Uma Ferramenta para Desenvolvimento de Aplicações para Dispositivos Móveis

André C.O. Minelli, **Antonio A.F. Loureiro**

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
{minelli, loureiro}@dcc.ufmg.br

Resumo

A convergência entre telefonia celular, Internet e computação móvel tem trazido novos desafios em praticamente todas as áreas da Ciência da Computação. O principal objetivo é permitir o acesso a informações em qualquer lugar, a qualquer momento.

Muitas promessas têm sido feitas, várias tecnologias têm sido propostas e desenvolvidas, mas uma questão ainda não foi inteiramente resolvida: como desenvolver aplicações para este novo ambiente, que está evoluindo e se modificando rápida e continuamente. É nesta questão que este artigo é focado.

Abstract

The convergence among cellular telephony, Internet and mobile computing has brought new challenges to almost every Computer Science area. The main goal is to allow information access anywhere anytime.

Many promises has been made, many technologies has been proposed and developed, but an issue was not fully resolved yet: how to develop applications to this new environment which is quickly and continuously evolving and changing. This article is focused on this issue.

Palavras-chave: Desenvolvimento de aplicações, Dispositivos móveis, Computação móvel.

1 Introdução

A área de computação móvel está gerando uma nova e ampla frente de pesquisa e desenvolvimento, não só em Ciência da Computação mas em outras áreas de conhecimento. A crescente proximidade da computação móvel com a telefonia celular e com a Internet abriu ainda mais as possibilidades para a criação de novas tecnologias e, em conseqüência, de novos serviços e produtos.

Atualmente, dispositivos computacionais móveis com as mais diferentes características estão se tornando populares pelo mundo. Celulares com acesso a Internet, assistentes pessoais

(PDA's), *laptops* e vários tipos de *handhelds* proliferam-se em diversos setores e atividades, fazendo com que serviços e informações possam ser acessados a qualquer momento e em qualquer lugar. Estes dispositivos fazem parte da área de computação móvel, ou computação ubíqua [1]. Relógios, carros e até eletrodomésticos também deverão fazer parte deste conjunto em um futuro próximo.

Existem muitas projeções sendo feitas e expectativas sendo criadas com base nestas novas tecnologias. De acordo com o *DC Research* e *The Yankee Group* [2], em 2005 o número de usuários de telefones com acesso a Internet somarão aproximadamente 1 bilhão de pessoas no mundo, sendo que esta marca deverá ser alcançada ainda em 2003 se considerarmos a utilização de outros dispositivos móveis. Por outro lado, o *Gartner Group* [3] prevê que tecnologias sem fio irão tornar os trabalhadores móveis cerca de 30% mais produtivos. Existem muitos outros números semelhantes a estes, o que parece apontar para, no mínimo, uma grande massificação dos dispositivos móveis.

Apesar de todos os possíveis e prováveis benefícios, estes dispositivos possuem diversas limitações se comparados aos computadores disponíveis atualmente:

1. CPU menos poderosa
2. Menos memória
3. Consumo de energia restrito
4. Telas menores
5. Mecanismos de entrada de dados restritos
6. Largura de banda reduzida
7. Maior latência das respostas
8. Menor estabilidade de conexões

Enquanto os cinco primeiros itens são características de dispositivos móveis, os três últimos são devidos às restrições de um ambiente de comunicação sem fio. Apesar de todas estas restrições, estes dispositivos trazem consigo a conveniência de poderem acessar informações a qualquer momento e em qualquer lugar.

Podemos então distinguir duas formas principais de aplicações para dispositivos móveis operarem:

1. *Stand-alone*, onde o dispositivo móvel funciona como um "canivete suíço", incorporando sistemas utilitários diversos, como por exemplo calculadora, conversor de medidas e jogos. A principal característica destas aplicações é o isolamento: todo processamento feito é mantido no dispositivo, sem qualquer troca ou alimentação de entidades externas.

2. *Cliente/Servidor*, onde o dispositivo móvel deve comunicar-se com entidades externas para enviar ou receber dados. Podemos distinguir dois tipos de comunicação:

    - *Off-line*, quando o dispositivo deve ser conduzido até o sistema computacional fixo para que ambos possam comunicar entre si. É caracterizado pela necessidade de sincronização: o dispositivo móvel deve trocar e sincronizar informações com um

sistema computacional fixo central de tempos em tempos. Organizadores pessoais e bancos de dados são exemplos típicos.

- *On-line*, quando o dispositivo pode comunicar-se com o sistema computacional fixo a qualquer momento, normalmente através de um enlace sem fio. Compreende aplicações como leitor de e-mail, envio de mensagens de texto através de *Short Message Service* (SMS), serviços baseados em localização, acesso a bases de dados e *m-commerce*.

As aplicações cliente/servidor são as mais interessantes e úteis, tendo como dispositivos típicos os *laptops*, os celulares e os PDA's. Celulares e PDA's oferecem um desafio à criação destas aplicações por três razões principais: são mais limitados computacionalmente se comparados aos *laptops*; existem em grande variedade; possuem uma enorme gama de ferramentas para desenvolvimento de aplicações.

2 Motivação

A grande quantidade de dispositivos móveis traz consigo uma variedade de plataformas e, junto com esta, uma gama enorme de ferramentas de desenvolvimento. Somente o Palm OS, sistema operacional com penetração de mais 78% do mercado de *handhelds* em 1999 [4], possui mais de 15 ambientes de desenvolvimento reconhecidos pela Palm Inc., empresa responsável pelo Palm OS.

2.1 Desenvolvimento para celulares

O desenvolvimento de aplicações para celulares tem se baseado na utilização de linguagens de marcação, principalmente WML, sub-conjunto do WAP [5]; HDML, precursora do WAP, desenvolvida e mantida pela Openwave [7], anteriormente conhecida como Phone.com; e CHTML, subconjunto do HTML padrão, utilizado no sistema i-Mode, no Japão. Pode-se ainda incluir nesta lista o VoiceXML [9], linguagem de marcação para aplicações baseadas em comandos de voz.

2.2 Desenvolvimento para PDA's

Já o maior poder computacional dos PDA's, se comparados aos celulares, permite o desenvolvimento de aplicações com mais recursos. Linguagens de marcação ainda estão disponíveis (e, na prática, são bastante utilizadas), sendo em geral subconjuntos de HTML [14]. Mas neste segmento de dispositivos já é possível desenvolver aplicações nativas, utilizando todos os recursos que uma determinada plataforma ofereça.

Podemos separar os *handhelds* em três grandes plataformas: o *PalmOS* [10], desenvolvido pela Palm Inc., o *Windows CE/Pocket PC* [11] (que faz parte da família *Windows Embedded*) desenvolvido pela Microsoft, e o Symbian OS/EPOC [12], da Symbian. Existe ainda uma versão de Linux para PDA's [13], porém sua aceitação ainda é muito tímida.

2.3 Novas Tecnologias

Existem pelo menos duas tecnologias que poderão tornar bem mais ágil o desenvolvimento de aplicações para dispositivos móveis: *Java 2 Micro Edition* [15] e XHTML [16]. Ambas foram desenvolvidas visando a compatibilidade com uma variedade de dispositivos.

2.3.1 *Java 2 Micro Edition*

Espera-se que, em breve, dispositivos como celulares e *pagers* sejam muito mais personalizáveis do que hoje. Ao invés de virem com um conjunto de instruções praticamente imutável, este novos dispositivos permitirão aos usuários incluir novos serviços ou aplicações (provavelmente direto da Internet) [17].

Para atender a esta demanda, a Sun estendeu a tecnologia Java com a introdução da plataforma *Java 2 Micro Edition* (J2ME). Esta plataforma foi desenvolvida para ser modular e escalável, de modo a atender a grande variedade de dispositivos e configurações existentes. Baseia-se em três camadas:

- *Java Virtual Machine Layer*, uma implementação da máquina virtual Java adaptada para cada dispositivo.
- *Configuration Layer*, que define as características mínimas de uma máquina virtual e bibliotecas Java disponíveis para uma categoria de dispositivos.
- *Profile Layer*, que define o conjunto mínimo de *Application Programming Interfaces* (API's) disponíveis para uma família particular de dispositivos.

Para dispositivos móveis e demais sistemas com limitações de recursos semelhantes, J2ME define a *Connected Limited Device Configuration* (CLDC). Esta configuração é composta pela KVM (*K Virtual Machine*) e um conjunto de bibliotecas que poderão ser usados em dispositivos como celulares, *pages* e organizadores pessoais, por exemplo. Um perfil também foi definido para esta família de dispositivos, *Mobile Information Device Profile* (MIDP), e a Sun já disponibilizada uma ferramenta para desenvolvimento específica, J2ME Wireless Toolkit.

A vantagem de utilizar-se a plataforma J2ME está na portabilidade do código, além das vantagens da programação orientada a objetos. Uma aplicação poderá ser desenvolvida para toda uma família de dispositivos que sejam compatíveis com KVM/CLDC/MIDP, o que deverá incluir desde celulares até PDA's. Atualmente está disponível para a plataforma Palm a CLDC, que já conta com bibliotecas adicionais de interface gráfica.

2.3.2 XHTML

XHTML 1.0 é uma especificação do W3C que define um tipo de documento rico o bastante para ser usado para criação de conteúdo e formatação de documentos, mantendo ainda a acessibilidade para diversas classes de dispositivos.

XHTML é baseado na especificação HTML 4.1, adicionando-lhe modularidade e aderência ao XML. A apresentação do documento é descrita através de folhas de estilo (*Cascading*

Style Sheets - CSS), sem sacrificar a independência do navegador. E como um documento XML, permitirá a transformação de conteúdo através da aplicação de XSLT. Além disso, a especificação WAP 2.0 irá evoluir do WML para o XHTML *Basic*, uma adaptação do XHTML 1.0 apropriada para dispositivos com telas pequenas. Isto significa que, futuramente, XHTML será compatível com uma ampla variedade de dispositivos móveis, oferecendo grande flexibilidade para o desenvolvimento de aplicações para a Internet móvel e para diferentes clientes.

2.4 Considerações sobre o desenvolvimento de aplicações

Utilizando as tecnologias disponíveis atualmente, uma determinada aplicação desenvolvida para um dispositivo em uma plataforma utilizando uma dada ferramenta, em geral, somente poderá ser portada para um outro ambiente através de um processo custoso. Pouco do trabalho já realizado poderá ser aproveitado na nova plataforma. Uma ferramenta capaz de facilitar o desenvolvimento de aplicações para esta gama de dispositivos, que não ignore os futuros avanços (ou tendências) tecnológicos, e que possa atender as várias formas de aplicações será extremamente útil para integradores e desenvolvedores.

Esta complexidade para o desenvolvimento de aplicações para dispositivos móveis é bastante clara para qualquer desenvolvedor. Exatamente por isso, ferramentas que procuram criar aplicações que possam ser utilizadas em qualquer dispositivo já estão disponíveis no mercado. Muitos trabalhos foram desenvolvidos baseando a criação de aplicações em algum formato de documento XML, que então era transformado para um outro formato mais adaptado ao dispositivo [26]. Esta é a base teórica da grande maioria das ferramentas disponíveis atualmente [27, 28, 29, 31, 32, 30].

Por outro lado, tais ferramentas baseiam-se inteiramente na geração de aplicações baseadas em linguagens de marcação. Com raras exceções, nenhuma se propõe a gerar código nativo ou intermediário (que executa sobre algum tipo de *run-time*). Aplicações baseadas em uma linguagem de marcação oferecem as vantagens de serem facilmente atualizáveis e manipuláveis, podendo gerar de modo simples outras marcações. Porém é necessário que o dispositivo esteja sempre conectado a uma rede de dados para poder acessar estas aplicações. Muitas vezes este acesso não está disponível, seja por problemas de interferência ou sinal fraco, ou mesmo a simples ausência de infra-estrutura. Nestes casos faz-se imprescindível a geração de aplicações nativas, capazes de processar dados e, posteriormente, sincronizá-los com um sistema computacional central. Esta é uma lacuna que a ferramenta proposta procurará cobrir.

3 Análise de Requisitos

Seria extremamente útil uma ferramenta que fosse capaz de atacar os desafios deste desenvolvimento e que oferecesse flexibilidade para incorporar novos dispositivos e tecnologias. Os requisitos para tal ferramenta seriam:

1. Capacidade de descrever aplicações independente do dispositivo móvel que será utilizado, de modo que a mesma descrição possa ser utilizada em qualquer dispositivo. Esta

descrição deverá gerar uma aplicação nativa (da plataforma de desenvolvimento de cada dispositivo) ou então uma aplicação intermediária (que rodará sobre um *engine*, uma máquina virtual ou um protocolo de aplicação). Além disto, a descrição deverá ser modular o suficiente para que aspectos específicos da interface de um dispositivo possam ser definidos separadamente, para cada dispositivo suportado.

2. A descrição também deverá permitir a marcação de pontos de processamento da aplicação, onde código externo deverá ser inserido pelo projetista (usuário da ferramenta). Estas marcações podem ser parte da aplicação como um todo (isto é, será a mesma para todos os dispositivos) ou podem ser específicas para determinado dispositivo.

3. O código externo poderá ser inserido utilizando uma linguagem particular ou através de tecnologias multi-camadas como CORBA ou COM. No caso de uma linguagem, deverá prover ao projetista um método o mais uniforme possível de incluir seu código, seja para dispositivos móveis ou para o sistema central fixo. Uma linguagem já existente, adaptada à aplicação, poderia ser utilizada.

4. Deve possuir facilidades para a criação de mecanismos de sincronização, como por exemplo a geração de *stubs*, para a criação de aplicações *off-line*. Tais facilidades também deverão existir para a criação da aplicação correspondente no sistema central fixo.

5. Os protocolos de comunicação mais populares devem estar disponíveis para a criação de aplicações *on-line*, podendo citar TCP/IP e WAP. Mecanismos que permitam incorporar outros protocolos de comunicação são altamente desejáveis.

6. Deve ser possível adquirir dados e conteúdo de fontes diversas como XML, bancos de dados e protocolos de aplicação da Internet. Mecanismos que permitam a inclusão de outras fontes são altamente desejáveis.

7. Ser extensível, possibilitando que novas interfaces, novas aplicações e novos dispositivos sejam incorporados facilmente.

8. Incluir facilidades de verificação da aplicação sendo gerada. Por exemplo, um modelo como um grafo ou uma máquina de estados finitos poderia ser gerado. Desta forma o projetista poderá avaliar o comportamento desta aplicação e suas propriedades. A própria ferramenta poderá realizar tal verificação, ou exportar em uma representação capaz de ser analisada por outra, desenvolvida por terceiros).

9. Possuir facilidades para depuração de aplicações. Ferramentas como emuladores, depurador passo-a-passo e visualizador de estado da aplicação seriam muito úteis, juntamente com o item anterior.

10. Permitir recursos de internacionalização, ou seja, oferecer meios para que aplicações possam ser entregues em outros idiomas sem grande esforço.

11. Curva de aprendizado aceitável: a ferramenta não deve exigir esforços exagerados para sua utilização.

Cabe fazer duas considerações sobre este conjunto de requisitos. Primeiro, a implementação de apenas parte deste conjunto poderá ainda resultar em uma ferramenta bastante útil para determinados cenários de desenvolvimento. Segundo, mesmo que haja uma implementação de todos estes requisitos, ainda assim poderão existir aplicações que não poderão ser desenvolvidas usando tal ferramenta, provavelmente devido a singularidades de dispositivos, ambientes ou mesmo da própria aplicação.

4 Projeto e implementação da ferramenta

Esta seção descreve o projeto e implementação da ferramenta, a forma de utilização e um exemplo.

4.1 Decisões de projeto

A análise de requisitos feita na seção anterior foi feita de modo a atender amplamente as questões relacionadas ao desenvolvimento de aplicações para dispositivos móveis.

Uma ferramenta capaz de atender a todos os requisitos especificados demandaria recursos acima daqueles disponíveis dentro do escopo deste trabalho. Desta forma, somente um subconjunto destes requisitos serão selecionados e implementados nesta ferramenta, batizada como MAB (*Mobile Application Builder*). Apenas os requisitos relativos a verificação, depuração e internacionalização (respectivamente itens 8, 9 e 10) não serão implementados, mas foram considerados durante a etapa de projeto.

Outra decisão de implementação é a limitação do número de dispositivos de destino. A ferramenta implementada irá apenas gerar aplicações para as plataformas PalmOS utilizando CLDC e WAP 1.1 . Desta forma, foi coberto o desenvolvimento baseado em uma linguagem de marcação (WML 1.1) e na geração de código intermediário baseado em um *run-time* (KVM). Com isso consegue-se ainda cobrir as duas classes de dispositivos-alvo deste trabalho, os celulares e os PDA's, e atingir uma quantidade considerável dos aparelhos disponíveis atualmente.

Quanto às fontes de dados, serão suportadas apenas a aquisição de dados baseada em XML. Qualquer outra fonte deverá ser inserida através da transformação dos dados em um documento XML. A linguagem XML oferece uma interface flexível o bastante para permitir que qualquer fonte de dados seja importada para a ferramenta. Nesta implementação será utilizada a importação a partir de bancos de dados relacionais.

Para manter a regularidade e facilidade de desenvolvimento para o projetista (usuário da ferramenta), o código gerado pela ferramenta será Java. Além das vantagens de utilizar-se uma linguagem orientada a objetos, altamente portável e reusável, mantém-se a consistência ao utilizar a mesma linguagem para as aplicações em PDA's, baseado na J2ME, e celulares, onde poder-se-á utilizar *Java Servlets* [35] (capaz de gerar conteúdo em qualquer linguagem de marcação).

Códigos externos serão definidos através de uma classe abstrata Java, a qual possuirá a assinatura dos métodos definidos pelo projetista que ficará responsável por sua implementação.

Para a incorporação e importação de dados para as aplicações será utilizado XML. Usando-se esse formato para a leitura de dados, adaptadores para praticamente qualquer fonte de dados poderão ser criados e incorporados à ferramenta. A aplicação poderá acessar estes dados através de requisições HTTP, por exemplo, que então retornarão XML da fonte desejada. Estas requisições poderão ainda serem baseadas no envio de parâmetros ou utilizar SOAP[1] (*Simple Object Access Protocol* [36]). A vantagem de se permitir a utilização de SOAP via HTTP é que, desta forma, o projetista terá liberdade para utilizar outra linguagem de programação em sua implementação. Independente do modo que os dados serão buscados, o formato do documento XML retornado será bem definido *a priori*.

4.2 Arquitetura

Uma vez que linguagens de marcação e a arquitetura cliente/servidor são (e provavelmente continuarão sendo) utilizadas na criação de aplicações para dispositivos móveis, torna-se bastante evidente que um servidor *Web* possa ser usado como base da ferramenta. Esta arquitetura tem sido usada largamente na Internet, tanto fixa quanto móvel, e é capaz de atender perfeitamente aos requisitos propostos. A arquitetura proposta tem a vantagem de, potencialmente, ser escalável e portável, uma vez que é baseada inteiramente no modelo cliente/servidor e no protocolo HTTP. A figura 1 ilustra a arquitetura da ferramenta.

A ferramenta é baseada em uma interface de manipulação, onde o usuário define o fluxo de aplicação através de recursos de arrastar-e-soltar e folhas de propriedades. Foram definidas três unidades básicas de interação com o usuário, com as quais o projetista irá desenvolver uma aplicação:

- Listas: utilizadas para exibir um menu de comandos ou opções, geralmente para fins de navegação.
- Textos: utilizados para exibir informações para o usuário.
- Formulários: utilizados para receber entrada de dados, que pode consistir de combinações de itens como caixas de texto, campos de data ou hora, listas de seleção de opções e outros.

Estas três unidades de interação são defendidas em [18] e também utilizadas na ferramenta MobileDev. Possivelmente estas unidades não são suficientes para desenvolver qualquer aplicação para dispositivos móveis. No entanto, devem ser suficientes para desenvolver as aplicações mais usuais. As unidades de interação seguem o conceito de usabilidade de que as interações com os usuários devem ser simples. Isto torna-se mais relevante ao se considerar que as telas dos dispositivos têm tamanho reduzido e, em geral, possuem poucos recursos. Assim, apenas uma tarefa deve ser exibida por vez, de modo a tornar a navegação óbvia e limpa.

[1] SOAP é um protocolo baseado em XML para intercâmbio de informações através de chamadas de procedimentos remotos (*remote procedure calls*) em ambientes distribuídos. Pode ser usado em combinação com uma variedade de protocolos, mas a especificação mais madura o utiliza em conjunto com HTTP, o que será bastante conveniente dada a arquitetura proposta neste trabalho.

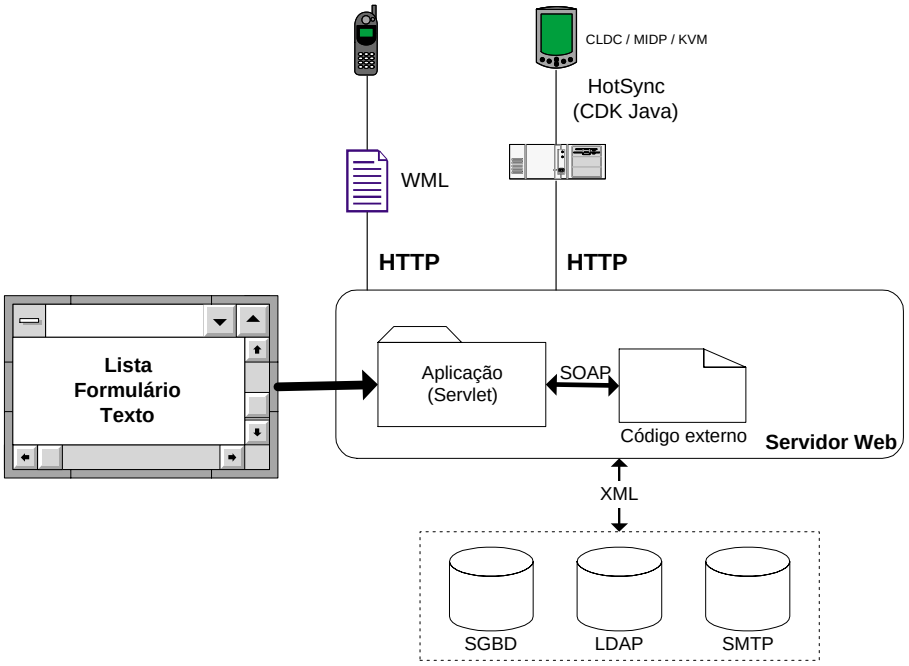

Figura 1: Arquitetura proposta

4.2.1 Descrição de parâmetros

Cada unidade pode receber parâmetros de uma unidade anteriormente exibida, e passar parâmetros para a próxima unidade a ser mostrada. Dentro de cada uma destas unidades o projetista pode definir um processamento de entrada, realizado antes que a unidade de interação seja exibida, e um processamento de saída, realizado após a finalização daquela unidade de interação. Estes processamentos consistirão de chamadas a código externo.

Para cada chamada poderão ser definidos parâmetros, que serão passados ao código externo, e os resultados esperados, retornados após processamento da chamada (parâmetros de retorno ou saída). Não serão definidos explicitamente os tipos (como tipo inteiro ou texto, por exemplo) destes dados: todos serão manipulados como `strings`. O projetista será responsável por fazer as devidas conversões de tipos em seu código.

Parâmetros serão definidos como uma lista da forma `nome=valor`, separados por ponto-e-vírgula. No caso dos resultados, apenas os nomes dos dados retornados serão definidos. Toda chamada retorna um número inteiro indicando o sucesso da operação (valor `0`) ou seu fracasso (qualquer outro valor). Este código de retorno poderá ser repassado para outras unidades.

Os parâmetros recebidos por uma unidade podem ser repassados diretamente para um chamada a um código externo. De modo semelhante, os resultados de uma chamada podem

ser passados para a unidade seguinte.

4.2.2 Ações

Finalmente, duas ações com semântica especial foram definidas. A primeira é identificada pela palavra-chave `!SYNC`. Indica que uma rotina de sincronização deverá ser chamada. A segunda é identificada pela palavra-chave `!EXIT`. Esta ação marca o final da execução da aplicação.

4.3 Formato de armazenamento

Aplicações geradas usando a ferramenta proposta são armazenadas através de documentos em XML. Uma das vantagens de utilizar-se XML é a facilidade de leitura (por pessoas) associada a facilidade de processamento (por máquinas). Além disso XML é um padrão internacional muito bem aceito atualmente.

Basicamente o documento é formado por um conjunto de unidades definidas pelo elemento `unit`. MAB recupera as unidades definidas em uma aplicação e insere os elementos no documento XML, de acordo com as propriedades de cada unidade.

4.4 Geração de código

O código será gerado a partir do documento XML especificado anteriormente. Cada plataforma de destino deverá possuir um gerador associado. Um gerador é um componente de *software* com uma interface padronizada, que receberá e enviará informações básicas de e para a ferramenta além de, provavelmente, solicitar ao projetista informações extras, essenciais a geração de código para a plataforma a que se destina.

Um gerador possuirá um *parser XML*, capaz de ler o documento armazenado e manipulá-lo de modo a gerar código para a plataforma a que se destina. Deverá ainda implementar a seguinte interface:

- `getName` - recupera uma sequência de caracteres com o nome do gerador para exibição na ferramenta
- `getVersion` - recupera a versão atual do gerador para exibição na ferramenta
- `getTargetPlataform` - recupera uma seqüência de caracteres com uma pequena descrição da plataforma de destino para exibição na ferramenta
- `needSync` - este método indica se o código gerado necessita de um mecanismo de sincronização. Deve retornar `1` se for necessário ou `0` em caso contrário.
- `initializeGenerator` - responsável por capturar informações necessárias a geração do código. Recebe dois parâmetros: o caminho e o nome da aplicação corrente. O comportamento deste método é altamente dependente da implementação e da plataforma de destino. Poderá, por exemplo, abrir janelas de diálogo para obter do projetista informações diversas, como opções de geração de código, local de destino do código gerado,

ou qualquer outra que seja relevante para o funcionamento do gerador. Retorna 0 se a chamada for bem sucedida.

- `generateCode` - este método só é chamado após o retorno a uma chamada do método anterior. É o local onde todo o trabalho de geração de código é realmente feito. É recomendável que ofereça ao usuário algum tipo de retorno visual do processamento. As seguintes padronizações também são recomendadas para o funcionamento de um gerador:
  - criar um diretório dentro da pasta onde o projeto corrente está armazenado, dentro do qual deverá gerar seus arquivos
  - criar todas as assinaturas e demais métodos auxiliares das rotinas externas definidas no projeto na classe `Routine`, definida no arquivo `Routine.java`

  Este método retorna 0 se a chamada for bem sucedida.

Um gerador qualquer que atenda a estes requisitos poderá, assim, ser registrado na ferramenta e apresentado como opção de plataforma de destino da aplicação desenvolvida pelo projetista. Esta interface deve ser implementada através de um servidor de automação ActiveX.

Foram implementados dois geradores, um para a plataforma WAP 1.1 e outro para dispositivos que utilizam J2ME/CLDC/MIDP.

4.4.1 Extensão do formato de armazenamento

A inclusão de código externo pelo projetista seria bastante trabalhosa no caso de alterações no projeto, pois a cada geração todos os arquivos são criados novamente. Desta maneira, o trabalho de codificação eventualmente já realizado teria que ser feito de novo. Mesmo que isso signifique apenas "recortar e colar" o código de um lugar para outro, isto demanda tempo.

Para facilitar o trabalho do projetista (usuário da ferramenta) foi criada uma extensão através de uma anotação[2] no documento XML de armazenamento.

Esta anotação será identificada pelo texto `@GENERATOR-OPTIONS`. Em seguida, cada um em uma linha, será indicado o nome do gerador de destino - o mesmo retornado pelo método `getName`, através do texto `@@GENERATOR-NAME`, e o nome do arquivo que implementa as rotinas externas, usando o texto `@@ROUTINES-CLASS`. Assim, esta anotação teria o seguinte formato dentro do documento de armazenamento:

```
<!--@@GENERATOR-OPTIONS @GENERATOR-NAME WML @ROUTINES-CLASS MyRoutine
-->
```

A interpretação desta anotação será feita pelo gerador. Este deverá utilizar a classe de rotinas externas definida na anotação contendo seu nome, se esta existir.

[2]Uma anotação é uma marcação com uma semântica especial definida geralmente dentro de comentários.

4.5 Sincronização

Para o caso de plataformas que necessitem de um mecanismo de sincronização com um servidor fixo será utilizado SyncML [39].

SyncML provê um *framework* para sincronização de dispositivos que não possuem conexão permanente a uma rede de comunicações qualquer. Para isso define um documento XML para fazer o intercâmbio de dados no processo de sincronização, além de um protocolo para diferentes procedimentos de sincronização.

Será utilizado o protocolo HTTP como mecanismo de transporte para os procedimentos de sincronização[3].

4.6 Acesso a dados

Acessos a bases de dados serão sempre realizadas a partir do código externo. Desta forma pretende-se conseguir um isolamento entre interface com usuário, lógica de negócio e banco de dados, seguindo o modelo de programação em camadas.

Qualquer tipo de dado, vindo de qualquer fonte, deverá ser retornado como um documento XML.

4.7 Plataforma de desenvolvimento

O ambiente Borland Delphi foi utilizado para o desenvolvimento da ferramenta propriamente dita. Esta escolha foi motivada pela maior familiaridade com este ambiente, além da grande disponibilidade de componentes de *software* gratuitos (que proporcionam um desenvolvimento acelerado).

Desta forma a ferramenta estará disponível apenas para a plataforma Windows. Mas existe a possibilidade de poder ser disponibilizada também para plataformas Linux, com o lançamento do pacote Borland Kylix[4].

O servidor Web escolhido para este trabalho foi o Tomcat [42] versão 3.2, uma implementação das especificações *Java Servlets API* 2.2 e *Java Server Pages* 1.1 derivada do projeto Apache.

Um componente fundamental deste trabalho é o *parser* XML da Microsoft [43], disponível gratuitamente. Este componente ActiveX possui bom desempenho e oferece bons recursos para recuperação de elementos e atributos de documentos XML, tendo facilitado bastante esta parte do desenvolvimento.

4.8 Como utilizar MAB

Será descrito nesta seção as etapas necessárias para modelar uma aplicação e implementá-la utilizando MAB. Basicamente são necessários quatro passos para implementar uma aplicação qualquer:

- Especificação da aplicação

[3] A especificação de SyncML prevê alguns mecanismos de transporte, entre eles HTTP, WSP e OBEX.

[4] Kylix é, na prática, uma versão do Delphi portada para o ambiente Linux

- Mapeamento das unidades
- Geração
- Inserção de código externo

4.8.1 Especificação da aplicação

A especificação da aplicação começa com a definição de requisitos. Dada esta especificação, procede-se uma etapa de modelagem tradicional. Em princípio qualquer técnica poderá ser usada para isto.

4.8.2 Mapeamento das unidades

Uma vez definida a aplicação será necessário mapear as interações com o usuário em uma ou mais unidades do MAB. As unidades podem ser de três tipos básicos: lista (estática ou dinâmica), texto (estático ou dinâmico) e formulário.

Geralmente sempre será possível fazer um mapeamento direto de uma interação qualquer para uma dessas unidades. Porém recomenda-se fazer, a seguir, uma análise mais cuidadosa deste primeiro mapeamento, pois invariavelmente será possível realizar otimizações através de parametrização de unidades e rotinas e da utilização de listas e textos dinâmicos.

4.8.3 Geração

O passo seguinte é gerar as aplicações, ou seja executar os geradores disponíveis.

4.8.4 Inserção de código externo

Por fim, o projetista deverá inserir seu código, implementando as assinaturas de cada rotina definida no projeto. Isto deverá ser feito implementando uma sub-classe da classe de rotinas criada por cada gerador. Para tanto o projetista deve indicar o nome do arquivo desta sub-classe, para cada gerador (possível através da extensão do formato de armazenamento).

4.9 Aplicação-exemplo: *m-commerce B2C*

Uma aplicação não trivial foi implementada usando MAB, criada para abranger várias das etapas de desenvolvimento que um projetista, eventualmente, enfrentaria ao utilizar esta ferramenta. Resumidamente, a aplicação permitirá que um cliente faça compras dentro de um conjunto de lojas, como um *shopping center*, por exemplo. Os clientes, após cadastrarem-se previamente na base da dados da aplicação, poderão efetuar consultas sobre os produtos disponíveis, incluir produtos em uma "cesta de compras" e, por fim, confirmar sua intenção de adquirir os produtos em sua cesta, efetuando realmente a compra.

A figura 2 ilustra o mapeamento de unidades final, desenhado na ferramenta. Na figura 3 é mostrado a sequência de interações com a aplicação para a compra de um produto.

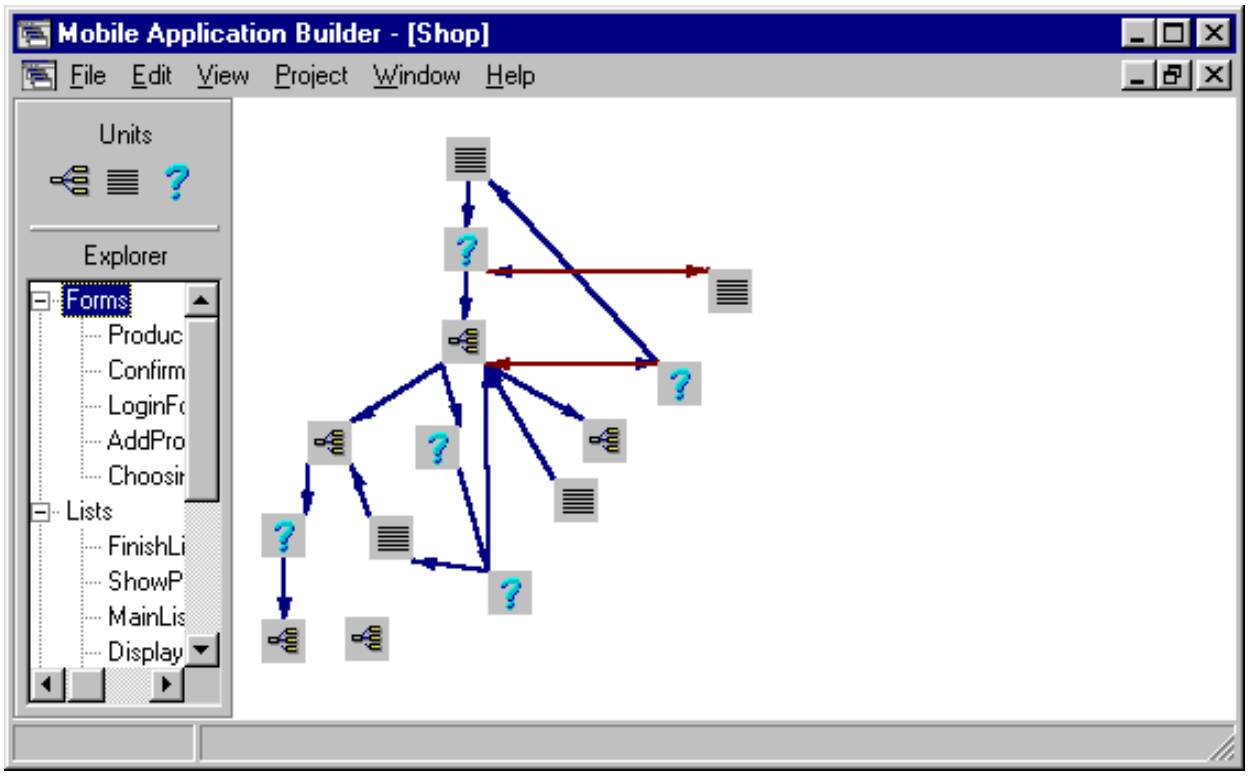

Figura 2: Unidades mapeadas no MAB - *M-commerce B2C*

Figura 3: Comprando um item através da aplicação

5 Conclusão

Após rever e analisar brevemente as alternativas de desenvolvimento existentes atualmente, foi levantado um conjunto básico de requisitos para uma ferramenta que pudesse ajudar na criação de aplicações para os dispositivos móveis existentes e para aqueles que poderão surgir com a chegada da novas tecnologias, como por exemplo 3G. Baseada num subconjunto destes requisitos, a ferramenta MAB foi especificada e implementada.

Baseada na geração de código em Java, esta ferramenta mostrou ser capaz de gerar aplicações para boa parte dos dispositivos móveis existentes atualmente. A mesma aplicação criada pela ferramenta atende a celulares compatíveis com WAP 1.1, servidos através de documentos JSP, e PDA's, desde que tenham suporte a J2ME. Nos testes realizados com MAB pode-se observar que aplicações relativamente complexas podem ser implementadas, como a aplicação de *m-commerce* proposta na seção anterior.

MAB tem como vantagem adicional a capacidade de criar protótipos rapidamente. Após especificar uma aplicação, a ferramenta criará toda a interface para as plataformas de destino. Desta maneira bastará ao projetista a criação de *stubs* nos pontos de inserção de código externo.

A extensibilidade de MAB também é uma característica importante. Geradores de código para outras plataformas podem ser desenvolvidos independentemente e depois incorporados a ferramenta. Mais ainda, é possível gerar qualquer tipo de código. Java foi a opção neste trabalho, mas não há nada que impeça um desenvolvedor de criar um gerador para código nativo de uma determinada plataforma, por exemplo.

Referências

[1] WEISER, M. *The computer for the 21st century*. 1991, Sci. Am. 265, 3 (Sept.), 94-104

[2] WAPMetrics – http://uolwap.uol.com.br/metrics1.php

[3] *How to Build a Wireless Office: The Next Wireless Revolution* – http://www.gartnertechwatch.com/public/static/techwatch/sponsor_100900.html

[4] Palm Brasil - Imprensa 2001 – http://www.palm.com/br/press2001/102901.html

[5] *WAP - Wireless Application Protocol* – http://www.wapforum.com/what/index.htm

[6] RISCHPATER, R. *Desenvolvendo Wireless para Web*. Makron Books, 2001.

[7] Openwave Systems Inc. – http://www.openwave.com/

[8] *XML - Extensible Markup Language* – http://www.w3.org/XML/

[9] *VoiceXML Forum* – http://www.voicexml.org

[10] Palm OS Platform – http://www.palmos.com/

[11] Microsoft Windows CE – http://www.microsoft.com/windows/embedded/ce.net/

[12] Symbian OS/EPOC – http://www.symbian.com/technology/symbos-v6x-det.html

[13] Embedded Linux Consortium – http://www.embedded-linux.org/

[14] *Web Clipping Application* – http://www.palmos.com/dev/tech/webclipping/

[15] *Java 2 Platform, Micro Edition* – http://java.sun.com/j2me/

[16] W3C HyperText Markup Language Home Page – http://www.w3.org/MarkUp/

[17] *Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices - White Paper. Sun Microsystems*, 2000.

[18] *Applications for Mobile Information Devices - White Paper. Sun Microsystems*, 2000.

[19] *Connected Limited Device Configuration* – http://java.sun.com/products/cldc/

[20] *K Virtual Machine* – http://java.sun.com/products/cldc/

[21] *Mobile Information Device Profile* – http://java.sun.com/products/midp/

[22] *JavaTM 2 Platform Micro Edition, Wireless Toolkit* – http://java.sun.com/products/j2mewtoolkit/

[23] *Palm CLDC Reference Implementation* – http://java.sun.com/products/cldc/

[24] XHTML 1.0 – http://www.w3.org/TR/xhtml1

[25] XHTML *Basic* – http://www.w3.org/TR/xhtml-basic

[26] MESSIAS, E. *Adaptação automática de interfaces para dispositivos WAP utilizando XSL*. Dissertação de Mestrado - DCC/UFMG, 2001.

[27] MobileDev 2.0 – http://mobiledev.speedware.com/

[28] OracleMobile Online Studio – http://studio.oraclemobile.com/

[29] AvidRapidTools – http://www.avidwireless.com/AVIDRapidTools.htm

[30] HiddenMind Active Universe Platform – http://www.hiddenmind.com/products.html

[31] Everypath Mobile Application Platform – http://www.everypath.digitalenterprises.com/

[32] SmartServ – http://www.smartserv.com

[33] SmartServ W_2W Platform – http://www1.smartserv.com/solution/w2w.htm

[34] XML:DB Projects – http://www.xmldb.org/projects.html

[35] *Java Servlet Technology* – http://java.sun.com/products/servlet/

[36] *Simple Object Access Protocol* – http://www.w3.org/TR/SOAP

[37] *Common Object Request Broker Archictecture*. OMG, Julho de 1995.

[38] *Component Object Model Technologies* – http://www.microsoft.com/com/

[39] *SyncML Protocol* – http://www.syncml.org/

[40] *JavaBeans Component Architecture* – http://www.java.sun.com/products/javabeans/

[41] TURAU, V. *Making legacy data accessible for XML applications.*

[42] *The Jakarta Project – Tomcat* – http://jakarta.apache.org/tomcat/

[43] Microsoft XML Core – http://www.microsoft.com/xml/