base para as adaptações.

3. FERRAMENTA DROIDRICHWEB

O desenvolvimento do *DroidRichWeb* foi baseado na ferramenta *DroidDraw*[13], tanto na interface gráfica quanto na arquitetura usada. A *DroidRichWeb* permite que o desenvolvedor crie modelos de interfaces ricas, gerando modelos genéricos de interfaces. A partir da definição dos modelos genéricos o desenvolvedor irá especificar quais pontos da interface possuem variabilidades definindo quais critérios serão considerados para a adaptação daquele ponto. A ferramenta gera a estrutura da interface do usuário no formato XML (*Extensible Markup Language*)[2]. Também cria um arquivo XML com regras de adaptação das interfaces ricas que será utilizado para a adaptação da interface em tempo de execução da aplicação. São considerados dois aspectos contextuais: o perfil do dispositivo e o perfil do usuário. Durante a modelagem é possível selecionar que características relativas ao perfil dispositivo e ao perfil do usuário possuem impacto na adaptação da interface.

3.1 Arquitetura e Funcionalidades

Na Figura 1 são apresentados dois diagramas de classes, contendo classes centrais da ferramenta. A classe `Component` é a especialização da classe `Element` do JDOM[4] (representando cada componente HTML), já a classe `PropertyCapability` estende a classe `Attribute` do JDOM (representando os atributos de uma tag).

Na Figura 1 (a) podem ser observadas as classes que representam os componentes da ferramenta: a classe `Component` é a superclasse de todos os componentes. A classe `Generic` representa os componentes genéricos, que são especificados na inicialização do objeto. Já as classes `Html`, `Img`, `Link`, `Text`, `Form` e `Input` representam os componentes mais elaborados que precisam de métodos especiais para sua manipulação. Já na Figura 1 (b) é apresentada a manipulação do componente `Adaptation` pela ferramenta. As classes `Profile` e `Match` fazem parte da classe `Adaptation` e implementam funções que facilitam a montagem desse elemento. Por fim a classe `PropertyCapability` estende a classe `Property` com algumas funções que automatizam a leitura e alteração dos atributos das tags do componente `Adaptation`.

A classe `Adaptation` define um ponto de adaptação da interface. A classe `Profile` define o perfil da adaptação, cada perfil tem vários `Matchs` que armazena o valor do elemento que satisfaz o perfil; a próxima subseção explica em detalhes como a adaptação é feita.

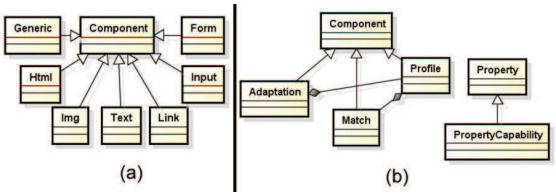

Figure 1: Principais classes da ferramenta.

Existem outras classes, que implementam a interface gráfica e realizam os detalhes das ligações entre as classes. A estrutura da ferramenta apresentada aqui é baseada na arquitetura utilizada pelo *DroidDraw*[13].

A ferramenta exibe de forma simplificada a estrutura da página por meio de modelos, ou seja, não é exibido exatamente como será a página e sim onde ficam os seus componentes. Na Figura 2 é ilustrada essa situação: o componente *body* está selecionado (2) (portanto a visão exibida é desse elemento) e possui os componentes *header*, *nav*, *section* e *footer* (1).A inserção de componentes é feita com o clique sobre o elemento no menu *Components* (4) da ferramenta. Ao clicar-se em algum elemento (por exemplo, *header*) a visão passa a ser desse elemento e é exibido um menu com suas propriedades (5), permitindo sua alteração. Também é permitida a exclusão desse elemento. Através do componente *link* (7) a ferramenta permite que sejam usados elementos pertencentes à Web 2.0[11]. A ferramenta também facilita a geração do código CSS para estilo da página, essa funcionalidade é acessada por meio da aba CSS (6) no painel superior direito.

É possível também inserir na interface formulários e campos de texto (disponíveis no menu *Components*) com a formatação definida pelo desenvolvedor. A ferramenta tem três formatações pré-definidas e são elas: alfanumérico, somente números e data. Essas regras são adicionadas a um arquivo XML e para adicionar uma validação basta editar o mesmo.

A geração de código é feita por meio do botão *Generate* (3), onde o código é gerado no painel inferior: código HTML (9) e código CSS (10). O código HTML produz um código XML em arquivo que representa o modelo da página. Esse modelo é interpretado por adaptador, cujo protótipo foi elaborado nessa pesquisa com a finalidade de validar a adaptação.

Para acessar a funcionalidade de adaptação é necessário inserir o componente *Adaptation* (8). Por meio de sua inserção é determinado que naquele ponto haverá uma adaptação. Ao criar esse componente é necessário informar quais serão os critérios de adaptação de acordo com o dispositivo e o usuário. O componente *Adaptation* permite escolher o tipo de perfil no qual se deseja incluir um elemento de adaptação. Para cada Adaptation inserido no modelo é possível adicionar vários critérios de adaptação. Cada um destes critérios será traduzido para uma regra que será executada durante o processo de adaptação.

A execução da adaptação é apresentada na Figura 3. A partir da chamada efetuada em um dispositivo com *Android*, o servidor Web efetua a requisição para respectiva página (1), esta gerada pelo *DroidRichWeb*. Quando um ponto de adaptação é encontrado na página é acionado o adaptador (2), este referencia as regras de adaptação da página que estão associadas ao ponto de variabilidade em questão (3),

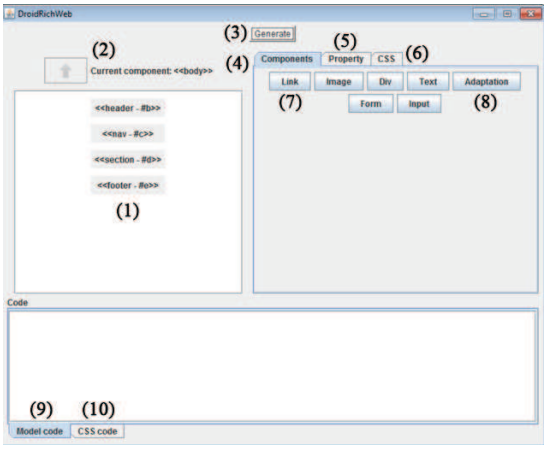

Figure 2: Visão Geral da Ferramenta.

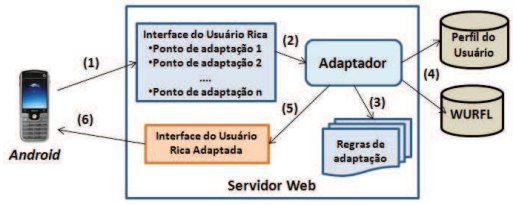

Figure 3: Execução da adaptação.

consultando os dados necessários dos perfis do usuário e do dispositivo (4). O adaptador então realiza as alterações na página (5). Este procedimento é realizado por toda a página requisitada. Ao final das adaptações a nova versão é enviada para o dispositivo (6). Como a ferramenta gera um código XML, este precisa ser transformado em um HTML válido e essa conversão é feita pelo adaptador. O adaptador é um *servlet* que recebe o arquivo XML como entrada e inicia sua interpretação. O adaptador utiliza o *WURFL* (*Wireless Universal Resource File*)[8] , que se trata de um arquivo XML que contém informações de milhares de dispositivos móveis. Assim o adaptador apenas copia as tags HTML ordinárias, e os pontos adaptáveis são analisados. Ao final um código HTML válido e adaptado é gerado.

4. VALIDAÇÃO DA FERRAMENTA

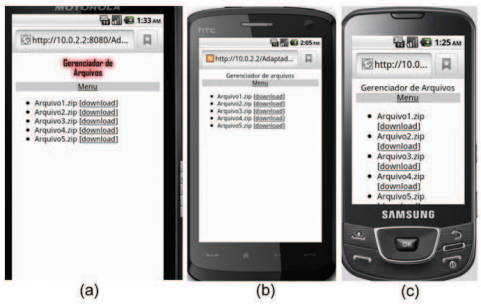

Figure 4: Interfaces adaptadas de acordo com a largura do display.

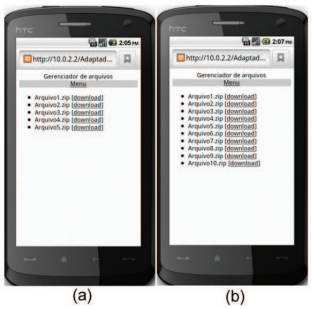

Figure 5: Interfaces adaptadas de acordo com a largura de banda.

Para validação da ferramenta foi desenvolvida uma aplicação de *download* e *upload* de arquivos, cuja interface adaptável foi modelada através do *DroidRichWeb*. A aplicação tem a finalidade de testar o modelo adaptável da interface de acordo com dois critérios: largura da tela do dispositivo (`max_image_width`) e largura de banda do usuário (`bandwidth`). A aplicação conta com um menu que é implementado usando os efeitos do componente *link* (efeitos da Web 2.0). Existem dois modelos de cabeçalho, que é adaptado de acordo com o dispositivo: um com imagem e um em texto puro. Caso a largura da tela do dispositivo seja menor que `350px` o texto puro é exibido, caso a largura seja maior ou igual, a imagem é mostrada. A exibição dos arquivos no servidor é adaptável de acordo com o usuário: se a largura de banda for maior que `127 kbps` serão mostrados os últimos 20 arquivos enviados, se for menor serão exibidos os últimos cinco.

No teste utilizamos três dispositivos: Motorola Droid (Milestone) de display `854 x 480`; Samsung i7500 de display `480 x 320` e HTC Touch de display `320 x 240`. Os dois primeiros foram testados em uma conexão de `64 kbps`, o último foi testado em uma conexão de `64 kbps` e em uma de `1 Mbps`.

Na Figura 4 podem ser observados três dispositivos conectados a uma rede de `64 kbps`. Na Figura 4(a) é apresentado o resultado no dispositivo Motorola Droid; na Figura 4(b), no HTC Touch e na Figura 4(c), no Samsung i7500. Já na Figura 5(a), temos o HTC Touch com conexão de `64 kbps` e na Figura 5(b), o HTC Touch com conexão de `1 Mbps`.

5. CONCLUSÃO

Este artigo apresentou a ferramenta *DroidRichWeb* que automatiza a criação interfaces ricas para *Android*, implementando componentes HTML 5 e de Web 2.0. A ferramenta permite além da modelagem da página web a especificação de pontos variáveis que deverão ser adaptados durante a execução da página de acordo com o perfil do dispositivo e do usuário.

O desenvolvimento da ferramenta já foi finalizado e nesta última versão, há funcionalidades de adaptação para os componentes *link*, imagens, *divs*, textos, formulários e campos de texto. Também foi desenvolvido um adaptador para o processamento da interface adaptável de acordo com os perfis de usuário e dispositivo.

6. AGRADECIMENTOS

Ao CNPq pelo apoio financeiro.

7. BIBLIOGRAFIA

[1] *Cascading Style Sheets 3 (CSS3)*. 2011. Disponível em: http://www.w3schools.com/css3/. Acesso em 14/10/2011.

[2] *Extensible Markup Language (XML)*. 2011. Disponível em: http://www.w3.org/XML/. Acesso em 05/05/2011.

[3] *HyperText Markup Language 5 (HTML5)*. 2011. http://www.w3.org/TR/2011/WD-html5-20110525/. Acesso em: 31/03/2011.

[4] *JDOM*. 2011. Disponível em: http://www.jdom.org/. Acesso em 05/05/2011.

[5] *Jo HTML5 Mobile App Framework*. 2011. Disponível em: http://www.joapp.com/. Acesso em 06/05/2011.

[6] *JQuery*. 2011. Disponível em: http://jquery.com. Acesso em: 31/03/2011.

[7] *Qooxdoo*. 2011. Disponível em: http://www.qooxdoo.org/. Acesso em 06/05/2011.

[8] Wurfl. 2011. Disponível em: http://wurfl.sourceforge.net/. Acesso em: 02/04/2011.

[9] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven development of context-aware web applications. *ACM Transactions on Internet Technology*, 7(1), 2007. Article 2.

[10] C. E. B. G. Daniel Billsus, Clifford A. Brunk and M. Pazzani. Adaptive interfaces for ubiquitous web access. pages 34, 38, 2002.

[11] P. J. Deitel and H. M. Deitel. *AJAX, Rich Internet Applications, and Web Development for Programmer*. Prentice Hall, first edition, 2008.

[12] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[13] DroidDraw. *Projeto DroidDraw*. 2011. Disponível em: http://www.droiddraw.org//. Acessado em: 05/05/2011.

[14] L. Findlater and J. McGrenere. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. pages 1257, 1256, 2008.

[15] Google. *Plataforma Android*. 2011. Disponível em: http://code.google.com/android/. Acessado em: 15/04/2011.

[16] E. G. Nilsson. Design patterns for user interface for mobile applications. *Advances in Engineering Software*, 40(12):1318, 1328, 2009.

[17] O. Pastor and F. Valverde. *Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach*, volume 5802 of *Lecture Notes in Computer Science*, pages 131, 144. 2009.

[18] F. Paternò, C. Santoro, and A. Scorcia. Automatically adapting web sites for mobile access through logical descriptions and dynamic analysis of interaction resources. *ACM Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 260–267, 2008.

[19] W. Viana and R. M. C. Andrade. Xmobile: a mb-uid environment for semi-automatic generation of adaptive applications for mobile devices. *Journal of Systems and Software*, 81(3):382–394, 2008.

[20] L. Wroblewski. *Web Form Design: Filling in the Blanks*. Rosenfled Media, 2008.