NCLFORMS: Uma API para desenvolvimento de GUIs em aplicações interativas para TV Digital

Renan Ribeiro de Vasconcelos
Universidade Federal do Rio de Janeiro - UFRJ
Centro de Tecnologia, bloco H, sala H-217
Rio de Janeiro, RJ
renanrv@cos.ufrj.br

Cláudia Maria Lima Werner
Universidade Federal do Rio de Janeiro - UFRJ
Centro de Tecnologia, bloco H, sala H-217
Rio de Janeiro, RJ
werner@cos.ufrj.br

Wagner Schau
Universidade Federal do Rio de Janeiro - UFRJ
Centro de Tecnologia, bloco H, sala H-217
Rio de Janeiro, RJ
schau@cos.ufrj.br

Débora Christina Muchaluat Saade
Universidade Federal Fluminense - UFF
Instituto de Computação - IC
Laboratório MídiaCom
Niterói, RJ
debora@midiacom.uff.br

Glauco Fiorott Amorim
Centro Federal Celso Suckow da Fonseca - Cefet/RJ
Coordenação de Telecomunicações/TV Digital
UnED Petrópolis, RJ
glauco.amorim@gmail.com

ABSTRACT

Ginga-NCL is the Brazilian declarative middleware that provides support for the development of interactive TV applications. Ginga-NCL indicates the use of NCL (Nested Context Language), which is a declarative language for building interactive multimedia applications, and Lua, which is a script language that can be used together with NCL. NCL itself does not provide facilities to build forms, as HTML does. However, form components can be implemented in Lua and included in NCL documents. This work proposes an API, named NCLForms, to provide the creation of Lua form components inside NCL programs. Using NCLForms, it is easy to create and manipulate forms inside an NCL program, without the need to program in Lua.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Design Tools and Techniques; H.5.2 [**User Interfaces**]: Graphical user interfaces (GUI)—*developement, interaction styles*

General Terms

Application

Keywords

Aplicações de TV Digital, Interface Gráfica, TV Digital, Reutilização de Software, Padrões de Projeto

1. INTRODUÇÃO

Com a chamada TV Digital interativa, aplicações podem ser executadas diretamente na televisão. Do ponto de vista do desenvolvimento, novos paradigmas devem ser seguidos ao se pensar em uma aplicação para TV, além de aspectos como domínios de aplicações, tipos de usuários, interface de interação, dentre outros.

Uma forma de apoiar o desenvolvimento de aplicações é por meio de bibliotecas de código. Tal alternativa permite colocar em prática técnicas de reúso a fim de acelerar a construção de tais aplicações. Outro caminho muito comum em desenvolvimento de software diz respeito aos *design patterns*, que fornecem orientações na forma de melhores práticas para a implementação de projetos em domínios específicos.

Assim como em qualquer área de software, aplicações interativas de TV Digital também podem contar com bibliotecas e *patterns*, porém ainda necessitam de novas propostas e principalmente da aplicação de tais propostas no cenário brasileiro, pelo fato do middleware brasileiro Ginga [1] utilizar tecnologias inovadoras e ainda não muito comuns no mercado de desenvolvimento de software .

O presente trabalho, ao identificar uma dificuldade em inserir elementos de interface gráfica em aplicações desenvolvidas para o ambiente declarativo Ginga-NCL, apresenta uma biblioteca de código que propõe uma *Application Programming Interface* (API) para desenvolvimento de interfaces de usuário com elementos gráficos em aplicações com componentes de formulários. Tal proposta é colocada sob a ótica da reutilização de software no contexto do ambiente Ginga-NCL, através dos recursos fornecidos pela integração entre as linguagens NCL e Lua. Com isso, o estudo em questão busca colaborar no desenvolvimento de interfaces gráficas em aplicações de TV Digital no cenário brasileiro.

Dessa forma, o texto é organizado em cinco seções, con-

tando com esta introdução. A Seção 2 apresenta uma descrição dos tipos de aplicações interativas para TV Digital, além de destacar as principais características do modelo brasileiro para interatividade. A Seção 3 faz uma revisão sobre APIs em termos de reutilização de software, abordando também características de elementos de interface gráfica, além de ferramentas voltadas para a construção de *Graphical User Interfaces* (GUIs). Na Seção 4, é apresentada a API desenvolvida, chamada de NCLForms, e é proposta a sua aplicação na implementação de soluções em uma linguagem de *patterns* estudada. A Seção 5 conclui o trabalho, apresentando as considerações finais.

2. APLICAÇÕES INTERATIVAS DE TV DIGITAL

A fim de guiar o progresso da pesquisa, alguns pontos devem ser estabelecidos, de forma que o conceito de aplicações interativas para TV Digital não seja interpretado erroneamente. Nesse contexto, algumas características podem ser ressaltadas, como independência de conteúdo (o que diferencia aplicações interativas para TV de TV interativa ou programas de TV interativos) e tipo de aplicação, e estar relacionado à televisão digital. Uma simples definição que abrange esses tópicos é a que aponta aplicações interativas de TV como serviços avançados ou interativos com TV Digital [3].

Um dos equipamentos fundamentais em ambientes de TV Digital é o receptor digital (*set-top box* ou a própria TV). O software incluído no receptor possui alguns componentes, sendo o *middleware* o componente central, funcionando como uma camada de interface entre o hardware e o software básico do dispositivo e as aplicações. O *middleware* determina as linguagens de programação que podem ser utilizadas no desenvolvimento de aplicações interativas.

O modelo brasileiro de TV Digital é baseado no *middleware* Ginga [1]. Tal sistema tem suporte a aplicações declarativas e a aplicações procedurais. O suporte a esses recursos é dividido em dois ambientes principais: um ambiente de apresentação, conhecido como Ginga-NCL, e um ambiente de execução, denominado Ginga-J. No entanto, aplicações híbridas são igualmente possíveis, fazendo uso de recursos dos dois ambientes, além da possibilidade de usar uma linguagem procedural, como Lua, em conjunto com aplicações declarativas na máquina de apresentação Ginga-NCL. O módulo Ginga-NCL faz uso da linguagem declarativa NCL (*Nested Context Language*) e suas características podem ser encontradas em [7] e [8].

Scripts Lua podem ser adotados no subsistema Ginga-NCL a fim de implementar objetos procedurais em documentos NCL. Conforme destaca [7], a fim de se adaptar ao cenário de TV Digital, a linguagem Lua foi estendida com novas funcionalidades, adicionando os seguintes módulos à biblioteca-padrão da linguagem: *event*, *canvas*, *settings* e *persistent*.

3. REUTILIZAÇÃO E DESENVOLVIMENTO DE *GRAPHICAL USER INTERFACES*

A adoção de bibliotecas de código no processo de desenvolvimento de software é uma atividade comum e deve ser cada vez mais incentivada, dados os benefícios da reutilização neste contexto. Grandes sistemas são baseados em coleções reutilizáveis de funcionalidades implementadas na forma de bibliotecas [2].

No entanto, tais funcionalidades necessitam de uma interface que forneça o serviço esperado de tudo o que foi implementado na coleção em questão. A fim de exercer essa função, as APIs (*Application Programming Interfaces*) permitem aos desenvolvedores acessar bibliotecas e aplicar os serviços providos pelas mesmas. Dentre suas principais características, as APIs fornecem abstrações em alto nível, suportam reutilização de código (sua essência), e colaboram de forma a simplificar a fase de programação, uniformizando algumas tarefas [6].

As alternativas visando o reúso de software são aplicáveis a diferentes áreas, desde aplicações tradicionais em ambientes *desktop* até aplicações para TV Digital. No que diz respeito às aplicações interativas de TV, um campo que merece destaque é o de elementos de interface gráfica.

Em um ambiente como o da TV Digital, cuja interação se dá basicamente pelo controle remoto, não tendo à disposição um teclado ou um mouse, o projeto da interface gráfica deve levar diversos aspectos em consideração, a fim de produzir uma aplicação intuitiva. Nesse sentido, o desenvolvimento de interfaces para o usuário deve ter como preocupação quais os dispositivos em que tais interfaces serão visualizadas, os diferentes grupos de usuários, e os variados ambientes de uso [12].

4. NCLFORMS - APRESENTAÇÃO E CONTEXTO DE USO

Utilizando os conceitos de bibliotecas de código e elementos de interface gráfica, juntamente com as características próprias das aplicações interativas para TV Digital, mais propriamente dentro do modelo brasileiro, foi desenvolvida neste trabalho uma API para a construção de *Graphical User Interfaces* (GUIs), cujo nome é NCLForms.

A implementação da API NCLForms tem como objetivo fornecer um conjunto de elementos de interface gráfica, aqui tratados como *widgets*, ao desenvolvedor de aplicações NCL para o *middleware* Ginga, de modo a facilitar a adição de formulários em aplicações interativas para TV Digital.

Apesar de ser possível inserir formulários em aplicações NCL por meio de documentos HTML, não é possível uma customização tão profunda, a ponto de escolher aspectos relacionados à aparência dos elementos e, até mesmo, uma integração mais eficiente com os demais componentes da aplicação.

É importante lembrar que a exibição de formulários e elementos gráficos em HTML/CSS é apenas suportada pelo padrão brasileiro. Ou seja, dependendo da implementação do *middleware*, podem ocorrer incompatibilidades e variações na exibição desses elementos. Como a implementação dos *widgets* na API foi feita usando o padrão brasileiro NCLua, pode-se contar com a padronização da sua exibição e do seu comportamento, assim como é possível prevenir problemas de incompatibilidade entre diferentes versões do *middleware*. Além disso, a criação de documentos NCL que utilizam a API ocorre de maneira bem simplificada, cabendo ao desenvolvedor cuidar apenas da parte de tratamento dos dados relacionados ao formulário, pois a parte de interface já se encontra estruturada pela API.

A Figura 1 exibe um protótipo de aplicação, apresentando um formulário interativo, fazendo uso da API para a cons-

trução de uma GUI.

Figura 1: Exemplo de uma aplicação desenvolvida com a API NCLForms.

Durante o desenvolvimento da API foram utilizadas como ferramentas a máquina virtual Ginga-NCL Virtual Set-top Box v.0.12.3 [5], para emular o *middleware* Ginga-NCL, o software de virtualização VMware Player v.3.1.4 [11], para executar a máquina virtual, e o ambiente de desenvolvimento Eclipse [10], com *plug-in* NCL Eclipse [4] para validação de arquivos NCL e *plug-in* akdebugger [9] para edição de arquivos Lua.

4.1 *Widgets*

Visando o desenvolvimento de um único elemento NCLForms, todos os *widgets* existentes em um formulário foram implementados de modo uniforme e integrado.

De forma a ilustrar a relação entre os elementos propostos, a Figura 2 apresenta um modelo conceitual na forma de um diagrama de classes de todos os *widgets* oferecidos pela API NCL Forms.

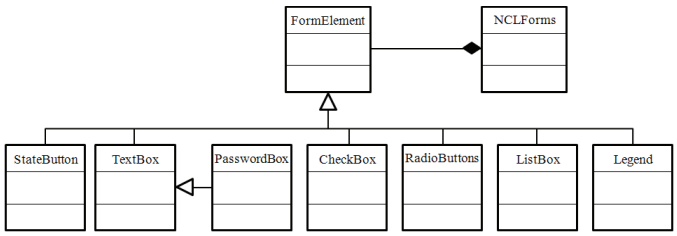

Figura 2: Modelo Conceitual das Widgets.

Cada *widget* conta com um conjunto de parâmetros que permite a sua customização.

Os parâmetros devem ser passados em uma ordem específica no documento NCL para que o *script* Lua possa interpretá-los corretamente. A fim de simplificar a relação dos mesmos, X e Y representam as coordenadas X e Y, respectivamente, do canto superior esquerdo de um *widget* e dX e dY representam o espaçamento nos eixos horizontal e vertical, respectivamente, entre um rótulo e uma caixa para entrada de caracteres em certos elementos. Os *widgets* e seus parâmetros são listados a seguir:

- **statebutton:** ordem de foco, rótulo, X, Y, largura, altura, cor de fundo, tamanho da fonte e cor da fonte;
- **textbox:** ordem de foco, rótulo, X, Y, dX, dY, largura, altura, cor de fundo, cor de borda, tamanho da fonte, cor da fonte e comprimento máximo;
- **passwordbox:** ordem de foco, rótulo, X, Y, dX, dY, largura, altura, cor de fundo, cor de borda, tamanho da fonte, cor da fonte e comprimento máximo;
- **checkbox:** ordem de foco, número de opções, rótulo, X, Y, largura, altura, cor de fundo, tamanho da fonte, cor da fonte e rótulo das opções;
- **radiobuttons:** ordem de foco, número de opções, rótulo, X, Y, largura, altura, cor de fundo, tamanho da fonte, cor da fonte e rótulo das opções;
- **listbox:** ordem de foco, número de opções, rótulo, X, Y, largura, altura, cor de fundo, número de opções visíveis, cor da fonte e rótulo das opções;
- **legend:** ordem de foco, número de opções, X, Y, largura, altura, cor de fundo, tamanho da fonte, cor da fonte e teclas e texto das dicas;

4.2 Utilização da API

Nesta seção, são apresentadas a forma correta de uso dos elementos oferecidos pela API NCLForms e a forma adequada de incorporar uma GUI desenvolvida com a API em um documento NCL.

A Figura 3 exibe um elemento NCL `<media>` fazendo referência ao *script* Lua que implementa a API NCLForms. Esse trecho de código informa à aplicação NCL qual o objeto NCLua deverá ser executado quando for solicitado. Deve-se destacar a importância de se definir um elemento `<property>` denominado *values* para o objeto de mídia. Tal elemento será importante na passagem de parâmetros para a API. Um elemento `<property>` denominado *result* também deve ser incluído no objeto de mídia. Esse elemento poderá armazenar, posteriormente, os resultados das seleções feitas em cada um dos *widgets* do formulário. O *script* NCLForms.lua é o arquivo principal da API, sendo responsável por exibir os elementos do formulário na aplicação.

```
<media id="forms" src="NCLForms.lua" descriptor="dsForms">
      <property name="values"/>
      <property name="result"/>
</media>
```

Figura 3: Adição da função principal da API no documento NCL

Após inserir o elemento `<media>` com a indicação do arquivo Lua a ser incorporado, deve-se informar quais os parâmetros de formulário para a API, a fim de saber quais os *widgets* que serão exibidos, bem como a forma deles. Isso é realizado através do elemento NCL `<link>`, atribuindo os parâmetros à interface de nome *values*. A Figura 4 ilustra esse passo.

```
<link xconnector="onBeginSetN">
   <bind role="onBegin" component="forms"/>
   <bind role="set" component="forms" interface="values">
      <bindParam name="var"
       value="textbox(1,AGENCIA:,420,210,70,0,350,16,white,black,12,black,10),
       textbox(2,CONTA:,420,250,70,0,350,16,white,black,12,black,10),
       passwordbox(3,SENHA:,420,290,70,0,350,16,white,black,12,black,10),
       radiobuttons(4,5,OPCOES:,420,350,220,150,white,12,black,Brasil,Cert.
                    Digital,Def.Visual,Nao Correntista,Exterior),
       radiobuttons(5,4,Titularidade,670,350,220,150,white,12,black,
                    1o. Titular,2o. Titular,3o. Titular,4o. Titular),
       statebutton(6,ENVIAR,600,520,105,35,silver,24,black),
       legend(1,3,1050,610,210,60,white,9,black,  OK,ENTER,Apagar,RED,Próximo,CURSOR_RIGHT),
       legend(2,3,1050,610,210,60,white,9,black,  OK,ENTER,Apagar,RED,Próximo,CURSOR_RIGHT),
       legend(3,3,1050,610,210,60,white,9,black,  OK,ENTER,Apagar,RED,Próximo,CURSOR_RIGHT),
       legend(4,5,1050,610,210,60,white,9,black,  OK,ENTER,Up,CURSOR_UP,Down,CURSOR_DOWN,Next,
              CURSOR_RIGHT,Previous,CURSOR_LEFT),
       legend(5,5,1050,610,210,60,white,9,black,  OK,ENTER,Up,CURSOR_UP,Down,CURSOR_DOWN,Next,
              CURSOR_RIGHT,Previous,CURSOR_LEFT),
       legend(6,2,1050,610,210,60,white,9,black,  OK,ENTER,Anterior,CURSOR_LEFT)"/>
   </bind>
</link>
```

Figura 4: Passagem de parâmetros para o script Lua

Após a passagem de parâmetros no documento NCL, a API irá interpretar esses dados e desenhar na tela os componentes indicados na interface *values*.

A comunicação entre o código NCL e o script Lua é feita através de uma função denominada *nclHandler*, presente no arquivo NCLForms.lua, contido no pacote que compõe a API e que foi registrada para tratar eventos de atribuição. Tal função é responsável por processar eventos voltados para a digitação de texto após a exibição dos componentes na tela, eventos de nome *"text"*, evento de atribuição de valores dos parâmetros e eventos de nome *"values"* reconhecendo quais primitivas gráficas foram informadas. Foi uma decisão de implementação, utilizar um único nó Lua para implementar todos os *widgets* de um mesmo formulário, passando todos os parâmetros necessários através da propriedade *"values"*. Isto facilita a gerência do foco do usuário na interface dentro do próprio código Lua, facilitando a especificação do código NCL pelo autor do documento.

A função *nclHandler* faz chamadas à função *formParser*, que recebe como parâmetros o valor da *string* com todos os parâmetros e uma *string* indicando qual seu *widget* correspondente. A função *formParser* irá criar efetivamente as metatabelas para cada elemento, separando os parâmetros específicos de cada um.

Após serem criadas as metatabelas de cada elemento, os métodos de cada estrutura podem ser executados pelos componentes de controle e desenho da API, a fim de exibir o resultado na tela. Cada elemento tem sua própria estrutura, com métodos próprios que farão uso dos parâmetros informados ainda no documento NCL.

É importante, ainda, destacar a função *redraw* do arquivo NCLForms.lua, responsável por executar o método *redraw* da metatabela de cada elemento. Esse método é utilizado para atualizar a tela com as interações provocadas pelo usuário como: mudança de um elemento para o outro ou escolha de um item do elemento que está em foco.

As soluções desenvolvidas não ficam restritas ao uso apenas como biblioteca de código em aplicações interativas com formulários. Os elementos de interface gráfica que compõem a API NCLForms (i.e., primitivas gráficas) podem ser aplicados na extensão da linguagem de *patterns* proposta por [3]. Pode-se fazer o download da API acessando a página http://www.midiacom.uff.br/nclforms.

5. CONCLUSÃO

A partir do estudo inicial das linguagens NCL e NCLua, a dificuldade de implementar elementos de interface gráfica de modo simplificado inspirou a decisão da proposta de uma API voltada para o desenvolvimento de GUIs.

A API NCLForms, desenvolvida para o ambiente Ginga-NCL, que viabiliza a construção de GUIs para aplicações interativas em TV Digital, representa a principal contribuição do presente trabalho.

Dentro do cenário de aplicações interativas de TV Digital, levando em consideração as orientações de *design patterns* de interação, a API NCLForms é uma proposta para incentivar a reutilização de software no ambiente Ginga-NCL. Em termos de trabalhos futuros, tal proposta é passível de adição de novas funcionalidades, como a implementação de um teclado virtual a ser exibido na tela e o desenvolvimento de novos *widgets*. Do ponto de vista do que já foi implementado, também seria possível pensar em outro formato para os parâmetros, como diminuir o número de parâmetros obrigatórios ou incluir valores *default* para algumas variáveis.

6. REFERÊNCIAS

[1] ABNT. *Data coding and transmission specification for digital broadcasting - Part 2: Ginga-NCL for fixed and mobile receivers - XML application language for application coding*. ABNT NBR 15606-2:2011 standard, 2011.

[2] D. Kawrykow and M. P. Robillard. Improving API Usage through Automatic Detection of Redundant Code. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, pages 111–122. ASE, 2009.

[3] T. Kunert. *User-Centered Interaction Design Patterns for Interactive Digital Television Applications*. Springer-Verlag, London, 2009.

[4] NCLEclipse. NCL Eclipse. Disponível em: <http://laws.deinf.ufma.br/ncleclipse>, 2012. Acesso em: 07 jan 2012, 16:00:00.

[5] Portal. Portal do software público brasileiro - Ginga. Disponível em: <http://www.softwarepublico.gov.br>, 2012. Acesso em: 07 jan 2012, 17:00:00.

[6] M. P. Robillard. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Software*, 26(6):27–34, November 2009.

[7] L. F. G. Soares and S. D. J. Barbosa. *Programando em NCL 3.0: Desenvolvimento de Aplicações para o Middleware Ginga, TV Digital e Web*. Elsevier Editora, Rio de Janeiro, 2009.

[8] L. F. G. Soares, R. F. Rodrigues, and M. F. Moreno. Ginga-NCL: The Declarative Environment of the Brazilian Digital TV System. *Journal of the Brazilian Computer Society*, 12(4):37–46, 2007.

[9] SourceForge. Akdebugger. Disponível em: <http://sourceforge.net/projects/akdebugger/>, 2012. Acesso em: 07 jan 2012, 16:00:00.

[10] TheEclipseFoundation. Eclipse. Disponível em: <http://www.eclipse.org/>, 2012. Acesso em: 07 jan 2012, 16:00:00.

[11] VmwarePlayer. Vmware Player. Disponível em <http://www.vmware.com/products/player/>, 2012. Acesso em: 07 jan 2012, 16:00:00.

[12] L. Wang, A. Sajeev, and L. Inchaiwong. A Formal Specification of Interaction Widgets Hierarchy Framework. *ITNG'06*, pages 658–664, 2006.