UsiXML4ALL – Uma Ferramenta para Criação de Aplicativos Multiplataforma

Francisco M Trindade
Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática,
Porto Alegre, Brasil, 91.501-970
fmtrindade@inf.ufrgs.br

e

Augusto A Moreira
Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática,
Porto Alegre, Brasil, 91.501-970
aamoreira@inf.ufrgs.br

e

Marcelo S Pimenta
Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática,
Porto Alegre, Brasil, 91.501-970
mpimenta@inf.ufrgs.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 multiplatform software. This article approaches the concept of plastic user interfaces, one proposal which aims at the solution of this problem, and presents the UsiXML4ALL application. This application has the objective of allowing the creation of multiplatform software, working with user interface rendering and logic application connection.

Keywords: Plastic User Interfaces, User Interface Description Language, UsiXML.

Resumo

Com o avanço da tecnologia, o desenvolvimento ad-hoc de software tem se mostrado insuficiente frente aos diferentes dispositivos computacionais existentes, tornando necessária a criação de novas técnicas para a implementação de soluções multiplataforma. Esse artigo aborda o conceito de interfaces plásticas, uma proposta para a solução desse problema, e apresenta a ferramenta UsiXML4ALL, a qual visa possibilitar a criação de aplicativos multiplataforma, tanto em relação à interface de usuário quanto à lógica de aplicação.

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

1 Introdução

O avanço da tecnologia dos equipamentos computacionais e das redes de comunicação e o 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 a comunidade de IHC [4], como o desenvolvimento e manutenção de versões de aplicações para diferentes dispositivos, a checagem de consistência entre essas versões para garantir a interoperabilidade entre os dispositivos e a capacidade de adaptação desses aplicativos a mudanças no contexto de execução. Para atender a essa demanda, o desenvolvimento *ad-hoc* de software, onde o programa é (re)desenvolvido especificamente para cada plataforma de uso, não é mais aceitável, tanto em termos do custo de desenvolvimento como de manutenção [3]. Dessa forma, muitos trabalhos de pesquisa vêm sendo realizados no sentido de permitir a criação de programas capazes de serem executados em múltiplos contextos de uso, com pouca ou nenhuma necessidade de alteração do código fonte desenvolvido.

Como forma de solucionar esse problema, foi proposto o conceito de Interfaces Plásticas, tecnologia que visa à criação de interfaces de usuário com capacidade de adaptação a diferentes contextos de uso. Resumidamente, essa abordagem visa a reutilização de descrições de interface, eliminando a necessidade da repetição 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 [5]. Sob a ótica da IHC, plasticidade é a capacidade de um sistema de suportar variações no contexto de uso, ainda assim preservando sua usabilidade. O conceito de contexto de uso pode ser interpretado aqui de diversas formas: desde atributos da plataforma de execução do aplicativo, tanto de hardware como de software, até as condições de ambiente existentes no momento de execução do programa, como temperatura, condições de luminosidade e perfil do usuário atual. Deve ser ressaltado que plasticidade não se trata apenas de condensar e expandir informações de acordo com o contexto de uso, mas também contrair e expandir o conjunto de tarefas disponíveis para o usuário de forma a preservar a usabilidade [4].

Para a realização de um projeto de software visando à reutilização de interfaces, as técnicas tradicionais de desenvolvimento de software interativo, onde cada interface é criada separadamente, não são aplicáveis. Ao contrário, uma descrição de mais alto nível é necessária de forma a possibilitar seu mapeamento para diferentes tipos de contexto. Essa descrição pode ser realizada de duas formas distintas:

- Descrição baseada em modelos (*model-based*) onde os modelos de interface de usuário têm o propósito de descrever abstratamente a interface de usuário a ser criada, e podem ser definidos como uma descrição formal, declarativa e livre de implementação da interface [9].

- Utilização de linguagens de descrição de interfaces de usuário [12], cujo objetivo é permitir a descrição em alto nível de interfaces de usuário de forma a permitir seu desenvolvimento independentemente da aplicação; esta descrição deve ser mapeada (na verdade, concretizada) em diferentes plataformas ou múltiplos contextos de uso, evitando o esforço duplicado de design.

Dentre as linguagens de descrição de interfaces de usuário existentes, pode-se destacar algumas que têm como objetivo serem utilizadas no desenvolvimento de interfaces de usuário multiplataforma: UIML [17], XIML [18] e UsiXML [10].

A UIML (*User Interface Markup Language*) é uma linguagem de marcação semelhante ao HTML com o principal objetivo de servir como uma forma canônica (única) para a descrição de interfaces e com a proposta de se tornar um padrão de descrição de interfaces [12]. A linguagem é baseada no modelo MIM, no qual a interface é separada em três componentes principais que a descrevem de forma canônica e independente de plataforma: Interface, Apresentação e Lógica.

A linguagem XIML (*Extensible Interface Markup Language*) foi padronizada pelo XIML Forum e se propõe a representar interfaces permitindo o suporte universal à funcionalidade através de todo o ciclo de vida de uma interface. As fases previstas neste ciclo são: design, desenvolvimento, operação administração, organização e avaliação. A principal dificuldade em se estudar a XIML é que praticamente todas as informações (incluindo a especificação da linguagem) são disponibilizadas apenas para membros do XML Forum. Para participar deste fórum, deve ser aceita uma licença que proíbe a ampla disseminação de qualquer informação contida no fórum.

A linguagem UsiXML (*User Interface Extensible Markup Language*) tem como objetivo descrever interfaces de usuário para múltiplos contextos de uso através de diferentes níveis de detalhamento e abstração. A UsiXML consiste de uma linguagem declarativa que captura a essência da uma interface independente das características físicas. Isto é feito através de uma descrição abstrata dos elementos que constituem uma interface. Estes elementos são agrupados em diversos modelos, como o de interface, de mapeamento, de domínio, de tarefa, de contexto, de transformação, entre outros.

Dentre as linguagens apresentadas acima, 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 linguagens de descrição de interfaces de usuário no desenvolvimento de software, se tornam 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 versão inicial da ferramenta UsiXML4ALL, a qual tem como proposta utilizar a linguagem de descrição de interfaces de usuário UsiXML para possibilitar a criação de aplicativos multiplataforma. A ferramenta apresentada realiza a reificação de descrições de interface concretas (CUI, conceito que será explicado adiante) em UsiXML, permitindo ao usuário a interconexão de diferentes plataformas de interface de usuário com diferentes linguagens de lógica de aplicação. Essa característica faz com que a ferramenta se torne útil tanto no desenvolvimento de novos aplicativos como na adaptação de aplicativos existentes para um contexto multiplataforma.

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 é introduzida na seção 4, e sua arquitetura descrita na seção 5. A seção 6 descreve a aplicação exemplo desenvolvida, e a seção 7 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 [10].

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).

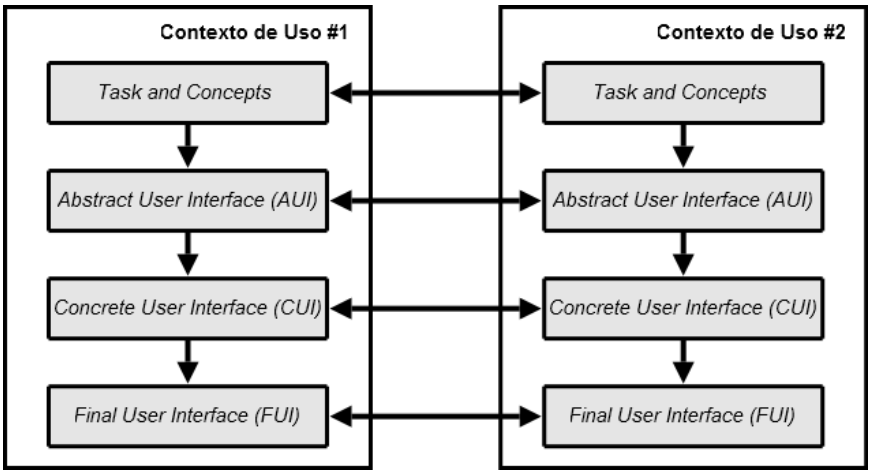

Figura 1. Os quatro níveis básicos do framework de referência Cameleon [10].

Os quatro níveis básicos existentes nesse framework são [10]:

- *Task and Concepts*: nível mais alto, onde as tarefas interativas (*tasks*) a serem realizadas são definidas 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. Pode-se considerar que uma definição de AUI é a expressão da reificação de um modelo de conceitos e tarefas, mas de maneira independente de qualquer modalidade de interação. A AUI define *containers* abstratos e componentes individuais através do agrupamento de tarefas de acordo com certos critérios (por ex: padrões estruturais de modelo de tarefa, análise de carga cognitiva, identificação de relacionamentos semânticos), um esquema de navegação entre *containers*, e seleciona componentes abstratos individuais para cada conceito de forma que sejam independentes de quaisquer ambientes de interação.

- *Concrete User Interface* (CUI): Definição que se caracteriza por ser independente do tipo de plataforma computacional ou conjunto de dispositivos de determinada plataforma. Uma CUI é composta de CIOs (*Concrete Interaction Obje*cts), os quais são uma representação abstrata de elementos encontrados em bibliotecas gráficas (Java AWT/Swing, por exemplo). Portanto, uma CUI consiste de uma decomposição hierárquica de CIOs de um determinado conjunto de componentes da plataforma-alvo em um dado contexto de uso. A CUI é uma representação abstrata de uma FUI (ver adiante) de forma que seja independente de qualquer plataforma computacional ou das linguagens de programação usadas para desenvolver IUs. Entretanto, CUIs dependem de ambiente de interação, ou seja, uma instância de uma CUI endereça um único ambiente de interação por vez. Os artefatos de uma CUI, geralmente, são especificações em XML da hierarquia de CIOs que compõe a IU.

- *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). Os artefatos de uma FUI são os códigos-fonte de linguagens que implementam a IU - tais como Java e HTML - e os códigos-objeto (executáveis) que renderizam a IU.

O *Cameleon Reference Framework* prevê que os passos de desenvolvimento possam ser obtidos através de transformações verticais e horizontais. As transformações verticais definem os processos para a transformação de um passo de desenvolvimento - um modelo de IU - em outro mais concreto (processo de reificação), ou em outro mais abstrato (processo de abstração). As transformações horizontais definem processos para a obtenção de um modelo de IU a partir de outro de mesmo nível de reificação, mas em um contexto de uso diferente do modelo original (processo de tradução).

Para representar os níveis de abstração do framework de referência, a UsiXML define os conceitos que serão apresentados a seguir.

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) [16]. 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.

O modelo de domínio (*Domain Model*) descreve os conceitos do mundo real e suas associações conforme compreendido pelo usuário. A UsiXML descreve os conceitos contidos no modelo de domínio da mesma forma que um diagrama de classes UML. Entre os conceitos existentes estão classes, atributos, métodos e relações entre objetos do domínio.

De forma a permitir a definição independente do conteúdo presente na interface, o modelo de recursos (*Resource Model*) permite que se especifique o conteúdo externo, como mensagens e imagens, por exemplo, de forma separada da definição dos componentes da interface. Desse modo, a adaptação da interface de usuário para diferentes contextos de uso passa a ser facilitada, sendo necessária a modificação apenas do modelo de recursos.

O modelo de contexto (*Context Model*) define todas as entidades que podem influenciar na execução de uma tarefa interativa pelo usuário com a interface de usuário sendo especificada. O modelo de contexto deve capturar qualquer atributo relevante do contexto de uso onde a aplicação se encontra, sendo composto por: modelo de usuário, o qual classifica os usuários do aplicativo em diferentes perfis (estereótipos), modelo de plataforma, o qual captura informações relevantes da combinação de software-hardware da plataforma sendo utilizada, e modelo de ambiente, o qual define propriedades de interesse do ambiente físico onde a aplicação está sendo executada.

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.

A definição de CUI em UsiXML é caracterizada por representar um conjunto de *Concrete Interaction Objects* (CIO), cada qual com a informação de suas características. O layout de uma CUI é definido sem qualquer informação de posição absoluta, mas sim com definições de elementos do tipo *box*, os quais agrupam componentes que possuam relações de posição na interface.

Além da definição dos componentes que compõem a interface de usuário, a CUI possui um mecanismo capaz de estabelecer o comportamento dinâmico da interface, incluindo uma linguagem de definição de navegação e uma linguagem para definição da relação entre eventos e ações (*event/action*) [10].

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 [10].

Em adição aos modelos citados acima, a especificação da UsiXML também cobre aspectos dinâmicos do ciclo de vida de desenvolvimento de interface de usuário, como engenharia reversa, adaptação de contexto de uso e especificação de diálogo mas estes aspectos não serão detalhados neste artigo.

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 [7], 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 [11] 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.

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 [8] 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 [2], 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 linguagens de descrição de interfaces de usuário 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 [1] (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.

4 UsiXML4ALL

As ferramentas de renderização mostradas na seção anterior possuem algumas limitações:

- Adotam uma única linguagem específica na qual a saída da ferramenta é gerada;
- 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.

Essas características dificultam o uso desses aplicativos no desenvolvimento prático de interfaces de usuário, pois o fato de limitarem a sua utilização a uma determinada tecnologia restringe o número de aplicações que poderão se beneficiar das mesmas, além de dificultar sua utilização com códigos-fontes já existentes, no caso em que esses foram desenvolvidos, por exemplo, em uma linguagem de programação diferente.

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 possibilita a renderização de interfaces de usuário para múltiplas plataformas. Como diferencial UsiXML4ALL é 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.

A intenção é que UsiXML4ALL atenda a um número maior de situações em que possa ser utilizada, tanto no desenvolvimento de novos aplicativos para múltiplas plataformas, como na adaptação de aplicativos existentes para um ambiente de execução multiplataforma.

Um benefício disto pode ser constatado no exemplo fornecido anteriormente. No caso de um desenvolvedor que possua seu código já desenvolvido em uma determinada linguagem de programação (p.ex. Java), ele poderia utilizar a UsiXML4ALL para portar seu aplicativo para múltiplas plataformas (p.ex. *desktop*, *web*, etc.), bastando que existisse um conector existente para sua linguagem de programação. Desse modo, com apenas a criação das interfaces necessárias em UsiXML, o aplicativo já estaria pronto para ser disponibilizado em múltiplas plataformas.

Além disso, como a ferramenta é extensível, no caso de a linguagem de programação desejada não estar relacionada entre as atendidas no momento, seria 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.

5 Arquitetura da Ferramenta

Nesta seção é apresentada a arquitetura da ferramenta desenvolvida, esclarecendo os blocos que a compõem e o funcionamento de cada um dos mesmos.

5.1 Visão Geral da Arquitetura

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 3). 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.

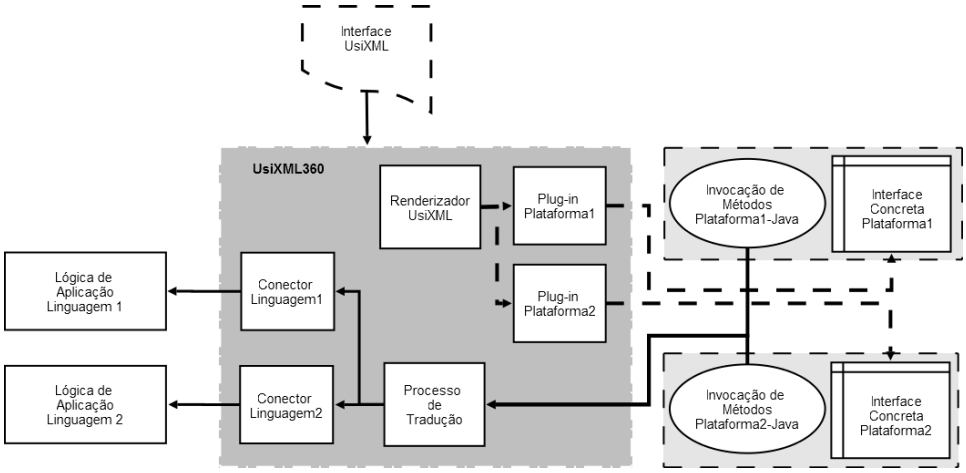

Figura 3. Arquitetura geral da ferramenta UsiXML4ALL.

Para realizar a etapa de renderização, a UsiXML4ALL recebe como entrada uma arquivo contendo uma definição de interface segundo a especificação UsiXML (Interface UsiXML na Figura 3) 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 a plataforma-alvo desejada (*Plug-in* Plataforma 2 na Figura 3, por exemplo). O *plug-in* de renderização é responsável então por criar a interface final correspondente (Interface Concreta Plataforma 2 na Figura 3) à descrição da interface obtida como entrada, sendo que as invocações de métodos existentes nessa descrição são direcionadas para o núcleo de controle do aplicativo (Processo de Tradução na Figura 3).

De modo a realizar a conexão com a lógica de aplicação em múltiplas linguagens de programação, o núcleo de controle (Processo de Tradução na Figura 3) 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 3, por exemplo), a qual converte novamente a chamada, fazendo-o chegar até o código sendo executado (Lógica de Aplicação Linguagem 1 na Figura 3). No caso da existência de valores de retorno, o procedimento inverso é executado.

5.2 Núcleo de Controle

O núcleo de controle da UsiXML4ALL é responsável por duas tarefas principais.

A primeira se refere a receber a entrada do arquivo contendo uma definição de interface segundo a especificação UsiXML. Esse arquivo é então validado a partir do esquema XML da linguagem e então repassado, sem nenhuma modificação, para o *plug-in* de renderização de interfaces sendo utilizado.

Após a renderização da interface final pelo *plug-in* de renderização, o núcleo de controle mantém ativo um processo de tradução de eventos, o qual recebe as invocações de métodos da lógica de aplicação

pela interface. Nesse momento a chamada recebida é então transformada em um objeto que descreve o método sendo chamado, através de seu nome e parâmetros.

Esse objeto é transferido para o conector de lógica de aplicação correspondente, completando a invocação do método desejado.

5.3 *Plug-ins* de renderização

O *plug-in* de renderização realiza a transformação da descrição concreta da interface em UsiXML para uma interface final na sua respectiva plataforma de execução.

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.

5.4 Conectores de Lógica de Aplicação

Os conectores de lógica de aplicação são responsáveis por receber a descrição do método a ser invocado, traduzi-lo para a plataforma da lógica de aplicação e completar a chamada dos mesmos.

Para permitir a realização dessa 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.

6 Exemplo de Aplicação: Calculadora Multiplataforma

De forma a validar a primeira implementação da ferramenta UsiXML4ALL, foi desenvolvida uma aplicação exemplo, a qual consiste em uma calculadora multiplataforma.

O objetivo desse exemplo é que a calculadora seja renderizada utilizando duas plataformas diferentes, Java Swing e *WinForms*. Além disso, deve ser possível utilizar qualquer uma das interfaces de usuário renderizadas com duas opções de lógica de aplicação, também desenvolvidas nas linguagens C# e Java.

Para isso foram criadas as lógicas de aplicação em Java e C#. Ambas as bibliotecas contém apenas um método público, *buttonPressed*(), o qual recebe o botão pressionado pelo usuário e retorna um valor do tipo String, o qual corresponde ao valor que deve ser exibido pelo visor da calculadora.

Para descrever a interface, foi montado um arquivo no formato UsiXML, o qual contém a definição de uma CUI, com os componentes necessários para a calculadora, e esse foi fornecido como entrada para a ferramenta de renderização. Uma amostra do código UsiXML de definição da interface da calculadora é apresentada na figura abaixo (Figura 4).

```
<cuiModel id="5-cui_20" name="5-cui">
        <window id="window_1" name="window_1"
            width="192" height="200">
            <flowBox id="flow_box_2" name="flow_box_2" alignment="left">
                <inputText id="input_1"
                        name="input_1" isVisible="true"
                    isEnabled="true" textColor="#000000"
                    maxLength="50" numberOfColumns="15" isEditable="true"/>
                <button id="button_1"
                        content="/uiModel/resourceModel/cioRef[@cioId='button_1']/resource/@content"
                        name="button_1" isVisible="true"
                        isEnabled="true" textColor="#000000"/>
                <button id="button_2"
                        content="/uiModel/resourceModel/cioRef[@cioId='button_2']/resource/@content"
                        name="button_2" isVisible="true"
                        isEnabled="true" textColor="#000000"/>
                <button id="button_3"
                        content="/uiModel/resourceModel/cioRef[@cioId='button_3']/resource/@content"
                        name="button_3" isVisible="true"
                        isEnabled="true" textColor="#000000"/>
                ...
            </flowbox>
        </window>
</cuiModel>
```

Figura 4. Amostra do código UsiXML de definição da interface da calculadora.

Além dos elementos existentes na interface projetada, a definição UsiXML da mesma também contém a definição do conteúdo de cada elemento, assim como as ações a serem executadas quando da ocorrência de eventos específicos durante a execução da interface.

No contexto da aplicação exemplo, os eventos mapeados se referem à operação de *click* realizada pelo usuário em alguns dos botões da calculadora, os quais invocam o método *buttonPressed*() na lógica de aplicação sendo executada, recebendo como retorno o valor a ser exibido no display da calculadora.

No entanto, para manter a independência de linguagem da lógica de aplicação sendo utilizada, a interface concreta direciona a invocação de seus métodos para a UsiXML4ALL, informando o nome e parâmetros do método a ser invocado. Essa definição é então traduzida para a lógica de aplicação sendo executada, e então o método é invocado. A figura abaixo (Figura 5) mostra o trecho de código executado pela interface para invocar o método de maneira independente da lógica de aplicação.

```
logicConector.prepareMethodClass("br.inf.ufrgs.calc.logic.CalculatorLogic");
logicConector.prepareMethodName("buttonPressed");
logicConector.prepareMethodIntegerParameter(ButtonType.BUTTON_1.ordinal(), 0);

logicConector.prepareMethodStringReturnParameter();
logicConector.executeMethod();
```

Figura 5. Trecho de código executado para invocação de método da lógica de aplicação.

Dessa maneira, é possível que qualquer uma das interfaces concretas possíveis de serem criadas com a ferramenta, em Java e C#, realizem a invocação de métodos em ambas as lógicas de aplicação (Figura 6). Tendo como exemplo a figura abaixo, poderiam ser criadas 4 combinações de interface de usuário – lógica de aplicação (A,1;A,2;B,1;B,2).

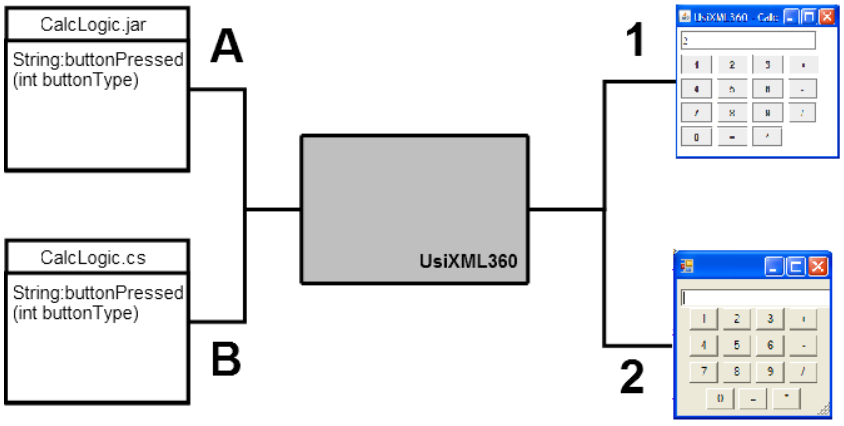

Figura 6. Ciclo completo da aplicação desenvolvida como exemplo.

Dessa forma, apesar do caráter demonstrativo da aplicação, o objetivo de permitir a renderização de uma descrição de interface de usuário UsiXML para múltiplas plataformas foi alcançado, possuindo ainda como vantagem a possibilidade de conectar a interface gerada com lógicas de aplicação em diferentes linguagens de programação.

7 Conclusão e Possibilidades Futuras

Este artigo descreveu uma abordagem prática para o desenvolvimento de aplicativos multiplataformas e apresentou a primeira versão da 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 o desenvolvimento de um aplicativo multiplataforma 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 linguagens de descrição de interfaces de usuário no desenvolvimento real de sistemas de software, tanto em novos projetos como em sistemas legados. 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. Também se pretende explorar mais profundamente as possibilidades de desenvolvimento de Interfaces Plásticas com UsiXML, principalmente em aplicativos multiplataforma, no contexto de livros eletrônicos falados, dentro do projeto de pesquisa onde este trabalho se insere.

Agradecimentos

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

References

[1] 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.

[2] 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.

[3] Bouillon, L., Vanderdonckt, J., Eisenstein, J. Model-Based Approaches to Reengineering Web Pages, International Workshop on Task Model and Diagrams for user interface design, TAMODIA 2002.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Eisenstein, J., Vanderdonckt, J., A. Puerta. Adapting to Mobile Contexts with User-Interface Modeling. Em Proceedings of Workshop on Mobile Computing Systems and Applications 2000. Monterey, CA: IEEE Press, 7-8 de Dezembro de 2000.

[10] 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.

[11] 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.

[12] Luyten, K., Coninx, K. Uiml.net: an Open Uiml Renderer for the .Net Framework. Em CADUI'2004 long paper track, 2004.

[13] Luyten, K., Thys, K., Vermeulen, J., e Coninx, K. A Generic Approach for Multi-Device User Interface Rendering with UIML. Em 6th International Conference on Computer-Aided Design of User Interfaces (CADUI'2006), Bucareste, Romênia.

[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] Paternò, F. Model-Based Design and Evaluation of Interactive Applications, Springer-Verlag, Berlin, 2000.

[17] Phanouriou, C. UIML: A Device Independent User Interface Markup Language. Tese de Doutorado. Virginia Polytechnic Institute and State University. 2000. Blacksburg, Virginia.

[18] Puerta, A.; Eisenstein, J. XIML: A Universal Language for User Interfaces. RedWhale Software. 2002.