Estudo comparativo de diferentes linguagens de interfaces baseadas em XML

Ricardo Alexandre G. C. Martins, José Carlos Ramalho, and Pedro Rangel Henriques

\{ram,jcr,prh\}@di.uminho.pt
Departamento de Informática — Universidade do Minho

Resumo A alguns anos, houve um grande esforço por parte do meio acadêmico e industrial para padronizar a representação de diferentes tipos de dados, de forma a facilitar a interoperabilidade de aplicações. Desse esforço surgiu o XML, e com ele foi possível representar dados de uma maneira estruturada e, com isso, obter diferentes resultados vindos de uma mesma fonte de dados.
Atualmente, o desafio é estabelecer uma forma única de representar a interface dos dados, de forma que a programação de uma interface para um web site seja igual a interface de um software convencional.
No momento, existem algumas propostas de linguagens baseadas em XML que visam solucionar este tipo de problema, de modo que um software completo possa ser gerado dinamicamente. Este artigo apresenta uma comparação entre três diferentes linguagens para definição de interfaces, a saber: XUL, UIML e MXML. Serão analisadas as vantagens e desvantagens de cada linguagem através de diferentes parâmetros de comparação, de modo a determinar a linguagem que oferece mais recursos para o utilizador.

1 Introdução

Durante os últimos cinco anos, temos visto um grande avanço no intercâmbio de informações e documentos entre aplicações. Devido a iniciativas para o estabelecimento de padrões, e suas respectivas utilizações por parte dos fabricantes de software, hoje dispomos de uma grande variedade de opções para trocar informações entre programas, servidores e o que mais precisar.

Entretanto, a situação era semelhante ao que aconteceu aos PCs duas décadas atrás: vários tipos de hardware foram desenvolvidos, e cada um com sua própria *application programming interface* (API). Com isso, surgiu um problema: os desenvolvedores de software não poderiam suportar novo hardware automaticamente assim que novos dispositivos entrassem no mercado.

Porém, após o estabelecimento do XML [5] como padrão pelo W3C (*World Wide Web Consortium*), surgiram novas tecnologias baseadas nesse padrão que facilitam a conversão de formatos de dados, como o XSL, a representação gráfica em imagens (SVG [6]) e alguns outros que facilitam a comunicação entre aplicações, como os Web Services [7].

Com todas essas tecnologias à disposição dos utilizadores, o novo desafio que se surge é desenvolver uma maneira única para representar as informações, seja em software tradicional, web ou qualquer outro meio.

Atualmente existem algumas linguagens baseadas em XML que se propoem a servir como uma forma canônica de programação, de modo que um mesmo arquivo contendo o código-fonte, represente simultaneamente a interface de dados para um software e a interface para um website.

Este artigo tem como objetivo estudar três propostas de linguagens de especificação de interfaces, de modo a determinar quais são as linguagens mais apropriadas em determinadas situações, assim como as mais simples de serem implementadas e as mais "leves"em processamento.

Na seção 2 serão apresentadas as linguagens de especificação de interfaces, assim como alguns exemplos de sua programação. Na seção 3 serão apresentados os parâmetros de comparação e os resultados das análises realizadas e por fim, na seção 4, será apresentada a conclusão do estudo.

2 Linguagens

Quando falamos de linguagens de interface, devemos ter em mente que uma das principais vantagens dessas linguagens sobre outras existentes é com relação a portabilidade.

Muitas aplicações necessitam serem desenvolvidas utilizando características de uma plataforma específica, o que torna a construção de software cross-platform caro e lento. No momento isto pode não ser importante, mas os utilizadores podem querer utilizar uma aplicação em outro dispositivo como um PDA.

Uma vez que as linguagens de interface funcionam sob um interpretador (similarmente a Java, onde uma máquina virtual interpreta o programa), diferentes dispositivos e sistemas podem tirar proveito de um mesmo arquivo, e com isso garantimos que o desenvolvimento de uma interface torna-se idependente de hardware ou software. PDA's, computadores portáteis, PC's possuem a capacidade de interpretar as interfaces, necessitando apenas a presença do interpretador ou "máquina virtual"correspondente.

2.1 XUL

XUL [2] é uma tecnologia baseada no padrão XML para expressar a parte interface de um software. Ela foi desenvolvida para expressar interfaces para diversas aplicações como navegadores web, clientes de e-mail, calendários, calculadoras, editores HTML, debuggers e qualquer ambiente desktop. Toda a implementação das características da XUL estão contidas dentro da plataforma Mozilla, que são o motor executável e as bibliotecas que acompanham todo produto baseado no Mozilla. Devido a isso qualquer documento XUL será exibido corretamente nos sistemas Microsoft Windows, Macintosh e Linux.

A XUL foi desenvolvida pela Netscape e agora tem sua continuidade garantida pela Mozilla Foundation. Apesar de utilizar alguns padrões do World Wide

Web Consortium (W3C) em seu design, a XUL não é um padrão oficial. Entretanto, todos os documentos XUL são documentos XML válidos e suportam XML Namespaces, DTDs, CSS's assim como JavaScript. Assim como os documentos HTML, os documentos XUL podem ser copiados pela internet e exibidos ou instalados localmente se necessário.

Elementos suportados A XUL, como uma linguagem de programação de interfaces, suporta (e fornece) a programação de alguns elementos e seus atributos de interfaces visuais. São eles:

- elementos simples - janelas, botões, labels, imagens, controles de entrada, controles de listas;
- elementos comuns - barras de progresso, scroll bars, pilhas e cartões, posicionamento de pilha, tabboxes, barras de ferramentas, painéis e splitters;
- caixas - modelos, posicionamento, detalhes, grupos e adição de mais elementos;
- menus - barras de menus simples, características, menus popup e menus deslizantes;
- tabelas, árvores e RDF - características dos listboxes, árvores, características das árvores, templates e fontes de dados RDF;
- eventos e scripts - event handlers, DOM, atalhos de teclado, seleção e focus, seleção de árvores e customização de visualização de árvores;
- Cross-platform Component Object Model(XPCOM) - interfaces XPCOM, manipulação de fontes de dados RDF e drag and drop;
- skins e localizações - adição de stylesheets, definição de estilos, criação de skins e localização de arquivos;
- bindings - adição de propriedades, métodos e event handlers;

Codificação A codificação de um documento XUL é bastante simples, parecendo bastante com um documento HTML. Sua sintaxe é baseada na utilização de etiquetas (tags) que permitem especificar a semântica e o valor do elemento.

Um exemplo de um documento XUL pode ser visto conforme abaixo:

```
<?xml version="1.0"?>

<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window id="example-window" title="Example 8.2.1"
   xmlns:html="http://www.w3.org/1999/xhtml"
   xmlns="http://www.mozilla.org/key...ekeeper/there.is.only.xul">
<toolbox>
 <menubar id="windowlist-menubar">
  <menu label="Window">
   <menupopup id="window-menu" datasources="rdf:window-mediator">
    <template>
```

```
     <rule>
      <menuitem uri="rdf:*"
              label="rdf:http://home.netscape.com/NC-rdf#Name"/>
     </rule>
    </template>
   </menupopup>
  </menu>
 </menubar>
</toolbox>
</window>
```

2.2 UIML

Confome [3], a UIML "é uma meta-linguagem derivada do XML para descrição de interfaces de utilizadores". O objetivo principal dessa linguagem é servir como linguagem de programação para diferentes dispositivos, de modo que um único documento possa ser lido em qualquer hardware.

Por se tratar de um documento baseado em XML, é possível a transformação para outras linguagens através da aplicação de stylesheets. Para a visualização da interface, é necessário a utilização de um *renderer* específico para cada sistema operativo e tipo de interface (HTML, WAP, Java, etc) e também é necessário o desenvolvimento em separado da interface para cada dispositivo.

Em UIML, uma interface de utilizador é um conjunto de elementos de interface com o qual o utilizador interage. Esses elementos podem ser organizados diferentemente para diferentes categorias de utilizadores e aplicações. Cada elemento de interface possui dados (por exemplo, texto, sons, imagens) que são utilizados para realizar a comunicação com o utilizador e também podem receber informações sobre as interações do utilizador em runtime.

Codificação

```
<?xml version="1.0"?>

<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
"http://uiml.org/dtds/UIML3_0a.dtd">

<uiml>
    <interface>
        <structure>
            <part id="TopHello">
                <part id="hello" class="helloC"/>
            </part>
        </structure>
        <style>
```

```
            <property part-name="TopHello" name="rendering">
                Container
            </property>
            <property part-name="TopHello" name="content">
                Hello
            </property>
            <property part-class="helloC" name="rendering">
                String
            </property>
            <property part-name="hello" name="content">
                Hello World!
            </property>
        </style>
    </interface>
    <peers> ... </peers>
</uiml>
```

2.3 MXML

O MXML [1] é uma linguagem de programação baseada no padrão XML desenvolvida pela Macromedia para servir como linguagem de programação de interfaces para seu novo servidor de aplicações, batizado **Flex** [4] (atualmente em fase de beta teste), que funciona sob um servidor de aplicações J2EE, como Macromedia JRun, IBM Websphere, BEA WebLogic ou Apache Tomcat.

Com o MXML, é possível declarar o layout de uma interface de uma maneira mais estruturada e menos ambígua que o HTML. Além disso, o MXML inclui um conjunto de tags específicas diferentes do HTML, como DataGrid, Tree, TabNavigator, Accordion, e Menu e também a possibilidade da criação de novos componentes. Entretanto, a diferença mais significativa do MXML para o HTML é que as interfaces são renderizadas pelo Flash Player, o que fornece aos utilizadores mais possibilidades do que o HTML.

Para a programação de respostas a eventos do utilizador, é possível a criação de funções utilizando ActionScript, que é uma linguagem baseada no padrão ECMA-262.

Tecnologias suportadas O Macromedia Flex foi desenvolvido para suportar diferentes tecnologias para a adição de funcionalidades. De acordo com o objetivo pretendido, é possível utilizar tecnologias diferentes para enriquecer o conteúdo de um documento MXML. As finalidades e as tecnologias são as seguintes:

- Acesso a dados - utilização de Webservices;
- Design - utilização de Cacade Style Sheets (CSS);

Codificação A codificação de um documento MXML é bastante simples, parecendo bastante com um documento HTML. Sua sintaxe é baseada na utilização de etiquetas (tags) que permitem especificar a semântica e o valor do elemento.

Um exemplo de um documento MXML pode ser visto conforme abaixo:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml">
    <mx:script>
        function copy()
        {
            destination.text=source.text
        }
    </mx:script>
    <mx:TextInput id="source" width="100"/>
    <mx:Button label="Copy" click="copy()"/>
    <mx:TextInput id="destination" width="100"/>
</mx:Application>
```

3 Estudos comparativos

3.1 Parâmetros de comparação

Para estabelecer as vantagens e desvantagens de uma linguagem perante a outra, foram estabelecidos alguns parâmetros de comparação, a saber:

- Facilidade de programação;
- Extensibilidade;
- Tempo de Resposta;
- Portabilidade;
- Integração com outras tecnologias;
- Acesso a dados remotos.

3.2 Resultados

1. **Facilidade de programação -** devido ao fato de que todas as linguagens são baseadas no formato XML, a programação dos documentos nas diferentes linguagens é um trabalho bastante simples, podendo ser facilmente substituído por um editor;
2. **Extensibilidade -** é a capacidade de criar novos componentes necessários no desenvolvimento da interface. As linguagens XUL e MXML fornecem suporte à extensibilidade através da capacidade de criação de novos elementos;
3. **Tempo de Resposta -** em todos os casos o tempo de resposta para a execução das interfaces foi baixo;

4. **Portabilidade -** é a capacidade de um documento poder ser executado em diferentes sistemas operativos. Para a linguagem XUL basta a utilização do motor Gecko, disponível para os sistemas operativos Linux, Windows e Machintosh. Para executar documentos UIML é necessário a instalação de um renderer específico para cada formato de saída (software tradicional, web, wap, etc). Os documentos MXML necessitam apenas o Flash Player instalado, o que os tornam acessíveis em qualquer sistema operativo atual ;
5. **Integração com outras tecnologias -** em todas as linguagens existe suporte a CSS e linguagens de script para o tratamento de eventos realizados pelo utilizador;
6. **Acesso a dados remotos -** de todos os casos, somente o MXML possui suporte nativo a acesso de dados remotos, através de Webservices;

Parâmetros de comparação			
	XUL	**UIML**	**MXML**
Facilidade de programação	alta	alta	alta
Extensibilidade	sim	não	sim
Tempo de Resposta	baixo	baixo	baixo
Portabilidade	alta	média	alta
Integração com outras tecnologias	sim	sim	sim
Acesso a dados remotos	não	não	sim

Tabela 1. Tabela de comparação das linguagens

4 Conclusão

As linguagens de interface é um exemplo de tecnologia que visa estabelecer um formato único para descrever uma sitaxe de programação para diferentes tipos de hardware e software.

Devido às diferentes características de hardware e software existentes entre os dispositivos, as diferentes linguagens existentes direcionam-se para diferentes públicos, embora todas tenham o mesmo objetivo final: servir como uma base canônica para a programação de interfaces.

Das linguagens estudadas, pode-se dizer que a linguagem UIML encontra-se em um ível inferior às demais estudadas, pois as demais tecnologias possuem a capacidade de criar novos componentes e uma melhor portabilidade, pois apenas necessitam de um único software para executar seus respectivos documentos para os mais diferentes tipos de interface (web, wap, software tradicional, etc).

Diferentemente da demais tecnologias estudadas, o MXML apresenta-se como uma grande aposta da Macromedia para uma completa integração de dados, conseguindo trabalhar de maneira eficiente tanto o conteúdo (através do acesso

a dados remotos por Webservices) quanto a parte visual (utilização de CSS, ActionScript e Flash).

Referências

1. Christophe Coenraets. An overview of mxml, the macromedia flex markup language. http://www.macromedia.com/devnet/flex/articles/paradigm.html/.
2. Neil Deakin. Xul tutorial. http://www.xulplanet.com/tutorials/xultu/, Fevereiro, 2004.
3. Harmonia. Tutorial booklet. http://www.uiml.org/tutorials/uiml1/uiml_v10_tutorial.pdf/, Dezembro, 1997.
4. Macromedia. Macromedia flex: the presentation tier slution for enterprise rich internet applications. http://www.macromedia.com/software/flex/whitepapers/pdf/flex_tech_wp.pdf/, Novembro, 2003.
5. W3C. Extensible markup language (xml). http://www.w3.org/XML/.
6. W3C. Scalable vector graphics (svg) 1.1 specification. http://www.w3.org/TR/SVG11/, Janeiro, 16, 2003.
7. W3C. Web services architecture requirements. http://www.w3.org/TR/wsareqs, Novembro, 14, 2002.