Adaptando Sistemas Existentes para um Ambiente de Execução Multiplataforma

Francisco Trindade[1], Marcelo Pimenta[1], Fábio Petrillo[1,2],Cirano Iochpe[1,2]

[1] Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática, Porto Alegre, Brasil, 91.501-970
[2] Companhia de Processamento de Dados do Município de Porto Alegre (PROCEMPA)

{fmtrindade,mpimenta}@inf.ufrgs.br, {petrillo,ciranoi}@procempa.com.br

Abstract. As the technology evolves, the existence of different computational devices has made ad-hoc software development no longer acceptable in the development of multi-platform software applications. This article presents UsiXML4ALL, a software tool developed to facilitate the creation/adaptation of multiplatform applications. UsiXML4LL acts as a renderer, mapping concrete UI's described in UsiXML to multiple platforms, and also as a connector, linking the rendered UI to application logic code developed possibly in multiple programming languages.

Palavras-chave: Interfaces Plásticas, Linguagens de descrição de interfaces de usuário, UsiXML, Multiplataforma.

1 Introdução

O avanço da tecnologia dos equipamentos computacionais e das redes de comunicação associado ao advento de diferentes dispositivos computacionais, como computadores portáteis e telefones celulares, fez surgir a necessidade de aplicações que possam ser executadas em múltiplas plataformas, em diversos tipos de equipamentos, com diferentes capacidades de recursos computacionais. Assim, pode-se desejar executar uma mesma aplicação seja em um PC *desktop*, seja na Internet via *browser* ou mesmo em um PDA ou celular.

Essa situação estabelece novos desafios para desenvolvedores de sistemas interativos [17], como desenvolvimento de novas aplicações e/ou adaptação de sistemas existentes para um ambiente de execução multiplataforma. Para estes desafios, uma das soluções propostas é o conceito de plasticidade, o qual visa à reutilização de descrições de interface, eliminando a necessidade de múltiplos desenvolvimentos para múltiplas plataformas. O termo "plasticidade" tem como inspiração a propriedade de materiais que se expandem e contraem devido às condições de ambiente sem quebrar, preservando a continuidade de uso [6]. Especificamente, nesse artigo o termo multiplataforma corresponde à definição de contexto de uso, incluindo múltiplos dispositivos, sendo que alguns trabalhos também consideram múltiplas modalidades (interação gráfica e vocal, por exemplo) e

múltiplas condições de ambiente existentes quando o software está sendo executado (condições de luminosidade, perfil do usuário). Sob a ótica da IHC, plasticidade é a capacidade de um sistema de suportar variações no contexto de uso, ainda assim preservando sua usabilidade.

De forma a desenvolver interfaces plásticas, diversas técnicas foram propostas, as quais podem ser classificadas de acordo com a *World Wide Web (W3C) Note on Authoring Techniques for Device Independence* [20], em três grupos: *single authoring*, *multiple authoring* e *flexible authoring*. Cada uma dessas técnicas é descrita brevemente a seguir [20].

- *Multiple Authoring*: O desenvolvedor cria um tipo específico de aplicação para cada dispositivo ou categoria. Essa situação ocasiona a (re)criação e manutenção das interfaces de usuário para cada plataforma, sendo extremamente custosa, mas também provendo o maior controle sobre os resultados obtidos.
- *Single Authoring*: Nesse grupo, apenas uma implementação da interface de usuário é criada, a qual é adaptada para um dispositivo específico antes de ser apresentada para o usuário. Técnicas dessa categoria podem ser subdivididas em técnicas que utilizam **vocabulários ou *toolkits* independentes de plataforma**, como AUIML [3] ou UIML [1], técnicas que **estendem linguagens de marcação estabelecidas**, como RIML [8], ou ainda técnicas que utilizam **desenvolvimento de interfaces baseado em modelos**, como XIML [19] ou UsiXML [11].
- *Flexible Authoring*: Situação na qual o desenvolvedor combina técnicas de *single* e *multiple authoring*.

Nesse trabalho é apresentada uma ferramenta para adaptação de aplicações existentes para um ambiente multiplataforma, utilizando uma linguagem de descrição de interface de usuário (LDIU) de alto nível baseada em modelos, seguindo a filosofia *single authoring.*

Muitas propostas usam LDIUs para este fim (ver seção 3 adiante), pois permitem a descrição da interface em diferentes níveis de abstração, além de fornecer uma maneira uniforme de criar interfaces multiplataforma e até mesmo multimodais. Além disso, a maioria sendo linguagens de marcação, elas se tornam fáceis de serem aprendidas e utilizadas, possuindo potencial para serem adotadas por uma grande comunidade de desenvolvedores.

Dentre as linguagens existentes, a UsiXML foi escolhida para a realização desse trabalho obviamente por suas características técnicas mas também por que seu material e informações estão disponíveis sem custo, e principalmente por possuir uma comunidade de desenvolvedores e pesquisadores aberta e ativa atualmente que oferece um efetivo suporte e um rico ambiente de discussão entre seus usuários.

Para possibilitar a utilização de LDIUs no desenvolvimento de software, são necessárias ferramentas capazes de interpretar os modelos de interface existentes, realizando a reificação da interface de usuário concreta na plataforma-alvo do aplicativo.

Esse trabalho apresenta a ferramenta UsiXML4ALL, a qual tem como proposta utilizar a linguagem de descrição de interfaces de usuário UsiXML para possibilitar a adaptação de sistemas existentes - além do desenvolvimento de novas aplicações - para um ambiente multiplataforma. A ferramenta apresentada tem como objetivo

facilitar o desenvolvimento de IU multiplataformas, realizando a reificação de descrições de interface concretas (CUI, conceito que será explicado adiante) em UsiXML, de permitir a interconexão das IUs renderizadas com diferentes linguagens de lógica de aplicação.O principal objetivo do aplicativo é auxiliar o desenvolvedor, atuando no processo de engenharia de interfaces de usuário. Como será esclarecido no restante do artigo, não se tem a intenção de atuar na definição da IU, assim como não são consideradas questões de usabilidade da IU renderizada.

O restante desse trabalho está organizado da seguinte maneira: a seção 2 descreve de forma introdutória a *User Interface eXtensible Markup Language* (UsiXML). Na seção 3 são apresentados alguns trabalhos relacionados enquanto a ferramenta UsiXML4ALL, sua arquitetura e seu funcionamento são descritos na seção 4. A seção 5 descreve o estudo de caso desenvolvido com adaptação de um sistema existente para um ambiente multiplataforma, e a seção 6 apresenta as conclusões.

2 User Interface eXtensible Markup Language (UsiXML)

A *User Interface eXtensible Markup Language* (UsiXML) é uma linguagem de descrição de interfaces de usuário proposta com o objetivo de capturar as propriedades essenciais para a especificação, descrição, projeto e desenvolvimento de interfaces de usuário [11].

Como forma de atingir esse objetivo, a UsiXML é estruturada de acordo com os quatro níveis de abstração definidos no framework de referência Cameleon [6], o qual permite expressar o ciclo de desenvolvimento de interfaces de usuário para aplicações sensíveis ao contexto, conforme descrito na figura abaixo (Figura 1).

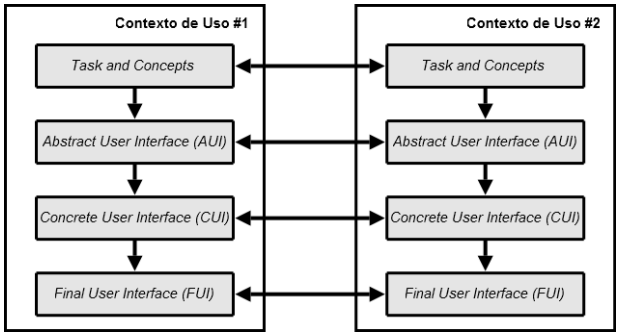

Figura 1. Os quatro níveis básicos do framework de referência Cameleon [6].

Os quatro níveis básicos existentes nesse framework são [6]:

- *Task and Concepts*: nível onde são definidas as tarefas interativas (*tasks*) sob o ponto de vista do usuário, em conjunto com os objetos (*concepts*) que são manipulados para a realização destas tarefas.

- *Abstract User Interface* (AUI): Nível de abstração em que a interface com usuário é definida de forma independente de qualquer modalidade de interação (gráfica ou vocal, por exemplo).
- *Concrete User Interface* (CUI): Definição que se caracteriza por ser independente do tipo de plataforma computacional ou conjunto de dispositivos de determinada plataforma.
- *Final User Interface* (FUI): Descrição final da interface de usuário, a qual pode ser executada ou interpretada em um determinado contexto de uso (ou seja, em uma plataforma específica, com um conjunto de dispositivos específicos, utilizando objetos de interação específicos).

Para representar os níveis de abstração do framework de referência, a UsiXML define os diferentes conceitos que estão presentes em uma IU. Apenas os mais relevantes para esse artigo serão apresentados aqui, sendo maiores detalhes apresentados em outros trabalhos [6, 11, 21].

O modelo de tarefas (*Task Model*) descreve as diversas tarefas que podem ser realizadas pelo usuário em uma interação com o sistema computacional. Para modelar as tarefas do sistema, UsiXML utiliza uma versão estendida da notação ConcurTaskTree (CTT) [14]. Nessa representação, um modelo de tarefas é composto por tarefas e relacionamento entre elas. As tarefas são descritas por um nome, um tipo e sua freqüência. Já os relacionamentos entre tarefas podem ser de dois tipos: decomposição, permitindo a representação de modelos hierárquicos, e temporal, possibilitando a especificação de uma relação temporal (seqüência, simultaneidade, etc..) entre tarefas de mesmo nível.

A interface de usuário abstrata (*Abstract User Interface*) representa uma expressão canônica da interface de usuário de forma a ser independente de qualquer modalidade de interação ou plataforma computacional. Uma AUI é composta por objetos de interação abstratos (*Abstract Interaction Objects*), os quais consistem em uma abstração de componentes presentes na maioria das plataformas, tanto gráficas, como janelas e botões, como para outros modos de interação, como vocal, por exemplo.

A interface de usuário concreta (*Concrete User Interface*) permite a especificação de uma interface de usuário de forma que seja dependente de modalidade, definindo o tipo de interação que será utilizado pela interface, e independente de plataforma, tornando possível a renderização da descrição de interface para múltiplos tipos de plataforma.

Um exemplo de declaração de uma interface contendo um *label* e um botão é mostrado na figura abaixo (Figura 2). Basicamente, esta especificação descreve uma janela (*window*), a qual contém três elementos de tipo *box* aninhados. Esse elementos agrupam os componentes de interação da interface, um campo de texto (*textComponent*) e um botão (*button*). Pode-se observar que, inseridas na declaração dos componentes estão seus atributos, como *id* e *name* para o botão, por exemplo.

```
<window id="W1" name="Main Window">
        <box … type = “main” splittable=true detachable=false… >
                <box … type = "horizontal" >
                <textComponent id="TX1" name="Text1" offsetVertical="top"
                        offsetHorizontal="center" defaultContent="Hello World!"/>
                </box>
                <box type="horizontal">
                        <button id="B1" name="OkButton" defaultContent="OK" />
                </box>
        </box>
</window>
```

Figura 2. Especificação UsiXML de uma CUI [11].

3 Trabalhos Relacionados

Os trabalhos relacionados à ferramenta apresentada nesse artigo podem ser classificados em duas categorias: a) as diversas ferramentas que trabalham com descrições de interfaces de usuário em UsiXML e b) as ferramentas de renderização de interfaces de usuário, tanto para a linguagem UsiXML como para outros modelos de descrição de interfaces de usuário.

Dentre as ferramentas que manipulam UsiXML, destaca-se SketchiXML [9], capaz de gerar uma descrição concreta de interface de usuário (CUI) em UsiXML utilizando como entrada descrições de interface desenhadas à mão (*sketchs*). Para a criação de descrições de interfaces em UsiXML, a ferramenta GrafiXML [12] é um editor de recursos visuais, através do qual o usuário especifica a interface desejada posicionando os componentes na tela e gerando como saída a descrição da interface projetada em UsiXML. Atuando no processo de engenharia reversa de IUs, a ferramenta ReversiXML [5] permite a criação de descrições UsiXML a partir de páginas HTML, tanto no nível de CUIs como de AUIs.

Renderização é o nome dado ao processo de mapear descrições de interfaces para interfaces concretas. Em relação ao framework de referência Cameleon, o termo renderização se refere ao processo de reificação que transforma uma definição concreta de interface (CUI) em uma interface de usuário final (FUI). A ferramenta que realiza o processo de renderização é denominada renderizador de interfaces.

Dentre os renderizadores descritos na literatura, o QTKiXML [10] mapeia interfaces de usuário descritas em UsiXML para a linguagem Tcl-Tk. Como o ambiente de execução necessário para linguagem existe para diferentes plataformas, a interface obtida como saída do renderizador pode ser considerada multiplataforma. Outro trabalho com características semelhantes descreve o FlashiXML [4], o qual também realiza a renderização de descrições de interface de usuário em UsiXML para interfaces finais. Neste caso, porém, a interface resultante é descrita em modo vetorial, podendo ser interpretada por qualquer plataforma equipada com plug-ins Flash ou SVG.

Outro trabalho similar propõe o InterpiXML [15], que realiza a renderização de descrições concretas de interface de usuário em UsiXML utilizando o toolkit de componentes visuais Swing, para a linguagem Java.

Em relação ao uso de outras LDIUs além da UsiXML, destacam-se alguns renderizadores que utilizam UIML - a biblioteca Uiml.NET [13] renderizando

interfaces descritas através da linguagem UIML para a plataforma .NET, e TIDE [2] (Tranformation-based Integrated Development Environment) renderizando descrições de interfaces UIML para a linguagem de programação Java – e TeresaXML, para a qual há o ambiente TERESA (Transformation Environment for InteRactivE Systems representAtions) [14], o qual permite a criação de interfaces de usuário para múltiplas plataformas de computação a partir das descrições TeresaXML. Também nessa categoria, o projeto MONA [20] investigou a interação multimodal em dispositivos móveis. Um dos objetivos do projeto foi o desenvolvimento de um método *single authoring* para a criação de interfaces de usuário multiplataforma.

UsiXML4ALL é um projeto similar aos trabalhos apresentados acima, já que explora a criação de aplicações multiplataforma. Como diferencial, a ferramenta possui também o objetivo de possibilitar a conexão da interface desenvolvida com lógicas de aplicação em múltiplas linguagens de programação.

4 UsiXML4ALL

De forma a criar IUs multiplataforma baseadas no framework de referência Cameleon, o ciclo de vida mostrado na Figura 3 deve ser segued. Esse ciclo de vida é baseado em um modelo de tarefas abstrato, e passa por diversas reificações até chegar à IU final.

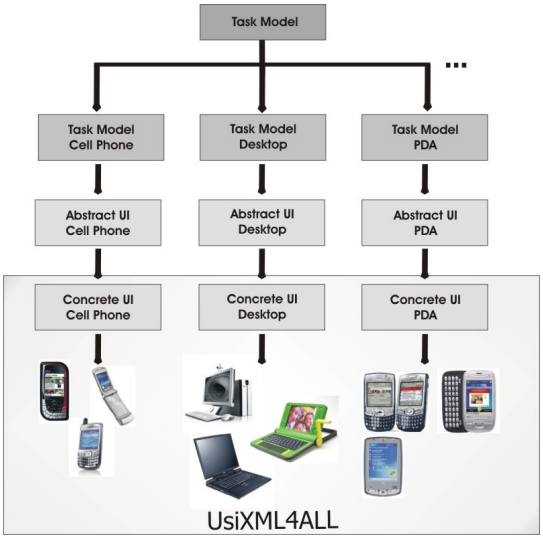

Figure 3. Ciclo de vida de IU multiplataforma.

Todas as etapas desse processo podem ser realizadas com a utilização de ferramentas de apoio, as quais podem realizar um mapeamento automático ou semi-automático de um nível para outro.

A UsiXML4ALL é um renderizador de IUs projetado para ser utilizado na ultima etapa do ciclo de vida mostrado na Figura 3, mapeando CUI's descritas em UsiXML para IUs finais para um dispositivo específico. Além disso, a UsiXML4ALL fornece um outro nível de independência, permitindo a conexão da IU renderizada com lógicas de aplicação desenvolvidas em diferentes linguagens de programação. Deve ser deixado claro que a ferramenta desenvolvida é um renderizador e não uma ferramenta de design, e assim não possui como objetivo detectar e/ou solucionar problemas de usabilidade e limitações existentes em cada plataforma. Conforme mostra a Figura 3, essas questões devem ser atendidas por etapas anteriores do ciclo de vida da IU, tanto de forma manual, como automática, através da utilização de outras ferramentas que trabalham com UsiXML.

A motivação para o desenvolvimento da UsiXML4ALL vem do fato que as ferramentas de renderização mostradas na seção anterior possuem algumas limitações:

- Não propõem o auxílio para a adaptação de sistemas já existentes.
- Não descrevem como é feita a conexão de interface de usuário gerada com a lógica de aplicação existente, embora a lógica de aplicação, nesses casos, deva ser desenvolvida utilizando a mesma tecnologia da ferramenta de renderização sendo utilizada.

Ao limitarem a sua utilização a uma determinada tecnologia, essas ferramentas restringem o número de aplicações que poderão se beneficiar das mesmas, além de dificultar sua utilização com códigos-fontes já existentes. Essa situação fica clara no caso de um aplicativo já existente, desenvolvido para uma única plataforma, como a plataforma *desktop*, por exemplo. Com a necessidade de se portar esse aplicativo para múltiplas plataformas, o desenvolvedor deverá encontrar uma ferramenta compatível com a linguagem de programação utilizada na criação de sua aplicação (C#, C++, Java). Apesar de menos grave, esta situação existe na criação de novos aplicativos também. Caso o desenvolvedor deseje que seu aplicativo atenda a múltiplas plataformas, ele deverá escolher a linguagem de desenvolvimento de acordo com as ferramentas de renderização existentes, limitando o seu universo de escolha.

UsiXML4ALL é um renderizador que supera estas limitações e tem como diferencial ser projetado para possibilitar tanto a renderização de interfaces de usuário utilizando múltiplas plataformas, como a conexão de diferentes lógicas de aplicação desenvolvidas em diferentes linguagens de programação.

Além disso, é extensível: a cada nova linguagem a ser atendida, é necessária apenas a criação de um novo módulo de conexão, e não de toda uma ferramenta de renderização.

Dessa forma, os principais objetivos da UsiXML4ALL são:

- Possibilitar a renderização de descrições concretas de interface (CUI) descritas em UsiXML para múltiplas plataformas.
- Permitir a conexão da interface de usuário gerado com lógicas de aplicação desenvolvidas utilizando diferentes linguagens de programação.
- Possuir uma arquitetura extensível, tornando possível a adaptação da ferramenta para novas plataformas, contextos de uso e linguagens de programação, quando se tornar necessário.

De modo a atingir os objetivos propostos na seção anterior, foi projetada uma arquitetura modular, capaz de atender as necessidades existentes atuais, assim como

aceitar a criação de extensões para novas plataformas e linguagens que devam ser atendidas no futuro.

A arquitetura proposta é mostrada na figura abaixo (Figura 4). Nessa figura, as linhas pontilhadas dizem respeito ao processo de renderização da interface de usuário, enquanto as linhas contínuas dizem respeito ao processo de conexão com a lógica de aplicação.

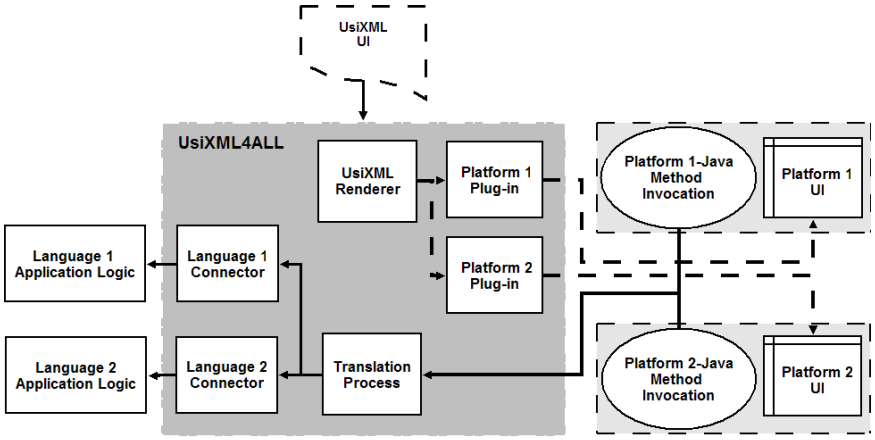

Figura 4. Arquitetura geral da ferramenta UsiXML4ALL.

Para realizar a etapa de renderização, a UsiXML4ALL recebe como entrada um arquivo contendo uma definição de interface segundo a especificação UsiXML (*UsiXML UI* na Figura 4) contendo uma descrição concreta (CUI) da interface desejada. Esse arquivo é validado pela ferramenta, a qual encaminha o mesmo para o *plug-in* de renderização relativo à plataforma-alvo desejada (*Platform 2 Plug-in* na Figura 4, por exemplo). O *plug-in* de renderização é responsável então por criar a interface final correspondente (*Platform 2 UI* na Figura 4) à descrição da interface obtida como entrada, sendo que as invocações de métodos existentes nessa descrição são direcionadas para o processo de tradução do aplicativo (*Translation Process* na Figura 4). De modo a realizar essa tarefa, o algoritmo de renderização percorre hierarquicamente a descrição do modelo de interface concreta (CUI) contido no arquivo UsiXML, instanciando os componentes contidos no mesmo de acordo com as características especificadas.

Durante o processo de renderização, o conteúdo presente na interface é obtido junto ao modelo de recursos (*resource model*) da descrição da interface, enquanto os métodos da lógica de aplicação a serem invocados são obtidos junto ao modelo de domínio (*domain model*) da interface.

Para conectar a interface com a lógica de aplicação em múltiplas linguagens de programação, o processo de tradução (Processo de Tradução na Figura 4) recebe as invocações de métodos da interface e realiza sua tradução para um formato independente de linguagem. Nesse momento, o método é então repassado para a extensão correspondente a linguagem da lógica de aplicação (Conector Linguagem 1

na Figura 4, por exemplo), a qual converte novamente a chamada, fazendo-o chegar até o código sendo executado (*Language 1 Application Logic* na Figura 4). No caso da existência de valores de retorno, o procedimento inverso é executado.

Para realizar essa tarefa, o conector é subdividido em duas partes, uma desenvolvida na linguagem Java, e outra na linguagem da plataforma da lógica de aplicação. Essas duas partes são conectadas através de chamadas realizadas através da interface nativa da linguagem Java, JNI (*Java Native Interface*), a qual possibilita a interconexão de código Java com outras linguagens de programação.

Nesse processo, a descrição do método a ser invocado é transmitida através de JNI para o código do conector da plataforma-alvo, e então são utilizadas técnicas de reflexão para realizar a invocação do método com base no seu nome e atributos.

5 Estudo de Caso: Fiscalização Eletrônica

De forma a validar a utilização da ferramenta UsiXML4ALL na adaptação de sistemas existentes para um ambiente multiplataforma, um estudo de caso foi realizado em conjunto com a Companhia de Processamento de Dados do Município de Porto Alegre (PROCEMPA), no âmbito do Projeto Fiscalização Eletrônica.

Esse estudo de caso teve como objetivo a adaptação de um sistema existente para um ambiente multiplaforma, permitindo a sua utilização tanto em dispositivos móveis como em computadores de mesa, utilizando a tecnologia *Java Swing* com Java SE e Java ME *Connected Device Configuration* (CDC).

Esse projeto é uma ação de modernização das tarefas de fiscalização da Secretaria Municipal de Indústria e Comércio de Porto Alegre (SMIC), seguindo as diretrizes do Programa Gestão Total da Prefeitura Municipal de Porto Alegre, em parceria com a PROCEMPA. A proposta central deste projeto é disponibilizar de forma automatizada as informações dos estabelecimentos comerciais a serem vistoriados pelos agentes de fiscalização da SMIC, através de um dispositivo móvel contendo os alvarás de funcionamento a serem vistoriados.

Com o objetivo de estudar as possibilidades de adaptação das interfaces do sistema, foram selecionados dois casos de uso do Projeto de Fiscalização Eletrônica: 1) Consultar Alvará e 2) Baixar Alvará. Estes casos de uso foram inicialmente implementados de fato na plataforma Microsoft .Net, sendo construídos de forma *ad hoc* através da ferramenta de construção de interfaces do Visual Studio. As figuras 5.1 e 5.2 ilustram a execução da aplicação no ambiente de teste do Microsoft Visual Studio .Net.

Figuras 5.1 e 5.2. Telas dos casos de uso implementadas de maneira *ad-hoc* com o Microsoft Visual Studio.

O caso de uso "Consultar Alvará" permite a um agente de fiscalização encontrar um alvará ao qual pretenda exercer alguma ação fiscalizatória, consultando os dados básicos do estabelecimento e o seu status (Figura 5.1). O processo inicia com a apresentação da interface contendo duas possibilidades de filtros: por número do alvará ou por logradouro. Conforme o atributo informando, é feita uma pesquisa na base de dados, alimentando o formulário detalhado com as informações do alvará encontrado.

Já o caso de uso "Baixar Alvará" permite a um agente de fiscalização efetuar a baixa ex-oficio de um alvará encontrado após a pesquisa de alvará. Com o alvará selecionado, o agente aciona o botão e o sistema efetua sua baixa, atualizando a interface com a mudança de status.

Para adaptar o sistema para um ambiente multiplataforma, as interfaces de usuário de ambos os casos de uso foram descritas em UsiXML, de forma a poderem ser interpretadas pela ferramenta UsiXML4ALL. Esses arquivos contêm não apenas a definição dos componentes presentes na interface, mas também a definição de seu conteúdo e comportamento dinâmico. A Figura 6 apresenta uma amostra do código UsiXML4ALL da IU do caso de uso "Pesquisar Alvará". Nessa descrição podem ser observados os componentes *outputText* e *inputText*, que representam os rótulos e campos de texto da interface sendo descrita.

```
<uiModel xmlns="http://www.usixml.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.usixml.org/ http://www.usixml.org/spec/UsiXML-ui_model.xsd"
    id="5_20" name="5" creationDate="2007-03-20T20:31:31.281-07:00" schemaVersion="1.6.4">
    <head>
    </head>
    <auiModel id="5-aui_20" name="5-aui"/>
    <cuiModel id="5-cui_20" name="5-cui">
        <window id="window_1" name="window_1" width="320" height="480" bgColor="#FFFFFF">
            <gridBagBox id="grid_1" name="grid_1"
                gridHeight="11" gridWidth="5">
                <constraint gridx="1" gridy="1" gridwidth="1" gridheight="1" weightx="1.0"
                          weighty="0.0" fill="horizontal" insets="5,5,1,5" anchor="center">
                   <outputText id="output_numero" name="output_numero" isVisible="true"
                          isEnabled="true" textColor="#000000"
                       content="/uiModel/resourceModel/cioRef[@cioId='output_numero'/@content"/>
                </constraint>
                <constraint gridx="1" gridy="2" gridwidth="3" gridheight="1" weightx="1.0"
                          weighty="0.0" fill="horizontal" insets="1,5,5,5" anchor="center">
                    <inputText id="input_numero" name="input_numero" isVisible="true"
                          isEnabled="true" textColor="#000000" maxLength="50" numberOfColumns="15"
                        isEditable="true"/>
                </constraint>
```

Figura 6. Amostra do código de descrição da IU para o caso de uso "Pesquisar Alvará" em UsiXML.

No contexto do caso de uso "Pesquisar Alvará", o comportamento dinâmico da interface se refere à chamada da função *getAlvara*, que implementa a pesquisa do alvará de acordo com os parâmetros fornecido pelo usuário.

Devido à independência de linguagem de programação fornecida pela UsiXML4ALL, o código desenvolvido na tecnologia *.Net* no contexto do Projeto Fiscalização Eletrônica pode ser acessado sem nenhuma modificação, permitindo uma fácil migração do aplicativo.

Para fazer isso, as chamadas de métodos realizadas pela IU são direcionadas para a ferramenta UsiXML4ALL, informando o nome e os parâmetros da função a ser executada. Essa definição é então traduzida para a linguagem de programação alvo, e então o método é chamado. A Figura 7 mostra o trecho de código que é executado na invocação do método.

Na figura 7, o método *getAlvara* declarado na classe *AlvaraFacade* é preparado e invocado, usando para isso a classe *logicConector*, a qual é baseada no padrão *Command* [GoF].

```
logicConector.initializeMethod();
logicConector.prepareMethodClass("AlvaraFacade");
logicConector.prepareMethodName("getAlvara");
logicConector.prepareMethodStringParameter(
        textoPesquisa,0);
logicConector.executeMethod();
```

Figura 7. Código executado para invocação do método *getAlvara*.

Dessa forma as interfaces de usuário descritas em UsiXML podem ser utilizadas em duas plataformas diferentes, sendo que as mesmas podem ser conectadas a lógicas de aplicação desenvolvidas em diferentes linguagens de programação. Essas

configurações são mostradas na Figura 8, onde a plataforma 1 é uma plataforma de aplicativos móveis, baseada na configuração Java ME CDC e a plataforma 2 é uma plataforma desktop, implementada em Java SE. Em relação à lógica de aplicação, a lógica A é implementada em Java, e a lógica B é implementada em C# .NET.

Desse modo, além de a aplicação poder ser executada em diferentes plataformas a partir da mesma descrição de interface, a mesma passa a ser independente de linguagem de programação. Sendo assim, caso no futuro se deseja modificar a tecnologia sendo utilizada no desenvolvimento do projeto, esse processo pode ser realizado de forma simples e sem nenhuma alteração para o usuário final. Tendo como base a Figura 8, 4 combinações diferentes de plataformas – linguagens de programação podem ser criadas (A,1;B,1;A,1;B,2).

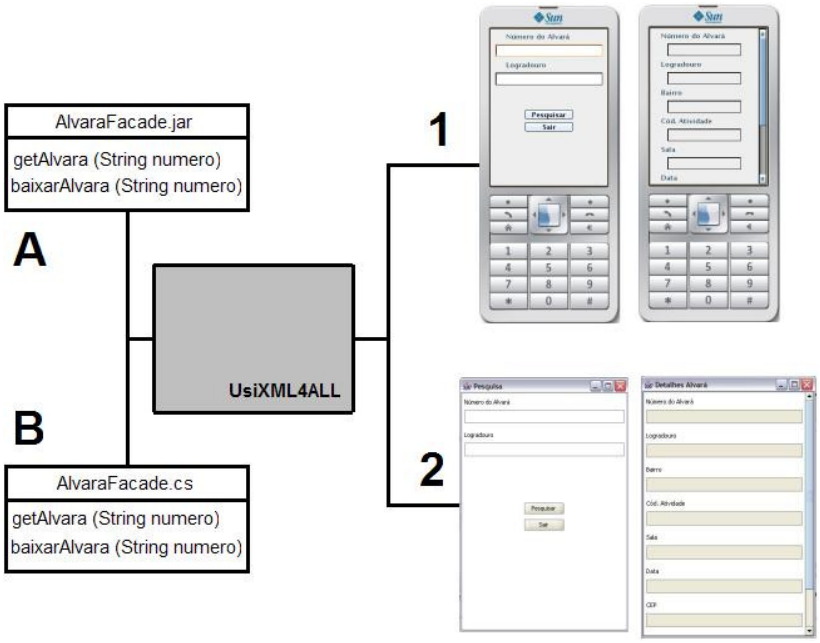

Figura 8. Estudo de caso.

Na prática, para criar as combinações descritas acima, o código da lógica de aplicação deve ser desenvolvido de forma a mesma interface que será visível para a camada de apresentação descrita em UsiXML. Nas situações apresentadas no estudo de caso, isso é realizado através da criação de um arquivo *.jar* para a linguagem Java e uma biblioteca *.dll* para .Net.

Com a lógica de aplicação definida, a IU é criada em UsiXML, declarando a estrutura da interface, assim como seu conteúdo e comportamento.

Para mudar a plataforma sendo utilizada, (A,1 para A,2 na Figura 8) basta renderizar a descrição da interface utilizando a versão da UsiXML4ALL para a plataforma desejada (*desktop* ou móvel).

De forma a modificar a linguagem de programação sendo utilizada (A,1 para B,1 na Figura 8) , basta mudar o parâmetro utilizado na renderização da IU, determinando o conector de lógica de aplicação desejado, desde que ambas as camadas de aplicação implementem a mesma interface.

6 Conclusões e Possibilidades Futuras

Este artigo descreveu uma abordagem prática para a adaptação de sistemas existentes para um ambiente multiplataforma e apresentou a ferramenta UsiXML4ALL, que realiza a interpretação e renderização de interfaces de usuário descritas na linguagem UsiXML para múltiplas plataformas, além de permitir conectar as interfaces de usuário renderizadas com códigos em diferentes linguagens de programação. Foi exemplificado o uso de UsiXML4ALL para adaptação de um sistema real utilizando-se dos conceitos de interfaces plásticas, com múltiplas opções de interface, criadas em uma descrição independente de tecnologia, utilizando-se da linguagem UsiXML.

A ferramenta apresentada tem potencial para permitir a difusão da utilização de LDIUs no desenvolvimento real de sistemas de software, tanto em novos projetos como em sistemas existentes. Com a sua utilização, permite-se que os aplicativos sejam desenvolvidos visando à execução em múltiplas plataformas, o que é um grande passo em direção ao desenvolvimento de sistemas de computação efetivamente ubíquos.

Como trabalho futuro planeja-se a evolução da ferramenta, visando à criação de interfaces de usuário para diferentes dispositivos, além de interfaces de usuário multimodais. Dentre os dispositivos/plataformas a serem atendidos estão telefones celulares, PDA's, interfaces *web* e até mesmo computadores XO do projeto OLPC [16]. O objetivo final é permitir a criação de interfaces de usuário baseadas em UsiXML para uma grande diversidade de dispositivos e plataformas, expandindo também o número de linguagens de programação passíveis de serem utilizadas. Em particular, nosso trabalho tem como visão a criação de uma ferramenta que possa ser utilizada no desenvolvimento de interfaces multiplataforma em uma grande diversidade de contextos de uso. Especificamente, se deseja investigar a criação de aplicativos no contexto de livros eletrônicos falados, dentro do projeto de pesquisa no qual este trabalho se insere.

Outra possibilidade interessante é o estudo da utilização de UsiXML4ALL na criação de interfaces multimodais, possibilitando não apenas apresentação independente de dispositivo, mas também novas modalidades de interação, como voz e gestos. Nesse caso, a diferença básica seria a adaptação do processo de renderização para essas modalidades de interação.

Agradecimentos. Este projeto é parcialmente financiado pelo CNPq (projeto LIFAPOR CNPq-GRICES).

Referências

1. Abrams, M., Phanouriou, C., Batongbacal, A. L., Williams,S. M., and Shuster, J. E. "UIML: An Appliance-IndependentXML User Interface Language." Proceedings of the 8th International WWW Conference. Toronto, Canada. 11-16 May 1999. Elsevier Science Publishers.
2. Ali, M.F., Pérez-Quiñones, M.A., Abrams, M., e Shell, E. Building Multi-Platform User Interfaces With UIML. Em Proceedings of 2002 International Workshop of Computer-Aided Design of User Interfaces: CADUI'2002. Valenciennes, França.
3. Azevedo, P., Merrick, R., Roberts, D. "OVID to AUIML -User Oriented Interface Modeling." http://math.uma.pt/tupis00/submissions/azevedoroberts/azevedoroberts.html
4. Berghe, Y. Etude et implémentation d'un générateur d'interfaces vectorielles à partir d'un langage de description d'interfaces utilisateur, M.Sc. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, Setembro de 2004.
5. Bouillon, L., Limbourg, Q., Vanderdonckt, J., Michotte, B., Reverse Engineering of Web Pages based on Derivations and Transformations, Proc. do 3o. Latin American Web Congress LA-Web'2005 (Buenos Aires, 31 de Outubro – 1o. de Novembro, 2005), IEEE Computer Society Press, Los Alamitos, 2005, pp. 3-13.
6. Calvary, G., Coutaz, J., and Thevenin, D. A Unifying Reference Framework for the Development of Plastic User Interfaces, Proc. of EHCI'2001, Springer-Verlag, 2001.
7. Calvary, G., Coutaz, J. Thevenin, D. Limbourg, Q., Bouillon, L. Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, Vol. 15, No. 3, Junho de 2003, pp. 289-308.
8. Consensus Project. http://www.consensus-online.org/
9. Coyette, A., Faulkner, S., Kolp, M., Limbourg, Q., SketchiXML: Towards a Multi-Agent Design Tool for Sketching User Interfaces Based on UsiXML. Em Proc. of Tamodia'2004.
10. Denis, V. Un pas vers le poste de travail unique: QTKiXML, un interpréteur d'interface utilisateur à partir de sa description, M.Sc. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, Setembro de 2005.
11. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M. and Trevisan, D. UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces. Em Proc. of the AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages" UIXML'04 (Gallipoli, 25 de Maio 2004). EDM-Luc, 55–62.
12. Lepreux, S., Vanderdonckt, J., Michotte, B. Visual Design of User Interfaces by (De)composition, Em Proc. of 13th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2006 (Dublin, 26-28 de Julho de 2006), G. Doherty and A. Blandford (eds.), Lecture Notes in Computer Science, Vol. 4323, Springer-Verlag, Berlin, 2006, pp. 157-170.
13. Luyten, K., Coninx, K. Uiml.net: an Open Uiml Renderer for the .Net Framework. Em CADUI'2004 long paper track, 2004.
14. Mori, G., Paternò, F., Santoro, C. Tool Support for Designing nomadic Applications. Em Proc. of 7th ACM Int.Conf. on Intelligent User Interfaces. ACM Press, Nova Iorque, 2003, pp. 141-148.
15. Ocal, K., Etude et développement d'un interpréteur UsiXML en Java Swing, Haute Ecole Rennequin, Liège, 2004.
16. One Laptop Per Child (OLPC). http://www.laptop.org/index.en_US.html
17. Paternò, F. Model-Based Design and Evaluation of Interactive Applications, Springer-Verlag, Berlin, 2000.

18. Phanouriou, C. UIML: A Device Independent User Interface Markup Language. Tese de Doutorado. Virginia Polytechnic Institute and State University. 2000. Blacksburg, Virginia.
19. Puerta, A.; Eisenstein, J. XIML: A Universal Language for User Interfaces. RedWhale Software. 2002.
20. Simon, R., Wegscheider, F., Tolar, K. Tool-supported single authoring for device independence and multimodality. Proceedings of the 7th international conference on Human computer interaction with mobile devices & services MobileHCI '05. Salzburg, Austria. Pages: 91 - 98 ISBN:1-59593-089-2
21. Vanderdonckt, J., A MDA-Compliant Environment for Developing User Interfaces of Information Systems, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), O. Pastor & J. Falcão e Cunha (eds.), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31