FlexiXML
Um animador de modelos UsiXML

Sandrine Alves Mendes José Creissac Campos
Departamento de Informática/CCTC, Universidade do Minho
sandrine.mendes@gmail.com, jose.campos@di.uminho.pt

Resumo
Uma parte considerável do desenvolvimento de software é dedicada à camada de interacção com o utilizador. Face à complexidade inerente ao desenvolvimento desta camada, é importante possibilitar uma análise tão cedo quanto possível dos conceitos e ideias em desenvolvimento para uma dada interface. O desenvolvimento baseado em modelos fornece uma solução para este problema ao facilitar a prototipagem de interfaces a partir dos modelos desenvolvidos. Este artigo descreve uma abordagem à prototipagem de interfaces e apresenta a primeira versão da ferramenta FlexiXML que realiza a interpretação e animação de interfaces descritas em UsiXML.

Palavras chave
Prototipagem, linguagens de modelação de interfaces (UIDLs), UsiXML.

1. INTRODUÇÃO

A construção de interfaces com o utilizador é um processo complexo, dado o conjunto de aspectos a considerar: seja a complexidade/diversidade dos ambientes de desenvolvimento e execução, seja a quantidade de conhecimentos de engenharia de software e de engenharia da usabilidade necessários. Estas dificuldades aumentam quando a mesma interface gráfica necessita de estar disponível em múltiplos contextos. A noção de contexto caracteriza diferentes perfis de utilizador (com diferentes preferências, idiomas ou mesmo com diferentes capacidades cognitivas e físicas) e diferentes plataformas e dispositivos (telemóveis, PDA's, WebTV, etc.). Torna-se difícil disponibilizar uma interface em múltiplos contextos sem criar várias versões da mesma, consoante o contexto a ser utilizado.

Paralelamente a este problema verifica-se o facto de nem sempre as interfaces satisfazerem as reais necessidades dos utilizadores. Para minimizar este problema, e detectar possíveis erros o mais cedo possível no processo de desenvolvimento, os utilizadores finais devem participar no processo de desenvolvimento. O objectivo é que utilizadores finais possam acompanhar todo o processo, contribuindo com as suas perspectivas sobre a melhor forma de o software dar resposta às suas necessidades.

O desenvolvimento baseado em modelos fornece uma solução para estes problemas [Paternò99]. Neste paradigma, são criados vários modelos declarativos que descrevem as tarefas que o utilizador pode efectuar no sistema, as capacidades funcionais, o estilo/requisitos da interface, as características/preferências do utilizador e as técnicas de I/O suportadas pela plataforma utilizada.

Uma abordagem baseada em modelos favorece um desenvolvimento mais sustentado. Em particular permite a construção imediata de protótipos e/ou a animação dos modelos [daSilva00]. Deste modo, auxilia os profissionais informáticos no processo de concepção e desenvolvimento e facilita a participação dos utilizadores nesse processo.

No seguimento da problemática referida anteriormente este artigo descreve um animador de modelos (FlexiXML) capaz de realizar a prototipagem da interface com o utilizador a fim de permitir testes com a mesma. Não se pretende criar a interface final, definitiva, mas sim gerar automaticamente versões da interface que possam ser testadas e iterativamente adaptadas aos utilizadores. A ferramenta pode ser considerada como um "UI runtime system", de acordo com a classificação de sistemas de implementação de interfaces utilizada em [daSilva00].

1.1 Motivação e Objectivos

A maioria das falhas detectadas em interfaces é decorrente das deficiências de comunicação entre os profissionais de informática que as desenvolvem e os seus utilizadores finais. É, assim, importante que utilizadores possam acompanhar o processo de desenvolvimento para que possíveis problemas possam ser detectados atempadamente. É neste contexto que o FlexiXML surge, como uma ferramenta de prototipagem de interfaces gráficas.

A linguagem base para este projecto é UsiXML [UsiXML09], uma linguagem declarativa que permite modelar os aspectos essenciais de uma interface. A escolha da UsiXML deveu-se ao facto de se tratar de uma linguagem muito abrangente, que permite modelar

diferentes aspectos da interface: o seu aspecto gráfico, a interacção com os utilizadores, contexto de execução, entre outros. Embora nesta versão do projecto apenas sejam utilizadas algumas das potencialidades da linguagem, o facto de esta abordar diferentes áreas permite que o projecto possa evoluir para focar outros aspectos (p.e., interface multimodais).

O FlexiXML surge no seguimento de uma ferramenta já existente: FlashiXML [Youri04]. A escolha deste interpretador como ponto de partida do projecto deveu-se ao facto de se tratar de uma ferramenta interessante, desde o ponto de vista conceptual ao tecnológico. O FlashiXML está desenvolvido em Flash, o que traz várias vantagens das quais podemos salientar: as capacidades da plataforma a nível gráfico, disponibilização simples via Web e a utilização de um formato vectorial (o que permite que as aplicações possam ser redimensionadas e o seu conteúdo adaptado às novas dimensões).

Neste projecto, embora se utilizem alguns conceitos do FlashiXML, nomeadamente a nível das estruturas de dados utilizada, pretendeu-se criar uma versão mais actualizada e com funcionalidades adicionais. Os principais objectivos a atingir eram:

- Interpretação da versão mais recente de UsiXML. Actualmente esta encontra-se na versão 1.8 e o FlashiXML é compatível com a versão 1.4.6.;
- Implementação da ferramenta numa versão mais recente da tecnologia (Air/Flex3/ActionScript3). A utilização de Air, permite que o FlexiXML seja independente da plataforma computacional em que é executado. Em termos de desenvolvimento foram utilizados Flex3 e ActionScript3, uma solução mais robusta, com maior desempenho e mais modular que uma solução baseada em Flash e ActionScript2;
- Organização do FlexiXML de forma a ser uma ferramenta genérica e expansível, como exposto na secção 4.

1.2 Estrutura do Artigo

O artigo está organizado da seguinte forma: a Secção 2 contém uma breve descrição da linguagem UsiXML, a linguagem por omissão da ferramenta FlexiXML. Na secção 3 são apresentados trabalhos relacionados. Na secção 4 é efectuada a descrição da ferramenta FlexiXML, sendo na secção 5 apresentado um exemplo de uma aplicação gerada por esta ferramenta. Por fim, na secção 6 são apresentadas algumas conclusões e propostas de trabalho futuro.

2. USIXML

A USer Interface eXtensible Markup Language (UsiXML) consiste numa linguagem para descrição de interfaces com o utilizador (UIDL) que permite que a modelação possa ser efectuada independentemente da linguagem de programação, plataforma computacional ou ambiente em que a interface é utilizada. Esta UIDL descreve a interface de uma aplicação a um alto nível de abstracção e não exige conhecimentos de programação, o que permite que analistas, *designers*, programadores e utilizadores finais a possam utilizar durante o ciclo de desenvolvimento.

A UsiXML permite descrever uma interface a vários níveis de abstracção. A organização dos diferentes níveis de abstracção é inspirada na *Framework Cameleon* (**C**ontext **A**ware **M**odelling for **E**nabling and **L**everaging **E**ffective interacti**ON**) [Cameleon04] que define etapas de desenvolvimento para aplicações interactivas com vários contextos. A estrutura desta *Framework* é composta pelas seguintes camadas (Figura 1):

- **Final UI (FUI)** – Corresponde á interface operacional, ou seja, a interface que pode ser executada ou interpretada em um determinado contexto de uso (numa plataforma computacional específica e com um conjunto de dispositivos específicos, utilizando objectos de interacção específicos). Os artefactos desta camada são o código fonte (Java, HTML, etc.) e os executáveis que efectuam o *render* da interface;
- **Concrete UI (CUI)** – Especificação da interface com o utilizador em termos de objectos gráficos de interacção e as suas relações. Efectua uma abstracção da interface que é independente da plataforma computacional;
- **Abstract UI (AUI)** – Descrição abstracta da interface que é independente do modelo de interacção (interacção gráfica, interacção vocal, …);
- **Tasks & Concepts** – Descreve as tarefas a serem efectuadas e os principais conceitos necessário para que sejam executadas.

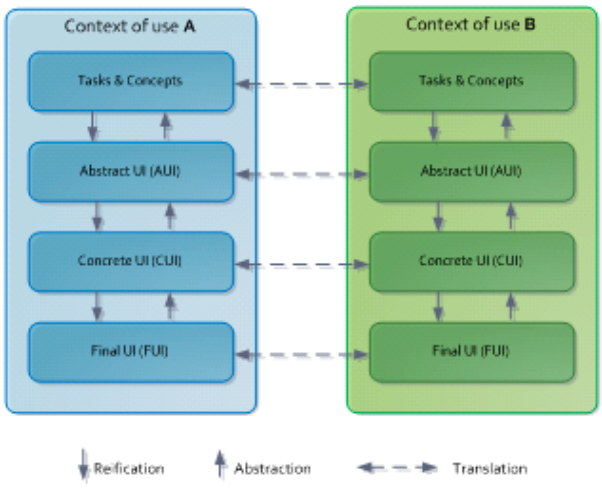

Figura 1: Framework *Cameleon*.

A *Cameleon Reference Framework* define que as etapas de desenvolvimento são realizadas através de transformações, as quais podem ser de dois tipos: transformações verticais e transformações horizontais.

As *transformações verticais* correspondem a processos para conversão de um modelo de interface de um nível

mais concreto para um mais abstracto e vice-versa. Incluem os seguintes dois tipos de transformações:

- **Reificação** – transformação de um nível de descrição mais abstracto para um nível mais concreto;
- **Abstracção** – transformação de um nível de descrição mais concreto para um nível mais abstracto.

As *transformações horizontais* permitem transformar um modelo dentro de um mesmo nível, mas entre contextos de usos diferentes, e incluem a transformação:

- **Tradução** – transformação entre etapas de desenvolvimento similares mas de contexto de uso diferente.

Com este paradigma de desenvolvimento é possível especificar uma interface em qualquer um dos níveis de abstracção e transformá-la para qualquer um dos outros níveis. A especificação da interface pode ser efectuada independentemente do contexto e/ou condições de interacção sendo possível obter uma ou mais interfaces gráficas para os diferentes contextos e nos diferentes níveis de abstracção.

2.1 Modelação

A linguagem UsiXML tem uma sintaxe e semântica bem definidas. A semântica é definida por um modelo expresso num diagrama de classes UML (Figura 2). A sintaxe é definida por *schemas* XML que resultam da conversão do diagrama de classes UML.

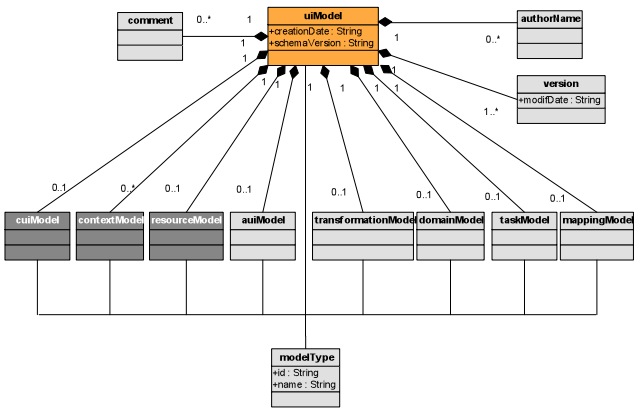

Figura 2: Diagrama de classes da UsiXML.

Descrever a linguagem em detalhe sai do âmbito deste artigo, não sendo possível por restrições de espaço. De seguida são apresentados os principais modelos interpretados pelo FlexiXML. Na secção 5, são apresentados excertos um exemplo concreto de utilização da linguagem.

2.1.1 uiModel

O *uiModel* correspondente ao modelo *core* de especificação da interface gráfica. Este componente contém as características comuns a todos os modelos, como a versão, autor, data de criação, *links* para a especificação UsiXML, entre outros (ver Figura 2).

O modelo *uiModel* é por sua vez composto por vários modelos. No caso específico do interpretador FlexiXML são analisados apenas: o *cuiModel*, o *contextModel* e o *resourceModel*. Estes são os 3 modelos que contêm a informação relativa ao desenho da interface gráfica e à interacção com o utilizador final.

2.1.2 cuiModel

O *cuiModel* especifica o *layout* da interface, ou seja, define os objectos que compõe a interface gráfica (CIO - *Concrete Interaction Objects*) e as relações entre eles (CUIR - *Concrete User Interface Relationships*).

2.1.3 contextModel

O *contextModel* é o modelo que descreve o contexto de utilização da aplicação, desde o tipo de utilizador à plataforma e ambiente computacional.

Embora neste momento o FlexiXML só interprete as características relativas ao tipo de utilizador, fica uma breve descrição dos diferentes tipos de contexto:

- **Plataforma** – Contém as características mais relevantes da plataforma de software, hardware e dispositivos anexados que influenciem a execução das tarefas do utilizador. A plataforma pode consistir numa série de dispositivos, numa série de componentes de software, características da rede a que a plataforma está conectada, capacidade de suporte de *wireless* (características Wap) e capacidades do *browser*;
- **Ambiente** – Descreve as propriedades do ambiente em que o utilizador está a utilizar a interface. As propriedades podem ser físicas (por exemplo: condições de luminosidade), psicológicas (por exemplo: níveis de *stress*) e organizacionais (por exemplo: localização e papel definidos no organograma);
- **Tipo de utilizador** – Constitui os estereótipos de utilizador, ou seja, define categorias de utilizadores com características semelhantes (p.e., idioma).

Deste modelo a propriedade com maior interesse para o FlexiXML corresponde ao idioma (*language*), que define o idioma em que o utilizador deve visualizar a aplicação gerada. No ficheiro UsiXML é possível definir mais do que um contexto, o que permite que o utilizador possa visualizar a aplicação em diferentes idiomas.

Para especificação das traduções a apresentar nos componentes gráficos é necessária a utilização de um modelo adicional, o *resourceModel*.

2.1.4 resourceModel

O *resourceModel* é o modelo que contém as definições dos objectos gráficos que dependem de um contexto (p.e. localização, idioma, cultura, etc). Este modelo contém todo o tipo de conteúdo que pode ser atribuído a um objecto de interacção (conteúdo, *tooltip*, etc).

3. TRABALHOS RELACIONADOS

Como referido na secção 2 a linguagem UsiXML tem uma arquitectura subjacente que lhe permite que a especificação de uma interface possa ser efectuada a dife-

rentes níveis de abstracção e dispõem de mecanismos de transformação entre eles.

Para dar suporte a estas transformações, a UsiXML dispõe de um conjunto de ferramentas, as quais podem ser dividas em duas categorias:

- **Editores** – Ferramentas para criação de descrições UsiXML. A forma como as descrições são criadas varia com o tipo de ferramenta. Podemos destacar, SketichXML [Coyette07] que utiliza como *input* interfaces desenhadas à mão (*sketchs*), GrafiXML [Michotte08] em que interface é criada por manipulação directa de componentes no ecrã, VisiXML [Coyette07] em que o desenho da interface é efectuado sobre o Microsoft Visio, entre outras ferramentas existentes.
- **Interpretadores** – Motores de geração de interfaces gráficas descritas em UsiXML. Cada interpretador gera interfaces com um conjunto de características específicas. Das ferramentas existentes podemos salientar, FlashiXML [Youri04] que gera interface vectorias, HaptiXML [Haptic09] que gera interfaces gráficas em 3D com interacção pelo toque, QtkiXML [Denis05] que criar interface para múltiplas plataformas, InterpiXML [Goffette07] que permite a interpretação simultânea de várias descrições UsiXML.

Para além de UsiXML, podem referir-se outras UIDLs que abordam diferentes aspectos de uma interface gráfica: AUIML [Argollo97], UIML [Abrams99], XIML [Puerta02], Teresa XML [Paternò02], WSXL [Abrams99], XUL [XUL09], entre outras. A maioria destas linguagens já é suportada por ferramentas para criação e interpretação das especificações. Alguns exemplos são: Teresa, Uiml.NET, ou Liquid UI.

Uma análise comparativa destas linguagens e ferramentas não é possível, no contexto deste artigo, por restrições de espaço. O leitor interessado poderá consultar, por exemplo, a revisão do estado da arte apresentada em [Souchon03].

4. PROPOSTA

O FlexiXML é um motor de geração de interfaces gráficas descritas segundo uma linguagem de modelação definida. A versão actual da ferramenta inclui apenas a linguagem UsiXML, mas permite a inclusão de outras linguagens declarativas, bastando para isso integrar *parseres* que as interpretem.

As interfaces geradas são criadas na tecnologia Flex3 + ActionScript3, contudo a ferramenta está pensada para permitir gerar interfaces recorrendo a outras tecnologias. Para isso será necessário desenvolver geradores de interfaces adequados (ver descrição da arquitectura nesta secção). Ou seja, a aplicação FlexiXML está estruturada para que o utilizador possa especificar em que linguagem pretende que a interface seja gerada.

As duas características acima surgem para facilitar a utilização da ferramenta, não limitando os utilizadores a uma linguagem de modelação e a uma tecnologia de geração específicas. A ferramenta propriamente dita está desenvolvida em Air + Flex, o que a torna independente da plataforma computacional em que é executada (actualmente o FlexiXML está disponível em dois formatos: desktop e Web).

Nas secções seguintes é apresentada uma visão geral da ferramenta, bem como a sua arquitectura.

4.1 Plataforma

O FlexiXML está estruturado em torno do conceito de *plugins*, em que cada *plugin* implementa um conjunto de funções específicas. Este tipo de arquitectura permite que a ferramenta possa estar em constante evolução, com a integração de novos *plugins* com outras funcionalidades e sem impacto nos já existentes.

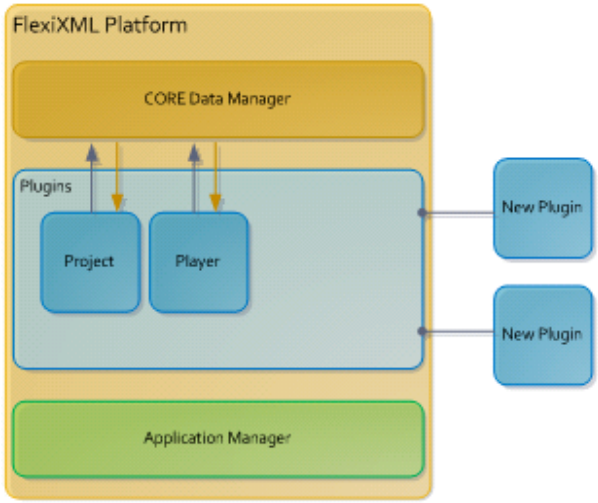

Figura 3: Plataforma do FlexiXML.

Na Figura 3 está representada a arquitectura base da aplicação. Esta pode ser divida em 3 camadas:

- **Application Manager** – Gestor da aplicação: responsável pela gestão de mensagens e da estrutura de *plugins*.
- **Plugins** – *Plugins* disponíveis na aplicação, sendo possível a integração de *plugins* adicionais;
- **CORE Data Manager** – Componente responsável por guardar e disponibilizar a informação que é partilhada por todos os *plugins*.

4.2 Workflow

Para realizar a interface gráfica, o FlexiXML recebe como entrada um ficheiro onde é indicada a localização de dois ficheiros adicionais: um com a descrição da interface, outro com a lógica da parte interactiva da mesma. Estes ficheiros são interpretados pelo *plugin* Project que envia a informação neles contida ao CORE Data Manager. O *plugin* Project cria entidades que representam os componentes presentes no modelo e que podem depois ser interpretadas pelo *plugin* Player.

O CORE Data Manager centraliza a informação partilhada por todos os *plugins*. Desta forma, sempre que

um *plugin* necessitar de informação sobre o projecto actual efectua o pedido a este componente.

Só depois de o projecto ser carregado pelo Project é que o Player pode gerar a interface gráfica baseado na informação disponibilizada pelo CORE Data Manager.

Este processo está apresentado na Figura 4.

4.3 Plugins

Como referido anteriormente esta aplicação esta desenvolvida sob o conceito de *plugins* para que novos *plugins* possa ser integrados de tal forma que a ferramenta possa estar em constante evolução. Para que esta integração seja possível cada *plugin* tem de respeitar a estrutura definida no diagrama de classes apresentado na Figura 5.

O FlexiXML é composto por dois *plugins* principais:

- **Project Plugin** – *Plugin* responsável por carregar o projecto. Efectua o *parsing* do ficheiro com a descrição da interface e carrega o ficheiro com o controlo da parte interactiva da mesma. É neste *plugin* que o utilizador pode especificar qual a UIDL que pretende utilizar.
- **Player Plugin** – *Plugin* responsável por gerar e animar a interface gráfica do projecto actual com a informação relevante do utilizador. Para além de gerar e animar a interface, permite efectuar alterações em *runtime* à mesma, tais como, mudança de estilos e de idioma.

4.4 Parsers

O Parser é a entidade responsável por interpretar o ficheiro com a descrição da interface e preencher as estruturas de dados com essa informação para que outros *plugins* a possam consultar.

Actualmente apenas está disponível o *parser* para UsiXML, mas outros podem ser integrados. Para tal o *parser* tem que implementar o método parse().

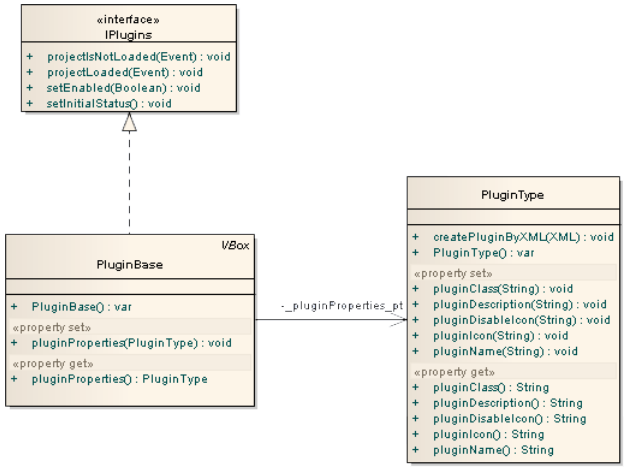

Figura 5: Diagrama de classes de um *plugin* FlexiXML.

4.5 Geradores

No momento de geração da interface o Player acede ao CORE Data Manager para obter a informação acerca da descrição actual.

Esta informação é então interpretada para serem criados os componentes e/ou objectos que a representam. Para cada componente definido no modelo é criado o componente gráfico correspondente, o comportamento e conteúdo associado e as possíveis transições entre eles. O mapeamento entre cada elemento UsiXML e os *widgets*/controlos na interface a gerar está definido num ficheiros de configuração. Esta configuração é apresentada na secção 4.6.

Actualmente o FlexiXML incluí apenas um gerador em Flex, mas podem ser integrados outros, desde que implementem o *interface* definido na Figura 6.

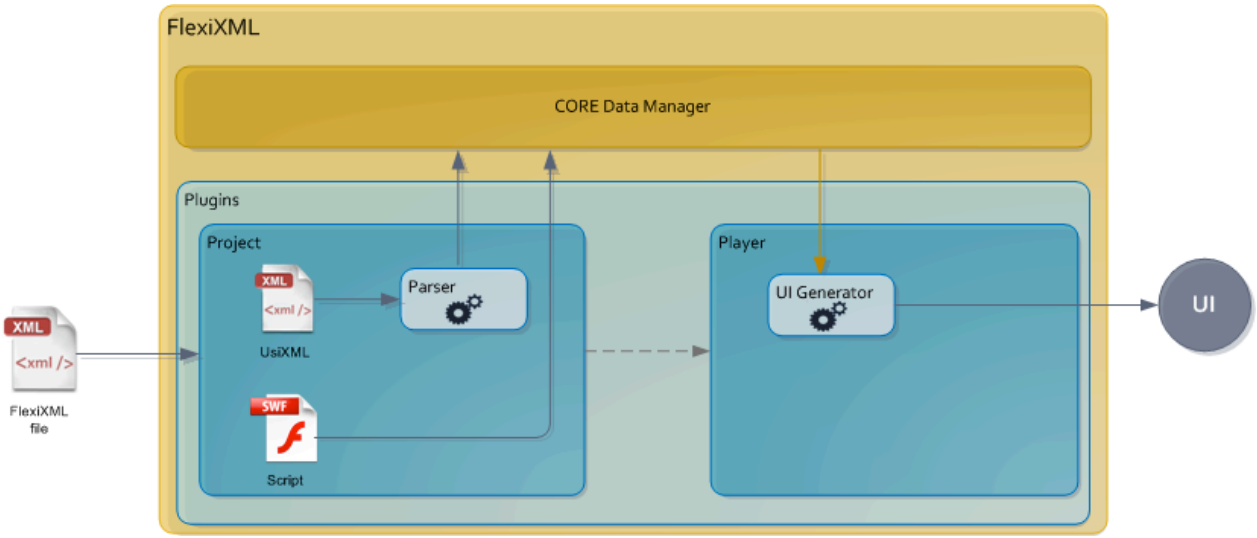

Figura 4: *Workflow* do FlexiXML

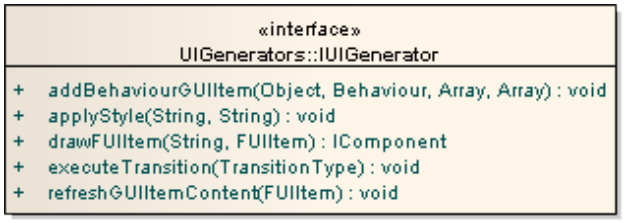

Figura 6: Interface de um gerador de *interfaces* FlexiXML.

4.6 Configurações

O FlexiXML utiliza um conjunto de ficheiros de configuração que permite ao utilizador efectuar alterações ao comportamento da ferramenta bem como ao seu aspecto visual. Os principais ficheiros de configuração são:

- **Mensagens** – Todas as etiquetas e mensagens utilizadas na ferramenta podem ser traduzidas para diferentes linguagens, sem ser necessário recompilar o código.
- ***Plugins*** – Define a lista de *plugins* que integram a ferramenta. Sempre que se pretende adicionar um *plugin* é necessário adicioná-lo neste ficheiro para que a ferramenta o disponibilize;
- **UIDLs** – Este ficheiro define a lista de linguagens de especificação de interfaces que o FlexiXML interpreta. Para além de listar quais as UIDLs disponíveis define todas as características necessárias para a sua integração na aplicação, nomeadamente: *parser* da UIDL, linguagem de geração disponíveis para esta linguagem, mapeamento entre os objectos da linguagem e os componentes gráficos que os representam (ver Figuras 17, 18 e 19 no Apêndice).
- **Estilos** – A lista de estilos disponibilizados no Player está definido num ficheiro XML. Desta forma, sempre que se pretender inserir novos estilos, só é necessário adicioná-los neste ficheiro.

5. EXEMPLO ILUSTRATIVO

Nesta secção é apresentado um exemplo que descreve a forma como FlexiXML pode ser utilizado para gerar uma interface gráfica descrita em UsiXML.

O exemplo consiste numa aplicação que simula um *player* de música. Dada a complexidade do modelo, nesta descrição são apresentados apenas alguns excertos do mesmo.

5.1 Desenho

A interface da aplicação "Music Player" pode ser divida em 4 áreas:

- **Player** – Área onde são apresentados os botões para efectuar o controlo sobre a música que se pretende ouvir.
- **CurrentMusic** – Corresponde à área onde é apresentada a informação da música que está actualmente a tocar;
- **View** – Composta pelos botões que definem os formatos de visualização da lista de álbuns;
- **CurrentView** – Área onde é apresentada a vista actual.

Tendo em consideração esta estrutura a interface pode ser decomposta nas áreas apresentas na Figura 7.

A área View (*box* com o botão das vistas) contém 3 botões que permitem alternar entre os 3 formatos de visualização que aplicação apresenta: **ListView** (permite visualizar os álbuns e as respectivas músicas em lista); **GridView** (visualização de todos os álbuns em grelha); **CoverView** (visualização de um álbum a cada momento). O utilizador pode, a qualquer momento, alterar a vista que pretende visualizar.

A Figura 8 apresenta a estrutura base do *cuiModel* da interface em UsiXML. A partir da versão final do *cuiModel,* o FlexiXML gera a aplicação apresentada na Figura 9.

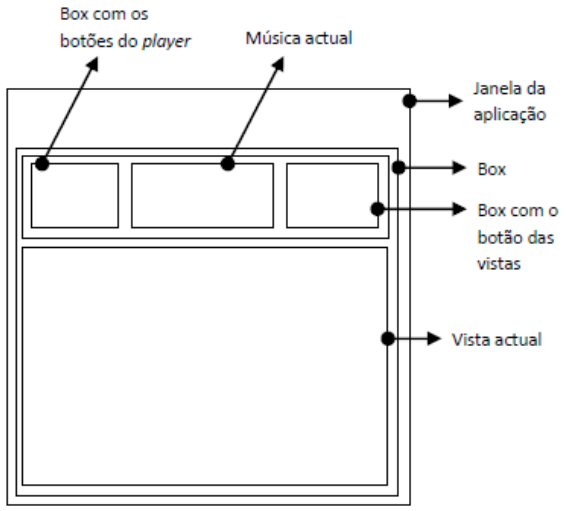

Figura 7: Decomposição da aplicação em *box's*.

```
<cuiModel id="musicPlayer-cui"
          name=" musicPlayer-cuiModel">
   <window id="playerWindow" …>
      <box id="mainBox" …>
         <box id="headerBox" …>
            <box id="playerBox" …/>
            <box id="currentMusicBox" …/>
            <box id="viewsBox" …/>
         </box>
         <box id="currentView" …>
                              […]
         </box>
      </box>
   </window>
</cuiModel>
```

Figura 8: cuiModel da aplicação MusicPlayer.

Figura 9: Music player (coverView e gridView) gerado pelo FlexiXML.

5.2 Comportamentos

Cada objecto que compõe a *interface* pode ter associado um comportamento (*behavior*). Um *behavior* corresponde às acções que o objecto deve efectuar quando o utilizador interage com o mesmo. A interacção do utilizador com o objecto gera eventos que despoletam a execução das acções. Uma acção pode corresponder à chamada de um método ou a uma mudança na interface gráfica.

Na aplicação "Music Player" é possível alternar entre as diferentes vistas, pressionando o botão correspondente. Por exemplo, se o botão "GridView" for pressionado, é visualizada essa vista e escondidas as restantes. Para além desta transição, deve ser invocado o método "updateGridView" que é responsável por atribuir os dados necessários a esta vista (p.e., lista de álbuns a visualizar).

A especificação deste exemplo está apresentada nas Figuras 10 e 11, o resultado final está representado na Figura 13.

5.3 Contextos (Idiomas)

Para a criação da aplicação em diferentes idiomas é necessário criar 2 modelos: o *contextModel* (no caso do FlexiXML apenas interpreta o idioma, como tal serve para criar os idiomas em que se pretende que a *interface* a descrever fique disponível) e o *resourceModel* (conteúdo dos objectos nos diferentes idiomas).

Na aplicação "Music Player" foram criados dois idiomas: inglês e francês (ver Figura 12).

```
<button id="gridButton”>
  <behavior id="gridView">
     <event id="gridViewEvt"
           eventType="depress"
            eventContext="gridButton"/>
     <action id="gridViewAct">
        <transition transitionIdRef
              ="GridViewTr1" />
        <transition transitionIdRef~
              ="GridViewTr2" />
        <transition transitionIdRef
              ="GridViewTr3" />

        <methodCall methodName
              ="updateGridView"/>

     </action>
  </behavior>
 </button>
```

Figura 10: Especificação do comportamento do botão da "*GridView*" .

```
<graphicalTransition id="GridViewTr1"
          transitionType="fadeOut">
   <source id="gridButton" />
   <target id="listView" />
</graphicalTransition>

<graphicalTransition id="GridViewTr2"
          transitionType="fadeOut">
   <source id="gridButton" />
   <target id="coverView" />
</graphicalTransition>

<graphicalTransition id="GridViewTr3"
          transitionType="fadeIn">
   <source id="gridButton" />
   <target id="gridView" />
</graphicalTransition>
```

Figura 11: Especificação das transições gráficas despoletadas pelo botão da "*GridView*".

```
<contextModel id="playerContextModel"
             name="playerContextModel">
   <context id="playerContext_En_US"
            name="playerContext_En_US">
      <userStereotype
            id="playerContextUser_US"
            language="en_US"
            stereotypeName
               ="playerContextUser_US"/>
   </context>

   <context id="playerContext_FR"
            name="playerContext_FR">
      <userStereotype
            id="playerContextUser_FR"
            language="FR"
            stereotypeName
               ="playerContextUser_FR"/>
   </context>
</contextModel>
```

Figura 12: Especificação dos idiomas.

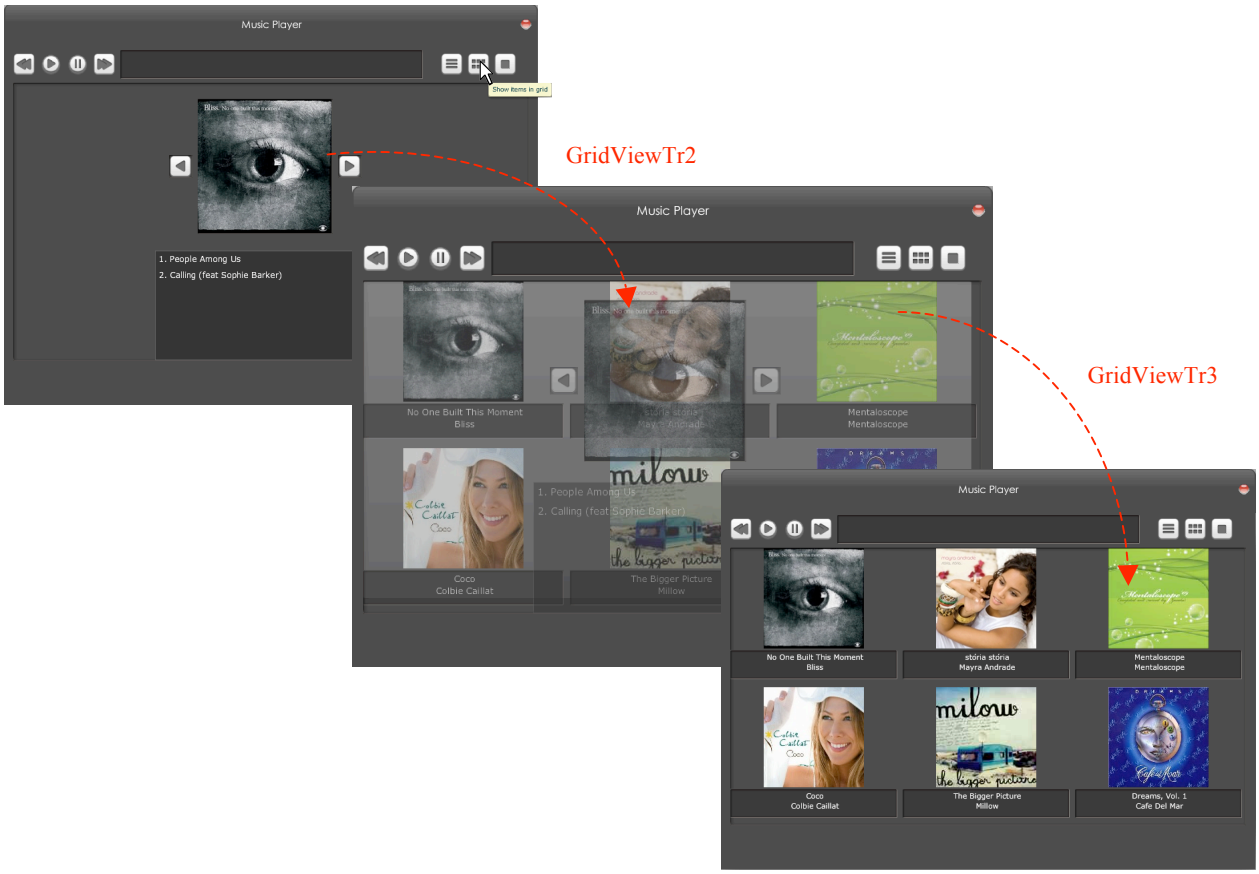

Figura 13: Transições gráficas para visualização da "*GridView*".

Na Figura 14 é apresentada a definição do título da aplicação nos diferentes idiomas.

```
<resourceModel id="playerResourceModel"
        name="playerResourceModel">
 <cioRef cioId="playerWindow">
   <resource content="Music Player"
        contextId="playerContext_En_US"/>
   <resource content="Lecteur de Musique"
        contextId="playerContext_FR"/>
 </cioRef>
</resourceModel>
```

Figura 14: Especificação do título da aplicação nos diferentes contextos.

Na Figura 15 é apresentado a janela da aplicação em ambos os idiomas.

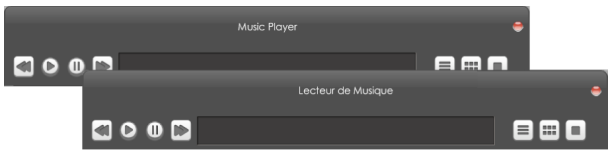

Figura 15: Aplicação "Music Player" em inglês e francês

5.4 Estilos

FlexiXML permite a alteração em tempo de execução do estilo da interface gerada. A lista de estilos que podem ser aplicados está definida num ficheiro de configuração (ver secção 4.6), onde é indicado qual o nome e localização do ficheiro com as definições do mesmo.

Cada estilo é definido no formato CSS, o que permite que possam conter característica como imagens (skins), fontes, class selectors, entre outros. O FlexiXML interpreta estes estilos a partir de um ficheiro externo no formato swf (gerado a partir do ficheiro CSS).

Na Figura 16. é apresentada a *interface* de "Music Player" com diferentes estilos[1].

6. CONCLUSÕES E TRABALHO FUTURO

Este artigo descreve uma abordagem à prototipagem de interfaces e apresenta a primeira versão da ferramenta FlexiXML que realiza a interpretação e animação de interfaces descritas em UsiXML.

Neste momento a ferramenta já permite:

- A interpretação de modelos UsiXML: desenho da interface, e interacção com a mesma (transições gráficas e invocação de métodos);
- A visualização da interface gerada nos diferentes idiomas em que é definida;
- A alteração do estilo das interfaces geradas em tempo de execução.

[1] Os estilos utilizados na aplicação tiveram por base alguns estilos disponíveis em http://www.scalenine.com/.

A arquitectura da ferramenta está ainda pensada para suportar a integração de novos *plugins*, bem como de novas linguagens de modelação e de programação das interfaces gráficas.

Neste momento ainda existe um conjunto de funcionalidades que podem ser adicionadas de forma a optimizar o FlexiXML, das quais podemos salientar:

- Criação de novos *plugins*, nomeadamente um editor de modelos.
- Alargar a biblioteca de componentes, de modo a acomodar um maior conjunto de representações possíveis;
- Geração de protótipos para diferentes dispositivos e plataformas;
- Implementação de *parsers* para novas UIDLs e geração em diferentes linguagens de programação.

7. AGRADECIMENTOS

Os autores gostariam de agradecer a Jean Vaderdonckt, da Universidade Católica de Louvain, pelo apoio prestado relativamente à linguagem de modelação UsiXML e à ferramenta FlashiXML. Sandrine Mendes gostaria de agradecer também á empresa *Alert Life Science Computing* por apoiar este projecto.

8. REFERÊNCIAS

[Abrams99] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S. and Shuster J. UIML: *An Appliance-Independent XML User Interface Language*. In A. Mendelzon, editor, Proc. of 8th Inter. World-Wide Web Conference WWW'8, Elsevier, 1999.

[Argollo97] Argollo M. Jr. and Olguin *C.. Graphical user interface portability*. CrossTalk: The Journal of Defense Software Engineering, 10(2):14–17, 1997.

[Arsanjani02] Arsanjani A. et al. *(WSXL) Web Service Experience Language Version 2*. IBM Note 10 April 2002. IBM, 2002.

[Cameleon04] The Cameleon Project – plasticity of user interfaces. http://giove.cnuce.cnr.it/cameleon.html. Acedido em 26/06/2009.

[Coyette07] Coyette, Adrien, *A Methodological Framework for Multi-Fidelity Sketching of User Interfaces*, Ph.D. thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 22 October 2007.

[Denis05] 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, September 2005.

[Goffette07] Goffette, Y. and Louvigny, H.-N. *Development of multimodal user interfaces by interpreta-*

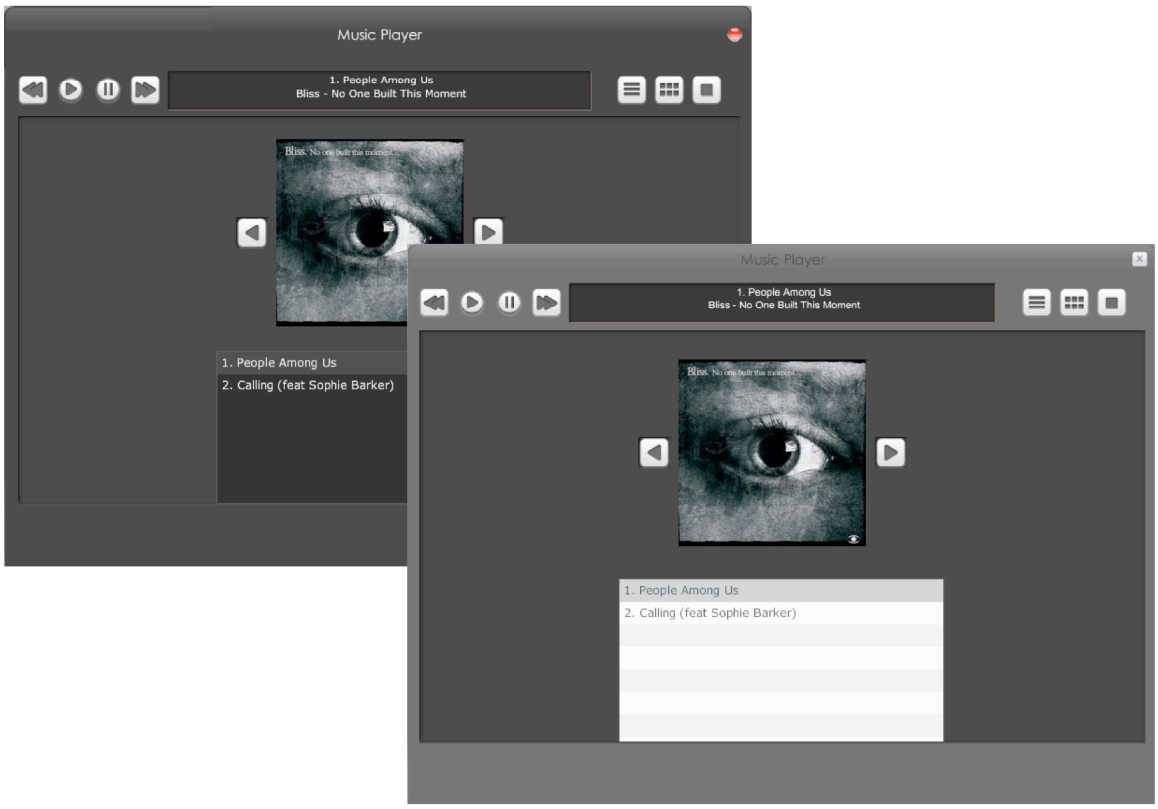

Figura 16: "Music Player" com diferentes estilos.

tion and by compiled components : a comparative analysis between InterpiXml and OpenInterface, Université Catholique de Louvain, 28 August 2007.

[Haptic09] Kaklanis, N. *3D HapticWebBrowser*. http://kaklanis.googlepages.com/nickkaklanis-3dhapticwebbrowser. Acedido em 26/06/2009.

[Michotte08] Michotte, B., and Vanderdonckt, J. *GrafiXML, A Multi-Target User Interface Builder based on UsiXML*. Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS'2008, IEEE Computer Society Press. 2008

[Paternò99] Paternò, F., *Model-based Design and Evaluation of Interactive Applications*, Springer Verlag, Berlin, 1999.

[Paternò02] Paternó. F. and Santoro. C. *One model, many interfaces*. In Ch Kolski and J. Vanderdonckt (Eds.), editors, Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CADUI'2002, pages 143–154, Kluwer Academics Publishers, 2002.

[Puerta02] Puerta, A. and Eisenstein, J. *XIML: A common representation for interaction data*. In Proc. Of the 7th International Conference on Intelligent User Interfaces, pages 69 – 76, ACM Press, 2002.

[daSilva00] da Silva, P.P. User interface declarative models and development environments: a survey. In Interactive Systems: Design, Specification, and Verification, Lect. Notes in Computer Science, Vol. 1946, pp. 207-226, Springer, 2000.

[Silva06] Silva, E. *Sistemas interactivos*. Departamento de Computação, Universidade Federal de Ouro Preto. 2006.

[Souchon03] Souchon, N. and Vanderdonckt, J., *A Review of XML-Compliant User Interface Description Languages*, Interactive Systems: Design, Specification, and Verification. Lect. Notes in Computer Science, Vol. 2844, pp. 377-391, Springer, 2003.

[UsiXML09] Université catholique de Louvain. *UsiXML – USer Interface eXtensible Markup Language*. http://www.usixml.org/. Acedido em 26/06/2009.

[XUL09] Mozilla foundation. *XUL Tutorial*, https://developer.mozilla.org/en/XUL_Tutorial. Acedido em 26/06/2009.

[Youri04] Youri, V. B. *Etude et implémentation d'un générateur d'interfaces vectorielles à partir d'un langage de description d'interfaces utilisateur*, Université Catholique de Louvain. 2004.

9. APÊNDICE

```
<ComponentsMapper>
   <window component = "Classes.Components.FlexiXMLWindow"/>
   <button component = "Classes.Components.FlexiXMLButton"/>
   <textComponent component = "Classes.Components.FlexiXMLText"/>
  <checkBox component = "mx.controls.CheckBox"/>
   …
<ComponentsMapper>
```

Figura 17: Ficheiro de configuração do mapeamento de componentes UsiXML para os *widgets* correspondentes.

```
<EventsMapper>
      <release event = "mouseUp"/>
      <depress event = "mouseDown"/>
      <rollOver event = "mouseOver"/>
      <rollOut event = "mouseOut"/>
</EventsMapper>
```

Figura 18: Ficheiro de configuração do mapeamento de eventos UsiXML para os interpretados pelo gerador.

```
<ActionsMapper>
      <transition>
            <boxOut effect="Classes.Animation.Zoom" direction = "OUT"/>
            <boxIn effect="Classes.Animation.Zoom" direction = "IN"/>
            <fadeOut effect="Classes.Animation.Fade" direction = "OUT"/>
            <fadeIn effect="Classes.Animation.Fade" direction = "IN"/>
            <close effect="Classes.Animation.Visibility" direction = "OUT"/>
            <open effect="Classes.Animation.Visibility" direction = "IN"/>
      </transition>
</ActionsMapper>
```

Figura 19: Ficheiro de configuração do mapeamento de transições UsiXML para transições do gerador.