Personalizações para Acesso à *Web* em Computadores Móveis

Leonardo Teixeira Passos, Marco Túlio de Oliveira Valente
Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais
E-mail: leonardopassos@mg.unitech.br, mtov@pucminas.br

***Abstract*— Apesar dos avanços recentes em tecnologias de comunicação sem fio, o acesso à *Web* por meio de dispositivos móveis continua sendo laborioso. Por esta razão, diversas soluções foram propostas para personalizar páginas *Web* de modo a facilitar a sua exibição em dispositivos móveis. *Wrappers* são uma das soluções mais promissoras que tratam esta questão. Neste artigo, descreve-se um *wrapper* que tem como objetivo a criação de personalizações de páginas a serem acessadas em dispositivos computacionais móveis. Uma personalização é uma subpágina de uma dada página *Web* que contém somente as informações que um dado usuário deseja acessar em seu dispositivo móvel. O sistema de personalização proposto atende aos seguintes requisitos: (i) suporta a personalização de páginas *Web* independentemente de serem bem formadas ou não; (ii) suporta personalizações que são robustas a mudanças comuns em páginas *Web*; (iii) suporta a criação de personalizações através de uma interface gráfica, isto é, a definição de personalizações dispensa o usuário de qualquer espécie de codificação. No artigo, são descritos em detalhes os algoritmos e estruturas de dados que apoiam o processo de definição e extração de personalizações. Mostram-se também alguns resultados experimentais obtidos na criação de personalizações para algumas páginas *Web*.**

I. Introdução

Tecnologias de computação móvel prometem acesso à *World Wide Web* a qualquer momento e em qualquer lugar. No entanto, apesar dos progressos recentes em redes de comunicação sem fio e em dispositivos computacionais móveis, esta promessa ainda não foi totalmente concretizada. Um dos principais motivos é que o conteúdo atual da *Web* - mais de 4 bilhões de páginas, contando-se apenas aquelas indexadas pelo Google - foi em sua grande maioria projetado para acesso em computadores pessoais, isto é, equipamentos que não possuem limitações relevantes em termos de processadores, memória, dispositivos de entrada e saída, suprimento de energia e interfaces de comunicação em rede. Por outro lado, todas estas limitações são relevantes em dispositivos computacionais móveis, como assistentes pessoais digitais (PDAs) e telefones celulares. Como resultado, o acesso à *Web* a partir de computadores móveis é uma experiência na maioria das vezes pouco produtiva. Em geral, devido às limitações em sua capacidade computacional, os navegadores (*browsers*) disponíveis nestes equipamentos não são capazes de exibir páginas HTML com recursos mais sofisticados, como por exemplo, *applets*, *scripts*, animações ou mesmo imagens coloridas [13]. Estes equipamentos também possuem uma tela de tamanho reduzido - tipicamente de cerca de 3.5 polegadas em um PDA - o que dificulta a navegação em uma página projetada para uma tela bem maior. Por fim, mesmo com o surgimento no mercado de redes celulares de terceira geração, redes sem fio ainda possuem uma largura de banda inferior àquela disponibilizada por *backbones* corporativos. Assim, a navegação por páginas *Web* mais complexas, com diversas imagens, pode exigir um montante bem maior de tempo.

Nos últimos anos, algumas soluções foram propostas para personalizar e adaptar páginas *Web* para dispositivos móveis. Em geral, estas soluções podem ser dispostas em três categorias [4]:

- *Reestruturação de sites Web para dispositivos móveis*. Esta é a melhor solução do ponto de vista de usabilidade e é comumente empregada em *sites* populares, como portais de notícias e lojas virtuais. No entanto, a maior desvantagem se encontra no custo de criar e manter múltiplas versões do mesmo conteúdo;
- *Transcodificadores*, isto é, *proxies* (ou servidores intermediários) que dinamicamente recuperam e reformatam documentos da *Web* para visualização em dispositivos móveis. Como exemplo, temos os sistemas Digestor [2], PowerBrowser [3] e o sistema descrito em [15]. Transcodificadores são usados, por exemplo, para remover *scripts* e *applets*, converter HTML em WML, transformar imagens coloridas em preto e branco, dentre outras tarefas. Entretanto, transcodificadores não podem alterar a estrutura geral ou remover informações não-desejáveis de documentos *Web*. Assim, muitas vezes não é gerado um código adequado para visualização em dispositivos móveis;
- *Wrappers*, isto é, programas que automaticamente recuperam e extraem conteúdos específicos de um documento *Web* [12]. Dada uma página *Web*, *wrappers* são capazes de extrair somente um fragmento da mesma que um dado usuário considera fundamental para exibição em dispositivos móveis. Portanto, *wrappers* são uma solução poderosa para personalização de páginas *Web* de acordo com requisitos expressos por um usuário. Como exemplo de *wrappers* projetados com esse objetivo, podemos citar os sistemas WebViews [5] e SmartView [14].
  A geração de *wrappers*, no entanto, apresenta diversos desafios. As ferramentas de geração de *wrappers* devem ser capazes de criar sistemas precisos e robustos e, ao

mesmo tempo, não exigir grande esforço por parte dos usuários.

Neste artigo é descrito um *wrapper* para adaptação de conteúdo *Web* para dispositivos móveis, denominado PWA (*Personalizations for Wireless Access to the Web)*. O sistema proposto atende aos seguintes requisitos:

- Permite que usuários finais *personalizem* uma página da *Web*, isto é, delimitem apenas o conteúdo que consideram mais relevante na mesma, visando sua exibição em dispositivos computacionais móveis.
- A personalização de um documento é feita de forma totalmente visual, utilizando procedimentos padrões em sistemas com interface gráfica (como marcação de textos usando o *mouse* e cópia de texto para a área de transferência do sistema de janelas). Em outras palavras, não exige-se que o usuário do sistema domine qualquer linguagem para extração de conteúdo da *Web*, como XPath [17] ou WebL [9].
- O sistema PWA manipula páginas *Web* independentemente de serem bem formadas ou não. Páginas bem formadas são aquelas que sempre possuam *tags* de fechamento e cujos *tags* estão aninhados corretamente. Esse requisito é fundamental, pois, devido à grande tolerância dos *browsers* na exibição de páginas com erros, existe um número significativo de páginas *Web* que não são bem formadas.
- O sistema PWA suporta a definição de personalizações robustas, isto é, personalizações que podem lidar com mudanças comuns no conteúdo e na estrutura de páginas *Web*.

O restante deste artigo está organizado conforme descrito a seguir. A Seção II descreve o funcionamento do sistema PWA, incluindo os passos necessários para criação e recuperação de uma personalização por parte de usuários finais do sistema. A Seção III descreve a arquitetura interna do sistema, incluindo os algoritmos e estruturas de dados propostos pelo mesmo. A Seção IV descreve um experimento real de uso do sistema, no qual foram criadas personalizações para páginas conhecidas da *Web*. Estas personalizações foram então acessadas diariamente, durante 50 dias, de forma a verificar a robustez das mesmas a alterações na página original. A Seção V descreve outros sistemas para adaptação de conteúdo *Web* para computadores móveis. Por fim, a Seção VI conclui o artigo, apresentando as principais contribuições do sistema PWA e apontando possíveis trabalhos futuros.

II. O Sistema PWA

O sistema PWA possui dois módulos: o sistema de definição de personalizações e o servidor de personalizações. O sistema de definição de personalizações executa em computadores de mesa, conectados a uma rede fixa. É utilizado por usuários que desejam criar personalizações e exportá-las para servidores de personalizações. Tais servidores executam na rede fixa, uma vez que são *proxies* que encapsulam ações para recuperar e extrair personalizações de páginas *Web*. As personalizações resultantes são transmitidas para os dispositivos móveis dos usuários.

A. Sistema de Definição de Personalizações

O sistema de definição de personalizações é encarregado de prover uma interface conveniente ao usuário, de modo que este possa definir, por meio de uma interface gráfica, expressões de extração. Expressões de extração contêm informações utilizadas pelo algoritmo de extração de uma subpágina referente a uma personalização.

De modo a facilitar o processo de definição de personalizações, o usuário tem à sua disposição uma interface gráfica. Para criar uma personalização utilizando esta interface, o usuário deve seguir os seguintes passos:

1) O usuário abre a página a ser personalizada em um navegador de páginas *Web* de sua preferência;
2) O usuário abre também a aplicação de definição de personalizações indicando a página alvo;
3) O usuário, utilizando o *mouse*, seleciona no navegador a seção de interesse, isto é, a subpágina referente à personalização. Esta seleção é realizada utilizando os recursos de marcação de texto usuais em sistemas com interface gráfica, isto é, arrastando o *mouse* sobre o texto que deseja-se selecionar.
4) Utilizando os recursos de *copiar* e *colar* do sistema de janelas, o usuário copia a seleção feita no passo anterior para a área de transferência do sistema (usualmente denominada *clipboard*).
5) O usuário solicita à aplicação cliente a visualização de sua personalização;
6) A aplicação cliente exibe ao usuário a subpágina referente à personalização. Para tanto, a aplicação obtém o texto disponibilizado no *clipboard* e o utiliza como entrada para o algoritmo de recuperação de personalizações;
7) Se o usuário estiver satisfeito com o resultado, ele então solicita à aplicação a exportação de uma expressão de extração para o servidor de personalizações. Ao fazer isto, o usuário deverá fornecer um nome de identificação para a mesma.

A Figura 1 mostra a definição de uma personalização associada ao portal *Google News* (http://news.google.com). A área selecionada no navegador corresponde às notícias principais desse *site*, as quais farão parte da personalização. As informações restantes do *site* serão descartadas.

Ressalte-se que este processo é baseado em navegadores padrões de mercado e no recurso de “copiar e colar” disponível em qualquer sistema de janelas (Windows, Motif etc). Portanto, o sistema PWA é baseado em ferramentas e tarefas incorporadas à rotina de seus usuários. O sistema dispensa o domínio de linguagens de extração complexas ou de procedimentos laboriosos para se definir personalizações.

B. Servidor de Personalizações

O servidor de personalizações é responsável por armazenar expressões de extração e executar as mesmas, medi-

Fig. 1. Definição de uma personalização para o *site* Google News

ante a solicitação de usuários móveis. Para solicitar uma personalização, um usuário móvel deve, utilizando o navegador de seu PDA, acessar o servidor de personalizações e então proceder da seguinte forma:

1) O usuário entra com seu nome de usuário e senha;
2) O servidor valida os dados de entrada;
3) Se os dados de entrada estiverem corretos, o servidor exibe ao usuário uma lista com as personalizações salvas no mesmo;
4) O usuário clica no *link* correspondente à personalização de interesse;
5) O servidor recupera a página alvo e com as informações disponíveis na expressão de extração referente à personalização selecionada, extrai a subpágina desejada, devolvendo-a ao usuário.

A presença do servidor de personalizações na rede fixa dispensa o dispositivo móvel de qualquer processamento, além de diminuir a quantidade de informações trafegada na rede móvel, já que não se exibe no PDA a página original, mas apenas uma seção da mesma.

III. Implementação

Nesta seção são descritos os algoritmos e estruturas de dados utilizados na implementação do sistema PWA.

A. Estruturas de Dados

Existem três estruturas de dados básicas no sistema PWA:

- *Lista de tags*: lista cujos nodos armazenam informações sobre cada elemento existente em uma página HTML alvo. Esta lista é obtida a partir da análise sintática da página alvo. Na lista de *tags*, existem três tipos de nodos: nodos com informações sobre *tags* de abertura, nodos com informações sobre *tags* de fechamento e nodos de texto. A lista de *tags* permite também obter o *nodo pai* de qualquer nodo de texto ou *tag* de abertura presente na mesma. O nodo pai de uma *tag* de abertura é o nodo no qual a mesma deve estar sintaticamente aninhada. Por exemplo, o nodo pai de uma *tag* `LI` (item de uma lista) é uma *tag* `UL`.
- *Buffer*: vetor responsável por armazenar todo o texto encontrado numa página HTML. Nenhuma informação de formatação é armazenada no *buffer*;
- *Mapeador*: possui a responsabilidade de mapear intervalos de textos contido no *buffer* para intervalos da lista de *tags*. Através deste mapeamento é possível determinar a subpágina HTML referente a uma porção de texto.

A Figura 2 apresenta o relacionamento destas estruturas para um pequeno trecho de código HTML, referente à manchete principal do portal *Google News*.

A definição dessas três estruturas de dados – e também do algoritmo para geração de expressões de extração descrito na próxima seção – foi inspirada na arquitetura MVC (*Model-View-Controller*) [10], [6], proposta originalmente para implementação de interfaces gráficas em Smalltalk. Resumidamente, a arquitetura MVC organiza uma aplicação visual em três partes: o *modelo* armazena os dados da aplicação; a *visão* representa graficamente estes dados e o *controlador* recebe uma entrada por parte do usuário através da visão e a traduz para o modelo.

A idéia básica da solução proposta é criar um modelo da página *Web* selecionada e, a partir dos dados deste modelo, gerar a expressão de extração. Ressalte-se que o modelo é o único componente do padrão MVC efetivamente utilizado na implementação proposta. Não são utilizados os componentes visão (já que reutiliza-se a interface gráfica provida por um navegador) e controle (já que toda interação entre o navegador e a ferramenta implementada é baseada no mecanismo de marcação e cópia de texto provido pelo sistema de janelas).

B. Expressões de Extração: Primeira Versão

Nessa seção, descreve-se uma primeira versão do algoritmo para geração e avaliação de expressões de extração proposto no sistema PWA. Essa versão não é capaz de tratar alterações na estrutura da página alvo e, portanto, não corresponde exatamente àquela que é suportada pela atual implementação do sistema. No entanto, a descrição da mesma contribui para ilustrar de forma simples o princípio básico de funcionamento do sistema PWA. Na próxima seção, esse algoritmo será estendido para tratar mudanças estruturais no código HTML associado a uma personalização.

Quando o usuário executa a aplicação de definição de personalizações para uma página alvo, as estruturas lista de *tags*, *mapeador* e *buffer* são criadas. Estas três estruturas permitem extrair uma personalização a partir do texto da mesma. Após o usuário selecionar e copiar para o *clipboard* uma personalização, o sistema de definição de personalizações realiza os seguintes passos:

1) A aplicação obtém do *clipboard* o texto *T* da personalização;
2) A aplicação procura pelo texto *T* no *buffer*, obtendo um intervalo de índices;
3) A aplicação, através da estrutura mapeadora, mapeia os índices obtidos no passo anterior para índices da lista de *tags*, os quais irão delimitar uma sublista *S* de elementos

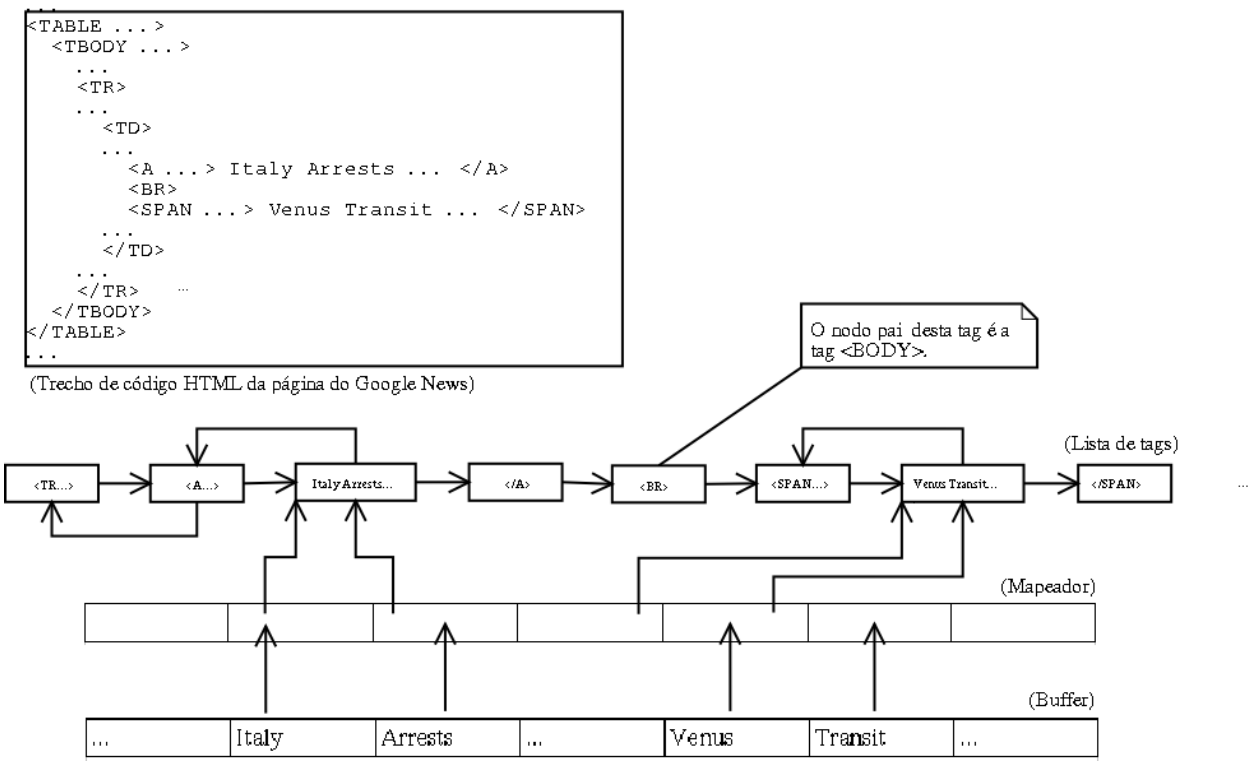

Fig. 2. Estruturas de Dados Usadas na Definição e Extração de Personalizações

que constituirão a personalização a ser apresentada ao usuário.

O usuário poderá então avaliar o resultado da personalização. Se estiver satisfeito, a expressão de extração dessa personalização é salva em um servidor de personalizações. Esta expressão consiste na tupla $(N, U, P(S), I)$, onde N é o nome atribuído à expressão de extração, U é a URL associada à página alvo, $P(S)$ é um padrão referente à sublista S da lista de *tags* e I é o número da ocorrência desse padrão na lista de *tags*. Como podem existir múltiplas ocorrências de S, o valor de I é usado para determinar a ocorrência de S associada à personalização que foi definida pelo usuário.

O padrão $P(S)$ consiste na transformação da sublista S da lista de *tags* em uma *string* que torne mais eficiente o processo de casamento de padrões. Para isto, a sublista S é percorrida e as seguintes tranformações são realizadas:

- Nodos texto são transformados na *string* `TXT`. No caso de nodos texto adjacentes, gera-se a *string* `TXT+`;
- Para nodos que representem *tags*, gera-se uma *string* equivalente ao nome da *tag* contida nesse nodo. Os atributos de uma *tag* na formação do padrão $P(S)$ são sempre desconsiderados, já que representam conteúdo *volátil*, isto é, frequentemente alterado.

Dada uma expressão de extração, o servidor executa os seguintes passos para obter a personalização associada à mesma:

1) O servidor recupera a página referente à URL U;
2) A partir da análise sintática dessa página, o servidor cria as estruturas *buffer*, mapeador e lista de *tags*;
3) O servidor procura pela I-ésima ocorrência do padrão $P(S)$ na lista de *tags*. Caso encontre essa ocorrência, a personalização a ser enviada ao usuário corresponde aos elementos da lista de *tags* que casaram com o padrão $P(S)$.

O *parser* usado para se criar as estruturas de dados descritas no passo (2) do algoritmo anterior foi implementado com auxílio do gerador de *parsers* `JavaCC` e de uma gramática de HTML definida em [8]. Ao contrário de *parsers* tradicionais XML, o *parser* utilizado não valida o documento de entrada junto a uma DTD e também não verifica sua boa formação. Assim, pode-se manipular qualquer página, independentemente dos erros de formação que esta contenha.

C. Expressões de Extração Robustas

A solução descrita anteriormente é extremamente frágil a *mudanças estruturais* que venham a ocorrer na página alvo. Diz-se que uma página alvo sofreu uma mudança estrutural se a lista de *tags* obtida no momento da recuperação de uma personalização for diferente daquela gerada quando da criação da mesma. Suponha, por exemplo, que no momento da definição da expressão de extração, o padrão de uma sublista *S* ocorra nos intervalos [200,300] e [500,600] da lista de *tags* de uma determinada página. Suponha ainda que a personalização desejada corresponda ao primeiro intervalo, isto é, *I=1*. Se no momento da recuperação da personalização,

TABLE I
TAGS VOLÁTEIS

A	ABBR	ACRONYM
ADDRESS	APPLET	AREA
B	BASEFONT	BDO
BIG	BLOCKQUOTE	BR
CENTER	CITE	CODE
COL	COLGROUP	DEL
DFN	DIV	EM
FONT	FRAME	FRAMESET
HEAD	HR	I
IMG	INS	ISINDEX
KBD	LINK	MAP
MENU	OBJECT	OPTGROUP
PARAM	Q	S
SAMP	SCRIPT	SMALL
SPAN	STRIKE	STRONG
STYLE	SUB	SUP
TBODY	TFOOT	THEAD
TT	U	VAR

uma mudança estrutural nessa página tiver inserido apenas uma *tag* no intervalo [200,300], o padrão não mais ocorrerá neste intervalo, mas apenas no segundo. Logo, uma resposta incorreta será retornada ao usuário.

Para tratar mudanças estruturais, o sistema PWA utiliza uma estrutura de dados denominada *esqueleto*. O esqueleto, assim com a lista de *tags*, é criado tanto no momento da definição da personalização quanto na recuperação da mesma.

O esqueleto é uma lista de *tags* onde *tags voláteis* são excluídas. *Tags* voláteis são *tags* que são freqüentemente inseridas, removidas e/ou alteradas em páginas *Web*. Exemplos incluem *tags* como `<FONT>`, `<B>`, `<I>`, `<OBJECT>` e `<IMG>`. A relação completa de *tags* voláteis é mostrada na Tabela I. Expressões de extração não devem considerar tais *tags*. A idéia é utilizar na recuperação de personalizações apenas *tags* que contenham texto ou que definam a estrutura de uma página HTML, tais como `<TABLE>` e `<TR>`, por exemplo.

Os passos para geração de uma expressão de extração na aplicação de definição de personalizações permanecem inalterados. No entanto, no momento em que o usuário escolhe a opção de salvar a expressão de extração, a aplicação cria o esqueleto da sublista *S* encontrada. O padrão gravado na expressão de extração não mais refere-se a *S*, mas sim ao esqueleto de *S*, isto é, *P(E(S))*. São incluídos também na expressão de extração, o texto *T* da personalização, sem qualquer formatação (isto é, o texto que o usuário marcou no navegador e copiou para a área de transferência) e os índices *i* e *j* que delimitam a sublista *S* da lista de *tags*. Salvar o texto *T* na expressão de extração é útil para tratar personalizações associadas a páginas estáticas, isto é, cujo conteúdo não muda ao longo do tempo. Já os índices *i* e *j* são uma indicação da posição original da personalização na página alvo.

Quando se transforma o padrão $P(S)$ no padrão $P(E(S)$ – padrão do *E(S)* – pode ocorrer de serem descartadas diversas *tags* voláteis, de forma que o padrão $P(E(S)$ passe a ter um número reduzido de elementos. No entanto, um padrão de tamanho pequeno pode originar múltiplos casamentos durante o processo de extração. Assim, define-se uma constante *k* que indica o número mínimo de elementos que esse padrão poderá conter. A escolha do valor de *k* é importante para que o casamento do padrão possa ocorrer com sucesso. Quanto maior o valor de *k*, maior será a dependência entre a expressão de extração e a estrutura do documento. Quanto menor for o valor *k*, maior será o número de casamentos obtidos para o padrão contido na expressão de extração. Experimentos realizados mostraram que para *k* igual a 10 obtém-se pouca dependência estrutural em relação à página alvo e, ao mesmo tempo, obtém-se um número reduzido de casamentos para o padrão da expressão de extração. Assim, adotou-se $k = 10$ na implementação do sistema.

TABLE II
EXEMPLO DE EXPRESSÃO DE EXTRAÇÃO

```
[Name] Exemplo
[URL] http://news.google.com
[Text] ITV.com Italy Arrests Suspected Madrid ...
[Offset] 238,308
[Left-Complement-Pattern]
[Pattern] TXT+ </table> TXT+ <table> <tr> <td> ...
[Right-Complement-Pattern]
```

Caso $P(E(S))$ tenha cardinalidade inferior a k, duas sublistas são criadas: uma sublista com os elementos da lista de *tags* logo à esquerda do início da sublista S, denominada Se, e uma sublista como os elementos logo à direita de S, denominada Sd. Estas sublistas são então transformadas em $E(Se)$ e $E(Sd)$, respectivamente, resultando no padrão $P(E(Se)) + P(E(S)) + P(E(Sd))$. A cardinalidade deste novo padrão deverá ser igual a *k*.

Por fim, uma expressão de extração robusta é definida pela tupla $(N, U, T, i, j, P(E(Se)), P(E(S)), P(E(Sd))$.

Um exemplo de uma expressão de extração para a manchete principal da página do *Google News* é mostrado na Tabela II.

D. Algoritmo de Extração de Personalizações

O algoritmo de extração de personalizações é acionado quando o usuário solicita, através de seu PDA, uma personalização. Para solicitar uma personalização, primeiramente o usuário deverá logar no servidor de personalizações, que então apresentará uma página contendo todas as personalizações anteriormente salvas no mesmo. Este processo foi descrito na Seção II-B.

O algoritmo de extração proposto em PWA é mostrado na Tabela III. Esse algoritmo recebe como entrada a expressão de extração $(N, U, T, i, j, P(E(Se)), P(E(S)), P(E(Sd)))$ e então realiza os seguintes passos:

1) Nos passos 1 e 2, a página associada à URL *U* é recuperada e o *buffer*, mapeador, le a lista de *tags* são criados. No caso específico do módulo servidor, existe um quinta estrutura, denominada *MapeadorEsqueletoParaListaTags*, responsável por mapear intervalos do esqueleto em intervalos da lista de *tags*;
2) No passo 3, o texto *T* é casado junto ao *buffer*. Se o casamento ocorrer com sucesso e os índices casados,

TABLE III
ALGORITMO DE EXTRAÇÃO DE PERSONALIZAÇÕES

```
Entrada: N, U, T, i, j, P(E(Se)), P(E(S)),
         P(E(Sd))

1. Recupere a página P definida pela URL U;
2. Crie as estruturas lista de tags, mapeador
   e esqueleto associadas a P;
3. Case o texto T junto ao buffer;
4. Para cada casamento bem sucedido faça:
     ic,jc= índices resultantes do casamento;
     im= mapeiaBufferParaListaTags(ic);
     jm= mapeiaBufferParaListaTags(jc);
     Se (im == i) e (jm == j)
        retorne S(im,jm);
5. Crie a estrutura esqueleto;
6. Case P(E(Se))+P(E(S))+P(E(Sd)) junto ao
   esqueleto;
7. Para cada casamento p bem sucedido faça:
     ic,jc= índices resultantes do casamento;
     im[p]= mapeiaEsqueletoParaListaTags(ic);
     jm[p]= mapeiaEsqueletoParaListaTags(jc);
8. Dentre todos casamentos p bem sucedidos
     retorne S(im[p],jm[p]), onde |i-im[p]|
             seja mínimo
9. retorne FALHA;
```

quando mapeados para índices da lista de *tags*, forem os mesmos índices $[i, j]$ salvos na expressão de extração, então a personalização é a sublista da lista de *tags* definida por esses índices (passo 4). Isto ocorre quando a página *Web* é estática e não sofreu nenhuma alteração estrutural desde a definição da expressão de extração (ou quando ocorreram alterações apenas em posições posteriores à região da personalização);

3) Se os passos 3 e 4 falharem, ocorreu uma mudança no conteúdo da página P e/ou na estrutura da mesma. Primeiramente, cria-se o esqueleto (passo 5) e então o padrão $P(E(Se)) + P(E(S)) + P(E(Sd))$ é casado junto ao mesmo (passo 6). Esse casamento pode ser bem sucedido em diversas posições do esqueleto. Para escolher o casamento associado à personalização que foi requisitada, mapeiam-se os índices ic e jc de cada casamento bem sucedido p para índices da lista de *tags*, usando para isso a estrutura denominada *MapeadorEsqueletoParaListaTags* (passo 7). Obtém-se então índices $im[p]$ e $jm[p]$ relativos a elementos da lista de *tags*. Dentre estes pares de índice, escolhe-se aquele onde $|i - im[p]|$ possui o menor valor (passo 8). A personalização retornada ao usuário é a subpágina formada pelos elementos da lista de *tags* localizados entre os índices $im[p]$ e $jm[p]$.
4) Se nenhum casamento for bem sucedido no passo 7, o algoritmo falha (passo 9).

A idéia do algoritmo proposto consiste basicamente em usar o texto T salvo na expressão de extração para recuperar uma personalização, no caso de *não* ter havido nenhuma mudança na página entre o momento da criação da personalização e sua recuperação. Caso esta estratégia falhe, o algoritmo utiliza o padrão salvo na expressão de extração para tentar recuperar a personalização. Esse padrão é constituído apenas por *tags* não voláteis, isto é, *tags* que definem a estrutura básica de uma página e que, por isso mesmo, são menos sujeitas a modificações. A segunda tentativa procura então recuperar a personalização mesmo no caso de a página ter sofrido uma modificação, seja ela no conteúdo ou na estrutura da página. Para isso, casa-se o padrão salvo na expressão de extração junto ao esqueleto e tolera-se como resposta o casamento mais próximo à posição original da personalização, no momento em que a mesma foi criada. O início dessa posição é dado pelo índice i salvo na expressão de extração. Essa tolerância permite recuperar uma personalização mesmo no caso de terem sido inseridos e/ou removidos *tags* na página original após uma personalização ter sido associada à mesma.

As *tags* de uma personalização retornada pelo algoritmo de extração descrito podem estar desprovidas de um contexto de exibição adequado. Entende-se por contexto de exibição o ambiente no qual uma determinada *tag* deve estar aninhada. Suponha, por exemplo, que a primeira *tag* da personalização seja <TD>. Esta *tag* não poderá ocorrer sem a existência de uma outra *tag*, a <TR>, que por sua vez depende de <TABLE>. Desta maneira, quando uma personalização é retornada a um usuário, o contexto de cada *tag* é recuperado. Este processo é possível graças à existência em certos nodo da lista de *tags* de um ponteiro para o nodo pai dos mesmos, conforme descrito na Seção III-A.

IV. RESULTADOS EXPERIMENTAIS

De modo a avaliar a solução de personalização proposta pelo sistema PWA, foram realizados testes com três portais:

- *Bloomberg* (http://www.bloomberg.com). Duas personalizações foram criadas para esse portal: a primeira referente à seção *Market Snapshot* e a segunda referente à seção *Insight & Commentary*.
- *Google News* (http://news.google.com). Uma personalização foi criada para esse portal, associada à seção *Top Stories*.
- *Yahoo Sports* (http://sports.yahoo.com). Uma personalização foi criada para esse portal, associada à manchete principal do mesmo.

As personalizações criadas para esses portais foram acessadas durante 50 dias. Para avaliar a robustez do algoritmo de extração proposto, contou-se o número de dias em que as personalizações definidas permaneceram válidas, isto é, ao serem visualizadas, produziram os resultados esperados. O resultado deste experimento é apresentado na Tabela IV. Nessa tabela, a Coluna *Sucesso* contém o número de dias nos quais a personalização foi recuperada com sucesso. A Coluna *Mudanças Internas* contém o número de solicitações nas quais ocorreram mudanças internas à área da personalização. A Coluna *Mudanças Externas* contém o número de solicitações nas quais ocorreram mudanças externas à área da personalização.

Como mostrado na Tabela IV, as personalizações associadas aos *sites* da *Bloomberg* e *Yahoo Sports* se mantiveram estáveis por todo o período do experimento. A única personalização

TABLE IV
RESULTADOS EXPERIMENTAIS.

Site	Successo (dias)	Mudanças Internas (dias)	Mudanças Externas (dias)
Bloomberg (Insight & Commentary)	50	50	50
Bloomberg (Market Snapshot)	50	5	50
Yahoo Sports	50	50	50
Google News	43	50	50

que falhou durante o teste foi a personalização associada à página do *Google News*, totalizando 7 falhas. Quando falhou, esta personalização não produziu o resultado esperado devido a mudanças radicais em sua estrutura. É importante mencionar que todas as personalizações foram associadas a páginas extremamente dinâmicas, tanto em seu contéudo quanto em sua sua estrutura. Com exceção da segunda personalização da *Bloomberg*, todas as demais personalizações sofreram alterações em todos os 50 dias do experimento, incluindo alterações internas e externas à área da personalização. O algoritmo proposto foi robusto o suficiente para tratar todas essas alterações, excetuando as 7 falhas geradas quando da recuperação da personalização do *Google News*.

O experimento foi realizado em uma estação com a seguinte configuração: processador AMD-K6 2, com 333 MHz, memória primária de 256 Mb e sistema operacional Linux, kernel 2.4.18-5.

V. TRABALHOS RELACIONADOS

Com a disseminação de tecnologias de comunicação sem fio, diversos projetos e sistemas foram desenvolvidos nos últimos anos na área de adaptação de conteúdo *Web* para computadores móveis. Dentre esses, os mais próximos do sistema PWA são descritos a seguir nessa seção.

WebViews [5] é um *wrapper* que funciona de maneira similar a PWA, isto é, cabe ao usuário selecionar o conteúdo de uma página *Web* que deseja visualizar posteriormente em um PDA. Apesar de ser bastante flexível para o usuário final, o sistema WebViews apresenta duas deficiências que consideramos relevantes. A primeira delas deriva da utilização de XPath como linguagem para extração de componentes de páginas Web. Tendo sido projetada para uso em documentos XML, XPath requer páginas bem formadas, isto é, que sempre possuam *tags* de fechamento e cujos *tags* estejam aninhados corretamente. Devido à grande tolerância dos *browsers* na exibição de páginas com erros, o fato é que um número significativo das páginas da *Web* não são bem formadas. Os projetistas de WebViews sugerem então o uso de uma segunda ferramenta para corrigir tais erros. Mais precisamente, sugere-se o uso do sistema `tidy` [16] para realizar esta tarefa. No entanto, mesmo removendo diversos erros, esta ferramenta não é capaz de corrigir toda a gama de "mal formações" toleradas pelos *browsers*.

A segunda deficiência detectada na avaliação que realizamos sobre WebViews relaciona-se com a codificação de expressões XPath, o que é uma tarefa de relativa complexidade para usuários finais de dispositivos móveis. Os projetistas de WebViews também reconhecem este fato e propõem uma interface gráfica para permitir que usuários selecionem de forma interativa os componentes de interesse em uma página *Web*. No entanto, apesar de mencionada em [5], esta interface não foi disponibilizada para uso.

Smartview [14] é um *wrapper* que cria seções lógicas de um documento e retorna uma visão geral do mesmo ao usuário, na forma de uma imagem. Para obter esse imagem, o sistema Smartview constrói uma árvore DOM do documento de interesse e, percorrendo esta árvore, cria as seções lógicas da mesma. O resultado é uma imagem representando a página HTML, onde cada porção corresponde a uma seção lógica. Esta imagem é ainda adaptada às dimensões da tela do dispositivo móvel do usuário. Quando este clica em uma das seções, o sistema retorna o código HTML do fragmento selecionado.

Em [7], descreve-se um outro sistema de extração de conteúdo relevante de páginas *Web*. Para guiar o processo de extração são aplicados seis filtros: removedor de tags e atributos, removedor de anúncios, removedor de lista de *links*, removedor de tabelas vazias e o colecionador de lista de *links*. Os filtros, com exceção do colecionador de lista de *links*, visam podar a árvore DOM, gerada a partir da análise sintática do documento, de modo a simplificá-la. Tal simplificação não deve, no entanto, gerar perdas significativas no *layout* original. Para determinação do que será considerado conteúdo relevante, o sistema precisa ser previamente configurado por um usuário especializado. Essa configuração consiste no estabelecimento de regras globais, a serem aplicadas a toda espécie de requisição de páginas HTML por parte dos usuários.

Além de serem usados na adaptação de páginas para computadores móveis, *wrappers* podem ser usados também para extrair informação estruturada a partir do conteúdo de um documento HTML/XML. Neste caso, no entanto, o objetivo central é descobrir o esquema de dados subjacente a um documento da *Web* e não extrair componentes visualmente relevantes do mesmo. Como exemplo de *wrappers* com este propósito temos os sistemas Lixto [1] e DEByE [11]. Lixto, por exemplo, visa extrair um dado conteúdo e reformatá-lo em XML, de modo a permitir consultas e processamentos posteriores. Um *wrapper* gerado no ambiente Lixto é um conjunto de cláusulas codificadas em uma linguagem denominada Elog, codificação esta que é totalmente transparente ao usuário, já que este tem à sua disposição uma interface gráfica. Essas cláusulas representam um conjunto de padrões de extração e seus respectivos filtros, isto é, restrições definidas pelo usuário.

Em [13], descreve-se uma solução alternativa para acesso a *Web* em PDAs. A solução proposta consiste no uso de um servidor de aplicações, localizado na rede fixa, e onde executa-se o navegador *Web*, incluindo toda a lógica do mesmo (recuperação de páginas, *parser*, etc). Apenas o *dispaly* do navegador é enviado para o PDA, usando-se um protocolo de *display* remoto (*remote display protocol*). Assim, o servidor proposto desempenha o mesmo papel de servidores de terminais X, em um sistema X-Window. Como todo o processamento ocorre no servidor de aplicação, a solução proposta é capaz de tratar páginas cuja visualização falha nos navegadores usualmente disponíveis em PDAs, devido, por exemplo, à existência nas mesmas de *scripts*.

VI. Conclusão

Considera-se que o projeto e implementação do sistema PWA apresenta as seguintes contribuições:

- O sistema é baseado em uma interface gráfica, o que torna bastante amigável e flexível a seleção do conteúdo de interesse de uma página *Web*. O usuário tem que basicamente apenas "marcar" e "copiar" este conteúdo. A marcação é feita no próprio navegador que o usuário utiliza para acesso à *Web* enquanto conectado a uma rede fixa.
- O sistema utiliza uma implementação inspirada na arquitetura MVC para apoiar o processo de extração de conteúdo da *Web* em *wrappers*. Na implementação realizada, os componentes visão e controle dessa arquitetura são reutilizados, já que são representados, respectivamente, por um navegador e pelo mecanismo de copiar-e-colar do sistema de janelas. O único componente realmente implementado foi o modelo, o qual é representado pelas estruturas de dados descritas na Seção III.
- Os experimentos realizados mostraram que os algoritmos e estruturas de dados propostos são robustos a um número considerável de mudanças no conteúdo e/ou estrutura de páginas HTML. Particularmente, o sistema PWA é bastante robusto a modificações realizadas fora da área de uma personalização. Além disso, o sistema é robusto a modificações internas a uma personalização, desde que as mesmas incluam apenas *tags* voláteis.

O sistema PWA pode ser facilmente modificado de forma que o mesmo possa ser usado pelos próprios desenvolvedores de uma página *Web*, visando a disponibilização de uma versão da mesma para computadores móveis. Com isso, estes desenvolvedores não teriam todo o custo inerente à manutenção de duas versões de uma mesma página, já que a versão para computadores móveis seria gerada automaticamente, pelo sistema proposto neste trabalho.

Nada impede ainda que a personalização gerada pelo sistema PWA passe por um processo de transcodificação, após a sua geração e antes de ser transferida para um dispositivo móvel. Este processo pode, por exemplo, converter imagens para um determinado formato ou converter a página gerada para um formato de voz. Na verdade, esta alternativa pode aumentar em muito a eficiência de transcodificadores, já que a principal deficiência dos mesmos está relacionada com a complexidade inerente a diversas páginas *Web* – complexidade essa que uma personalização consegue reduzir consideravelmente.

Como parte de trabalhos futuros, pretende-se investigar a definição de personalizações incluindo seções diferentes e não contínuas de uma mesma página *Web*. Na versão atual do sistema, adaptações com essa característica demandam a geração de diferentes personalizações. Pretende-se também prover suporte a personalizações que incluam conteúdos provenientes de mais de uma página *Web*. Por fim, planeja-se realizar experimentos mais exaustivos de uso do sistema, incluindo um maior número de personalizações, por um período de tempo também maior.

Agradecimentos: Este trabalho originou-se de um projeto de pesquisa financiado pela FAPEMIG (processo CEX488/02).

References

[1] R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. In *The VLDB Journal*, pages 119–128, 2001.

[2] W. Bickmore and B. Schilit. Digestor: Device-independent Access to the World-Wide Web. In *6th International World Wide Web Conference*, 1997.

[3] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power Browser: Efficient Web Browsing for PDAs. In *Conference on Human Factors in Computing Systems*, pages 430–437, 2000.

[4] J. Freire. Using Wrappers for Device Independent Web Access: Opportunities, Challenges and Limitations. In *WWW Workshop on Mobile Search*, 2002.

[5] J. Freire, B. Kumar, and D. F. Lieuwen. WebViews: Accessing Personalized Web Content and Services. In *Tenth International World Wide Web Conference*, pages 576–586, 2001.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, Oct. 1994.

[7] S. Gupta, G. E. Kaiser, D. Neistadt, and P. Grimm. DOM-based Content Extraction of HTML Documents. In *12th World Wide Web Conference*, pages 207–214, 2003.

[8] JavaCC HTML Grammar. http://www.quiotix.com/downloads/html-parser.

[9] T. Kistler and H. Marais. WebL - A programming Language for the Web. *Computer Networks and ISDN Systems*, 30:259–270, Apr. 1998.

[10] G. E. Krasner and S. T. Pope. A Cookbook for Using the Model–View–Controller User Interface Paradigm in Smalltalk-80. In *Journal of Object Oriented Programming*, pages 26–49, Aug. 1988.

[11] A. H. F. Laender, B. Ribeiro-Neto, and A. S. da Silva. DEByE - Data Extraction by Example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[12] A. H. F. Laender, B. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A Brief Survey of Web Data Extraction Tools. *SIGMOD Record*, 31(2):84–93, 2002.

[13] A. Lai, J. Nieh, B. Bohra, V. Nandikonda, A. P. Surana, and S. Varshneya. Improving Web Browsing on Wireless PDAs Using Thin-Client Computing. In *13th World Wide Web Conference*, pages 143–154, 2004.

[14] N. Milic-Frayling and R. Sommerer. SmartView: Flexible Viewing of Web Page Contents. In *11th World Wide Web Conference (poster presentaion)*, 2002.

[15] P. I. Oliveira and C. Camarão. Adapting Web Contents to Wap Devices using Haskell. In *XXI International Conference of the Chilean Computer Science Society*, 2001.

[16] Tidy. http://tidy.sourceforge.net. Última visita: julho de 2004.

[17] XPath Language. http://www.w3.org/TR/ xpath20. Última visita: julho de 2004.