Estendendo a NCL para promover interatividade vocal em aplicações Ginga

Lucas Carvalho
Departamento de Computação - UFS
lucasamcc@dcomp.ufs.br

Hendrik Macedo
Departamento de Computação - UFS
hendrik@ufs.br

ABSTRACT

Vocal access to TVDi content can promote social and digital inclusion. Actually, vocal interface enhances accessibility since it enables TVDi usage by physical-impaired people and blind community. In the scientific literature, however, TVDi voice-driven interactivity is not often considered. There are just few works concerning architecture proposals, but none of them is actually properly validated. We consider the integration of VoiceXML voice gateways and the middleware Ginga a promising hypothesis for architecture definition. In this paper, we describe such an integration proposal and perform validation by means of two different case studies, two different TVDi applications. The proposal key element is the syntax extension we provide to NCL in order to incorporate some native VoiceXML elements. Results show that the vocal interactivity mecanism has achieved similar functionality to that of the remote control.

Categories and Subject Descriptors

I.7.4 [**Document and Text Processing**]: Electronic Publishing; D.2.11 [**Software Engineering**]: Software Architectures; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Audio input/output*

General Terms

Design, Human Factors, Languages

Keywords

Ginga, NCL, Vocal Interaction, Acessibility

RESUMO

Uma forma de se promover inclusão digital e social é a incorporação do acesso vocal ao conteúdo da TVDi. De fato, interfaces vocais potencializam acessibilidade uma vez que possibilitam acesso a portadores de limitações físicas ou visuais. A proposta de inclusão da voz como meio de interatividade com a TVDi, entretanto, ainda é tratada de forma bastante incipiente na literatura científica relacionada. Existem pouquíssimos trabalhos que exibem esse tipo de preocupação e os que o fazem são normalmente relacionados à proposta de arquiteturas, mas sem qualquer compromisso com validação. Consideramos que a integração de gateways de voz que interpretam documentos VoiceXML com o middleware Ginga é uma hipótese promissora de arquitetura. Neste artigo, descrevemos uma proposta de integração nessa linha e a validamos com a utilização de dois estudos de casos distintos, duas aplicações diferentes para TVDi. O elemento central dessa proposta é a extensão da sintaxe da NCL para incorporar elementos nativos mínimos VoiceXML. Foi observado que o mecanismo de interatividade vocal conseguido possui funcionalidade equivalente ao do manuseio do controle remoto.

Extending NCL to support vocal interaction on Ginga applications

1. INTRODUÇÃO

É previsto que até 2016 cerca de 145 milhões de brasileiros passem a utilizar a TV Digital interativa (TVDi) em suas residências. Um dos objetivos da proposta governamental é criar a oportunidade de promover inclusão digital e social com a massificação da tecnologia. Entendemos que um mecanismo importante de acessibilidade e, portanto, uma forma de fomento à inclusão, é a interação por meio de voz ao conteúdo digital oferecido. A comunidade de usuários portadores de limitações físicas ou visuais é diretamente beneficiada por essa forma de acesso. Hoje, essa comunidade soma 14,5% da população brasileira [8].

A literatura científica aborda a questão de maneira ainda tímida e incipiente. Existem alguns trabalhos que tratam de estudos e propostas de interação vocal especificamente para guias de programação de canais (Eletronic Guide Programa - EPG). Entretanto, não é disponibilizada qualquer API ou linguagem de programação vocal para o desenvolvimento de aplicações genéricas. Um desses trabalhos [10] considera um conversor que faz uso de transformações XSL (XSLT) da estrutura de dados do guia de programação em XML para o formato VoiceXML, a ser posteriormente interpretado por um *browser* próprio. Outro trabalho [7] propõe o uso do controle remoto como microfone e, consequente, simplificação do número de botões. Em [4], os autores sugerem a integração de *gateways* de voz genéricos que interpretam documentos VoiceXML com o middleware Ginga como arquitetura de implantação. Na proposta, arquivos VoiceXML

funcionariam como um dos objetos de mídia das aplicações NCL. O problema dessa abordagem é que emissoras de TV devem considerar o envio do conteúdo programático já codificado na linguagem VoiceXML. Além disso, o trabalho não valida apropriadamente a arquitetura. Uma outra proposta sugere o mapeamento automático de elementos NCL para elementos VoiceXML [5]. Neste caso, o problema reside no custo computacional alocado a esse mapeamento, que necessitaria ser realizado a todo instante e o menor controle dos diálogos de voz por parte do desenvolvedor da aplicação.

Apesar dos poucos resultados apresentados até o momento, a integração de gateways de voz com o Ginga parece promissora como hipótese plausível de arquitetura. Neste artigo, propomos e validamos uma arquitetura que promove a integração de gateways de voz com o middleware Ginga. Esta arquitetura considera a API TeouVi proposta em [4]. O elemento chave da abordagem reside na extensão da sintaxe da NCL para incorporar elementos nativos mínimos VoiceXML. A proposta foi fundamentada em alguns princípios que definimos e entendemos serem cruciais para o sucesso de uma integração desse tipo: (i) abstração da estrutura de integração ao desenvolvedor de aplicações, (ii) desacoplamento das funcionalidades de voz, (iii) suporte a comandos semânticos, (iv) independência do gateway de voz utilizado, (v) interface restrita de comunicação VoiceXML-NCL, e (vi) concatenação de diálogos de voz.

À luz desses princípios, o método de trabalho consistiu (1) na definição em nível de componentes da arquitetura de integração, (2) especificação da extensão da linguagem NCL, (3) implementação do formatador da linguagem, (4) escolha criteriosa dos estudos de caso e (5) validação da arquitetura.

A seção 2 descreve as tecnologias de suporte à arquitetura, que é apresentada na seção 3. Os princípios introduzidos mais acima são também explicados nesta seção. A seção 4 descreve em detalhes a extensão da linguagem NCL proposta, apresentando sua formalização através da EBNF e principais funcionalidades. A apresentação dos estudos de caso e validação da arquitetura é realizada na seção 5. Finalmente, a seção 6 traz algumas conclusões e possibilidades de investigações futuras.

2. TECNOLOGIAS DE SUPORTE

A arquitetura de integração proposta está alicerçada em quatro tecnologias principais: a linguagem NCL, o middleware Ginga, a linguagem VoiceXML e os gateways de voz.

2.1 A linguagem NCL e o middleware Ginga

A NCL (*Nested Context Language*) [2] é uma linguagem declarativa baseada em XML [17] e dividida em módulos e perfis, seguindo o modelo NCM (*Nested Context Model*) [13] - modelo conceitual focado na representação e manuseio de documentos hipermídias. O *Basic Digital* TV (BDTV) e *Enhanced Digital* TV (EDTV) são os dois perfis da NCL utilizados para TV Digital. A NCL está particionada em treze áreas de funcionalidades, que ainda se dividem em vários módulos.

Entre as principais características da NCL estão o foco na sincronização espaço-temporal com o uso do elemento <link>, e a interação com o usuário ser tratada apenas como um caso particular da sincronização temporal. A separação entre o conteúdo e a estrutura da apresentação permite à NCL servir apenas como "cola" para os objetos de mídia. Entre os objetos de mídia que podem ser utilizados na aplicação estão objetos de vídeo (MPEG, AVI), objetos de áudio (MP3, MIDI), objetos de texto (HTML, TXT, XML) e outros. Os objetos de mídia suportados dependem dos exibidores de mídias definidos no Ginga-NCL. A NCL ainda disponibiliza um conjunto de comandos que permitem a edição de documentos NCL em tempo de exibição [2]. Estes comandos podem ser executados através de eventos enviados pela emissora ou por aplicações desenvolvidas em Lua [12] ou Java.

A camada de *software* (*middleware*) responsável pela execução das aplicação de TV Digital no Brasil, chama-se Ginga [1]. A arquitetura do Ginga é dividida em três módulos básicos: Núcleo Comum, Ginga-NCL [2] e Ginga-J [3]. O Ginga-J é responsável pela execução das aplicações desenvolvidas em Java, e o Ginga-NCL pela apresentação das aplicações desenvolvidas em NCL e Lua. Uma ponte integra os dois ambientes e permite a comunicação em ambas as direções. No Núcleo Comum localizam-se os decodificadores de conteúdo que servem a ambos os ambientes.

2.2 VoiceXML e gateways de voz

A linguagem VoiceXML [16] é uma linguagem baseada em XML desenvolvida pela W3C (World Wide Web Consortium). Seu objetivo é permitir o desenvolvimento de aplicações de voz de uma maneira simplificada, assemelhando-se ao desenvolvimento de aplicações para Web. A forma de organização de uma aplicação VoiceXML pode ser comparada com a organização de arquivos HTML em um *website*, por exemplo. Da mesma forma que um *website* é formado por um conjunto de páginas HTML que se relacionam através de *hiperlinks*, nas aplicações VoiceXML, uma página principal interage com o usuário e dá acesso às demais páginas de voz da aplicação (cada página é um documento VoiceXML).

Os documentos VoiceXML são enviados para um *gateway* de voz [11] que interpreta e realiza a interação com o usuário. Diferentemente da linguagem NCL, VoiceXML não é dividida em módulos, perfis ou funcionalidades.

O conjunto de elementos presentes na linguagem permite o uso de síntese de voz, digitalização de áudio, reconhecimento de voz e entradas DTMF (*Dual Tone Multi-Frequency*). Toda entrada do usuário é processada pelo *gateway* com a finalidade de realizar o reconhecimento do que foi dito pelo usuário. Atualmente, o uso de um vocabulário específico, limitando o escopo das palavras, é necessário para que se possa conseguir uma boa performance nessa etapa. Há também a necessidade de especificar gramáticas relacionadas ao contexto da aplicação para melhorar o desempenho do reconhecimento.

3. ARQUITETURA DE INTEGRAÇÃO

A arquiteura de integração está fundamentada em seis princípios que entendemos ser cruciais para viabilidade de uma solução para vocalização da forma de acesso à aplicações de TVDi:

- **Abstração da estrutura de integração.** O desenvolvedor de aplicações não deve necessitar conhecer como é realizada a integração do ambiente de voz com o Ginga. A especificação de parâmetros do elemento <object> da linguagem VoiceXML na requisição realizada para o *socket* da API TeouVi ou ainda o código para o botão do controle remoto devem ser abstraídos para o desenvolvedor. A arquitetura também deve conceber a geração automática de códigos repetitivos no documento VoiceXML a partir da extensão de forma a aumentar o nível de abstração do desenvolvimento.

- **Desacoplamento das funcionalidades de voz.** Os elementos da extensão não devem interferir na interpretação e execução dos elementos existentes da NCL. Os elementos da extensão devem ter o único objetivo de fornecer as informações para geração das aplicações VoiceXML, necessárias para a criação das interações vocais com o usuário, havendo sido especificadas para serem executadas durante o tempo de vida da aplicação NCL. Este requisito garante o desenvolvimento sem qualquer alteração das aplicações NCL que, por sua vez, não definam qualquer interatividade vocal.

- **Suporte a comandos semânticos.** Os elementos da extensão devem não somente possibilitar ao desenvolvedor da aplicação especificar o mapeamento dos comandos de voz diretamente para botões do controle remoto (como "vermelho" ou "seta para cima", por exemplo), como também disponibilizar formas mais elaboradas de interatividade por voz utilizando-se de semântica para os comandos vocais como "ver jogo de futebol" ou "encerrar aplicação". Esse requisito foi satisfeito com a utilização de gramáticas.

- **Independência do gateway de voz.** A arquitetura deve ser independente do gateway de voz utilizado e da forma como se processa integração. Caso haja mudança deste componente, o impacto deve ser mínimo (ou nulo). Por esta razão, um adaptador foi desenvolvido.

- **Interface restrita de comunicação VoiceXML-NCL.** A aplicação VoiceXML em execução deve possuir uma única interface de comunicação com a aplicação NCL em execução. Isto é obtido através do uso do elemento <object> para criação de uma requisição HTTP à API TeouVi, sendo esta a única forma de comunicação com o Gerenciador de Eventos do Ginga.

- **Concatenação de diálogos de voz.** Quando mais de um diálogo estiver simultaneamente ativo na aplicação, as informações sobre todos os diálogos de voz são mescladas e um único documento VoiceXML é gerado. Um alerta automático sobre as mudanças de opções de interatividade por voz pode ser fornecido ao usuário através do novo documento VoiceXML gerado para cada mudança das opções do diálogo de voz.

Para implementação da arquitetura foram utilizados o emulador do *player* do Ginga-NCL versão 2.1.1 [2], a API TeouVi [4] e dois gateways de voz: o JVoiceXML versão 0.6 [9] e o Prophecy [15]. A TeouVi permite a integração entre o Ginga-NCL e um *gateway* de voz. Essa API é responsável pela troca de mensagens entre os dois ambientes e a geração do diálogo especificado no documento da NCL estendida para VoiceXML em tempo de exibição da aplicação NCL. Nenhum dos dois gateways utilizados possuem suporte para o idioma português. Todos os componentes da arquitetura foram desenvolvidos na linguagem Java, exceto o Prophecy, e executados na mesma máquina local, embora a arquitetura de um *gateway* de voz permita sua execução em uma máquina remota. A figura 1 ilustra o diagrama de componentes da arquitetura.

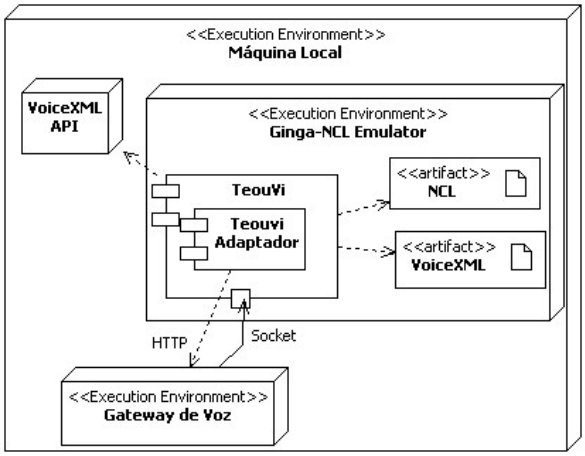

Figura 1: Diagrama de componentes da arquitetura.

A figura 2 ilustra a sequência de métodos executados para a execução de uma interação vocal desde a especificação do diálogo na NCL estendida, que é interpretada pelo Formatador NCL, até as requisições e envios dos documentos VoiceXML para o *gateway* de voz realizar a interação com o usuário.

Para o Formatador original da NCL interpretar os novos elementos da extensão, foram implementadas alterações em seu módulo Conversor:

1. Criação do conversor para a funcionalidade *Voice Interaction* criada com a extensão da linguagem; e
2. Alterações nos conversores das funcionalidades *Linking* e *Structure*.

A API TeouVi recebe os eventos resultantes da interação de voz com o usuário através de uma requisição HTTP enviada pelo *gateway* quando o elemento <object> da VoiceXML é interpretado e encaminha para o módulo Gerenciador de Eventos do Ginga. Um *socket* no componente TeouVi foi implementado para este fim.

4. EXTENSÃO DA NCL

O ponto central desta proposta diz respeito à extensão da linguagem NCL para incorporar elementos VoiceXML de modo a permitir vocalização através de um gateway de voz. A extensão foi realizada na versão 3.0 da NCL e na versão 2.0 da VoiceXML.

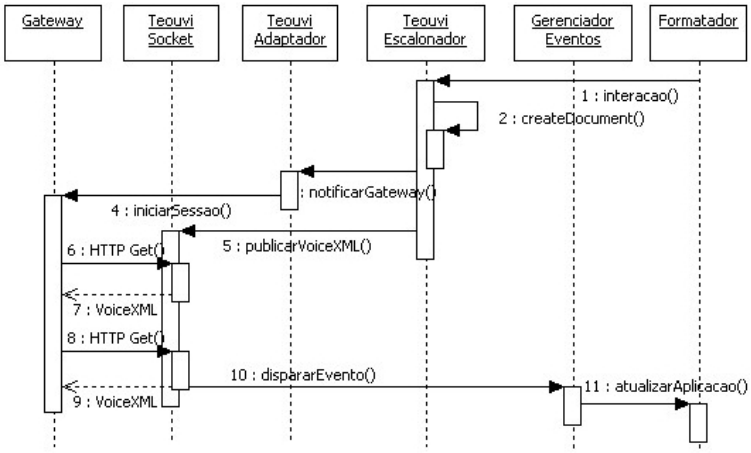

Figura 2: Diagrama de seqüencia geral para execução de uma interação vocal.

4.1 Descrição dos elementos da extensão

A primeira etapa do processo de extensão procurou reproduzir as funcionalidades ou eventos que possam ser efetuados diretamente por meio do controle remoto para comandos vocais. Como exemplo, permitir ao usuário navegar em um guia de programação utilizando comandos vocais simples como "direita", "cima", "um", "dois", "vermelho", "amarelo" e outros disponíveis no controle remoto.

Para definição do diálogo de voz, o elemento <link> da NCL, responsável pela sincronização das mídias e conseqüentemente da interação com o usuário, foi escolhido para incorporar os novos elementos da NCL que especificam a interatividade vocal. Com isso, ao mesmo tempo que são disponibilizadas as opções de interação pelo controle remoto, também são gerados os diálogos de voz que possibilitam esta mesma interação por meio da voz. Como exemplo destes elementos vocais, temos o <menu> que é pai de um ou mais elementos <choice> que por sua vez definem as opções vocais.

O diálogo de voz definido no elemento <link> é ativado quando o objeto de mídia observado por ele está no estado **ocorrendo** e este <link> aguarda uma interação do usuário através do controle remoto. Portanto, o diálogo de voz precisa ser gerado antes da ativação do elemento <link> da NCL, que se torna ativo apenas quando a condicional associada a este elemento é satisfeita.

Esta diferença ocorre devido ao fato de que é o elemento <link> que fornece as opções de interação disponíveis quando determinado objeto de mídia está no estado **ocorrendo**, e se este elemento for ativado, significa que aquela opção de interação provavelmente já ocorreu e não é mais possível naquele momento.

As gramáticas que definem as opções e regras de navegação são criadas através dos elementos incorporados da VoiceXML: <grammar>, <rule>, <one-of> e <item>.

Os demais elementos importados possuem como objetivo:

- definição de como o texto será sintetizado (<prompt>, <break>, <voice>, <emphasis>, <say-as>, <reprompt>);
- manipulação de variáveis (<var>, <value>, <field>); e
- controle condicional (<if>, <elseif> e <else>).

As principais modificações na sintaxe dos elementos da VoiceXML ocorreram em:

- **<object>** - define apenas dois atributos: *action* e *index*. O atributo *action* especifica o botão do controle remoto mapeado para o comando de voz. O atributo *index* especifica o índice da mídia selecionado diretamente pela voz. Essa seleção é equivalente a utilizar o controle remoto para navegar entre os objetos de mídia, e selecionar o objeto que estiver no foco através da tecla ENTER.
- **<choice>** - define apenas um atributo: *text*. Este atributo especifica um texto optativo que será definido no elemento <block> do documento VoiceXML gerado, para fornecer ao usuário um *feedback* da interação.
- **<script>** - define apenas um atributo: *id*, e de forma obrigatória. Este atributo especifica um identificador para que o elemento <vncl> criado para a extensão possa referenciá-lo e possa adicionar um *script* específico ao documento VoiceXML gerado.

Para especificação da extensão da NCL, houve a necessidade de criação de três novos elementos:

- **<scriptBase>** - define uma base de elementos <script> escritos em *ECMAScript* [6]. Esse elemento foi criado para seguir o padrão da linguagem NCL e reunir os elementos <script> que podem ser reutilizados em mais de um diálogo.
- **<grammarBase>** - define uma base de elementos <grammar>. Esse elemento também foi criado para seguir o padrão da linguagem NCL e reunir os elementos que podem ser reutilizados em mais de um diálogo.

- **<vncl>** - define o início da especificação do diálogo de voz com o uso dos elementos da extensão. Possui o atributo *script* que faz referência a um elemento <script> que será incorporado ao documento VoiceXML gerado.

Cabem ainda algumas considerações a respeito das funcionalidades da VoiceXML que não foram incorporadas na extensão em virtude de sua não aplicação a um ambiente de TV Digital:

- **Tratamento de Eventos e Navegação** - os tratadores de eventos foram restritos aos elementos <noinput> e <nomatch>, visto que a complexidade do documento deve ser bastante restrita, não oferecendo suporte à navegação por outros documentos ou dentro do próprio documento.
- **Controle de telefonia** - todos os elementos relacionados à telefonia foram ignorados.
- **Definição avançada da gramática** - elementos avançados não foram incorporados para facilitar ao usuário a criação das gramáticas.

Os elementos da extensão da NCL foram agrupados em uma nova funcionalidade criada especificamente para este propósito chamada de *Vocal Interaction*. Um novo perfil com suporte à voz denominado *Vocal Interactive Digital TV* (VIDTV) foi também definido. Alterações foram realizadas nas funcionalidades *Linking* e *Structure* da linguagem NCL para incorporação desses novos elementos.

4.2 EBNF

A extensão é especificada através de uma gramática EBNF (*Extended Backus-Naur Form*).

As regras referentes aos elementos da NCL, que recebem como filhos elementos da extensão, estão representadas abaixo.

```
1. ncl = ‘<head>’ head ‘</head>’, ‘<body>’ body ‘</body>’;
2. head = [‘<connectorBase>’ connectorBase ‘</connectorBase>’],
3.         [‘<regionBase>’ regionBase ‘</regionBase>’],
4.         [‘<descriptorBase>’ descriptorBase ‘<descriptorBase>’],
5.         [‘<ruleBase>’ ruleBase ‘<ruleBase>’],
6.         [‘<scriptBase>’ script ‘</scriptBase>’],
7.         [‘<grammarBase>’ grammar ‘</grammarBase>’];
8. body = {body_e};
9. body_e =  media | switch | link | port | context;
10. link = {bind} | [vncl];
```

O trecho da EBNF abaixo representa o elemento <script> filho do elemento <scriptBase> e que define os *scripts* implementados na linguagem ECMAScript.

```
11.  script = {‘<script id="’IdSymbol‘">’ string ‘</script>’};
```

A regra da linha 12 abaixo apresenta o elemento que define um bloco de elementos da extensão para criação de um diálogo de voz. As regras 14 a 20 definem o menu de opções e o tratamento de erros no diálogo, respectivamente. Na regra 21, o elemento <object> é representado com seus dois únicos atributos.

```
12. vncl = ‘<vncl id="’IDSymbol’" script="’ IdSymbol ‘">’
13.              [menu] ‘</vncl>’;
14. menu = ‘<menu id="’IDSymbol’"’ [’src="’ string ’"] ‘>’;
15.              [prompt] choice [nomatch] noinput ‘</menu>’;
16. nomatch = ‘<nomatch>’ [string] [reprompt] ‘</nomatch>’;
17. noinput = ‘<noinput>’ [string] [reprompt] ‘</noinput>’;
18. choice = ‘<choice text="’ string ‘">’;
19.              [prompt] [grammar] [field] [object] [value] [if]
20            ‘</choice>’ [choice];
21. object = {‘<object action="’string‘" index="’IDSymbol‘" />’};
```

As regras 22 a 31 abaixo estão definidos os elementos para a construção de gramáticas.

```
22. grammar =  inGram | exGram;
23. exGram = ‘<grammar src="’string’" scope="’GramScopeSymbol’"/>’;
24. inGram = ‘<grammar id="’IDSymbol’" scope="’GramScopeSymbol’">
25.                            ‘validGram’ </grammar>’;
26. GramScopeSymbol = ‘document’ | ‘dialog’;
27. validGram = {‘<rule scope="’RuleScopeSymbol’" id="’IDSymbol’">’
28.                           [oneof] item ’</rule>’};
29. oneof ::= {‘<one-of tag="’string’">’ item ’</one-of>’};
30. RuleScopeSymbol = ‘public’ | ‘local’;
31. item = {‘<item>’ string [item] [one-of] ‘</item>’};
```

As regras 32 à 46 abaixo são definidos os elementos de formatação de diálogo.

```
32. prompt = ‘<prompt>’ string [speech] [reprompt] ’</prompt>’;
33. speech = break, [speech] | emphasis, [speech] | voice,
34. [speech] | say-as [speech];
35. break = ‘<break size="’BreakSymbol’" />’;
36. BreakSymbol = ‘none’ | ‘x-weak’ | ‘weak’ | ‘medium’
37.               | ‘strong’ | ’x-strong’;
38. emphasis = ‘<emphasis level="’LevelSymbol’">’
39.              string ‘</emphasis>’;
40. LevelSymbol = ‘strong’ | ‘moderate’ | ‘none’ | ‘reduced’;
41. voice = ‘<voice gender="’GenderId’" age="’Number’"
42.              variant="’Number’">’ string ‘</voice>’;
43. GenderId = ‘male’ | ‘female’ | ‘neutral’;
44. reprompt = ‘<reprompt />’;
45. say-as = ‘<say-as interpret-as="’string’" [’format="’
46.              string ’"’] ’>’ [value] [string] ‘</say-as>’;
```

As regras 47 a 55 abaixo são definidos os elementos para criação e gerenciamento de variáveis e de estruturas de controle.

```
47. var = ‘<var name="’IDSymbol’"’ [’expr="’string’"’] ‘/>’
48. value = ‘<value expr="‘string’" />’
49. field = ‘<field name="‘IDSymbol’">’
50.            prompt [grammar] [nomatch] [noinput] ‘</field>’
51. if = ‘<if cond=" ’string‘ ">’ {value} {field}
52.           {object} {prompt} {if} {elseif} [else] ‘</if>’;
53. elseif = ‘<elseif cond=" ’string’ "/>’;
54.           {value} {field} {object} {prompt} {if} {elseif};
55. else = ‘<else/>’ {value} {field} {object} {prompt} {if};
```

As últimas regras representadas nas linhas 56 a 60 abaixo definem os não terminais da EBNF que tratam literais, números, elementos booleanos e identificadores.

```
56. number_e = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9;
57. Number = {number_e}
58. Bool = ’yes’ | ’no’;
59. IDSymbol = (qualquer símbolo XML válido);
60. string = (qualquer dado XML);
```

5. VALIDAÇÃO DA ARQUITETURA

Os dois estudos de caso foram obtidos no repositório de aplicações interativas para TV Digital chamado Clube NCL [14].

5.1 Viva Mais

Esse exemplo é um dos primeiros programas brasileiros de TV produzidos com foco voltado à interatividade do telespectador. O programa discute vários assuntos em torno da saúde e bem-estar e oferece algumas oportunidades para uma participação ativa do telespectador.

A interatividade pede ao telespectador que escolha entre 4 diferentes pratos de comida (ver figura 3). Após a escolha do prato, o telespectador é informado sobre a qualidade da sua escolha, dizendo se há nutrientes ausentes ou mesmo em excesso.

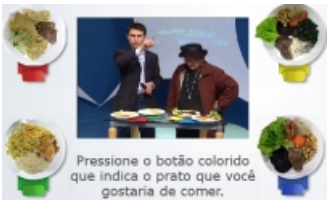

Figura 3: Opção de interatividade na escolha entre os pratos disponíveis.

A primeira interação da aplicação é realizada através do botão vermelho do controle remoto para ativar a parte interativa do programa. O código NCL juntamente com os elementos vocais estão representados no código abaixo. A linha 3 especifica o botão do controle remoto disponível para interação, bem como a linha 15 especifica as palavras disponíveis na gramática para interação vocal.

```
1. <link xconnector="onKeySelectionSetResizeStartStop">
2.   <bind component="icone" role="onSelection">
3.     <bindParam name="keyCode" value="RED"/>
4.   </bind>
5.         ...
6.  <bind component="..." interface="poEntPratos" role="start"/>
7. <bind component="icone" role="stop"/>
8   <vncl>
9.    <prompt>
10.      If you want to interact with the TV show, please say:
11.      <break size="medium"/>Interact   <break size="small"/>
12.                            or Red    <break size="small"/>
13.    </prompt>
14.    <choice text="Start Interaction">
15.       <grammar type="text/gsl">[red interact yes]</grammar>
16.       <object action="RED" />
17.    </choice>
18.    <nomatch>Sorry, please say again.</nomatch>
19.  </vncl>
20. </link>
```

O código abaixo representa o documento VoiceXML gerado a partir do código da NCL acima para o diálogo vocal. A gramática representada na linha 9 define as palavras que podem ser utilizadas pelos usuários (“yes”, “interaction” e “red”).

```
1.  <var name="button"></var>
2.  <menu id="menu" name="menu">
3.    <prompt>
4.      If you want to interact with the TV show, please say:
5.      <break size="medium"/>Interact  <break size="small"/>
6.                            or RED    <break size="small"/>
7.    </prompt>
8.    <choice next="#form1">
9.     <grammar type="text/gsl">[yes interation red]</grammar>
10.   </choice>
11.   <nomatch>Sorry, please say again.</nomatch>
12. </menu>
13. <form id="form1">
14.    <block>Starting Interaction
15.      <assign expr="10" name="button"></assign>
16.      <object next="http://.../event" namelist="button" />
17.    </block>
18. </form>
```

O código a seguir define a segunda interação do programa que possibilita ao usuário a escolha entre 4 opções de pratos diferentes. A escolha pode ser feita a partir das quatro cores disponíveis no controle remoto (“vermelho”, “azul”, “verde” e “amarelo”). Para a interação vocal foi disponibilizado a opção para o usuário escolher o prato de acordo com sua característica principal: “Dish with meat and fried potatoes” / prato com carne e batata frita e “dish with vegetables and fish” / prato com vegetais e peixe.

Note que é definido um elemento <link> para cada opção de interação através do controle remoto neste exemplo (somente exibido dois dos elementos, por questão de espaço). Um diálogo de voz a partir do elemento <vncl> é também especificado como filho de cada elemento <link>. A arquitetura trata de unir as opções de diálogo em um único documento VoiceXML.

```
1.  <link xconnector="onKeySelectionStartStop">
2.   <bind component="prato2" role="onSelection">
3.     <bindParam name="keyCode" value="YELLOW"/>
4.   </bind>
5.         ...
6   <vncl>
7.   <prompt>
8.      To choose the dish with meat and fried potatoes say:
9.      <break size="medium"/>   Yellow  <break size="small"/>
10.           or meat and french fries  <break size="small"/>
11.   </prompt>
12.   <choice text="Dish with meat and fried potatoes.">
13.    <grammar type="text/gsl">[yellow meat french fries]</grammar>
14.    <object action="YELLOW" />
15.   </choice>
16.     <nomatch>Sorry, please say again.</nomatch>
17.     <nomatch>Please, say something.</nomatch>
18.  </vncl>
19. </link>
20. <link xconnector="onKeySelectionStartStop">
21.   <bind component="prato3" role="onSelection">
22.     <bindParam name="keyCode" value="GREEN"/>
23.   </bind>
24.       ...
25.   <vncl>
26.     <prompt>
27.       To choose the dish with vegetables and fish, say:
28.       <break size="medium"/>Green   <break size="small"/>
29.       or meat and fried potatoes    <break size="small"/>
30.    </prompt>
31.    <choice text="Dish with meat and fried potatoes.">
32.     <grammar type="text/gsl">[green vegetables fish]</grammar>
33.      <object action="GREEN" />
34.    </choice>
35.   </vncl>
36. </link>
```

Abaixo está o código do documento VoiceXML gerado a partir da NCL acima. O diálogo foi integrado em um único elemento <prompt>.

```
1.  <var name="button"></var>
2.  <menu id="menu" name="menu">
3.    <prompt>
4.      To choose the dish with vegetables and fish, say:
5.      <break size="medium"/> Green   <break size="small"/>
6.       or vegetables and fish        <break size="small"/>
7   To choose the dish with meat and french fries say:
8.       <break size="medium"/>Yellow  <break size="small"/>
9.       or meat and fried potatoes   <break size="small"/>
10 ...
11.     </prompt>
12.  <choice next="#form1">
13.   <grammar type="text/gsl">[yellow meat french fries]</grammar>
14.  </choice>
15.  <choice next="#form1">
16.   <grammar type="text/gsl">[green vegetables fish]</grammar>
17.  </choice>
18. ...
19.   <nomatch>Sorry, please say again.</nomatch>
20.   <noinput>Please, say something or say repeat.</noinput>
21. </menu>
22. <form id="form1">
23.    <block>Dish with vegetables and fish.
24.       <assign expr="403" name="button"></assign>
25.    </block>
26.    <object next="http://ip:9000/event" namelist="button"/>
27. </form>
28. <form id="form2">
29.    <block>Dish with meat and fried potatoes.
30.       <assign expr="404" name="button"></assign>
31.   <object next="http://ip:9000/event" namelist="button"/>
32.    </block>
33. </form>
```

5.2 Roteiro do Dia

Neste exemplo, o telespectador é chamado para, ativamente, criar sua própria linha narrativa da história. Desde o início até o final do programa, é possível escolher quais os próximos passeios a serem realizados no decorrer de uma visita turística à cidade do Rio de Janeiro (ver figura 4). O roteiro prevê a possibilidade de que o telespectador não esteja disposto a interagir (ou escolher qualquer passeio). Neste caso, o telespectador pode explicitar o fato de forma ativa no início do programa. Neste caso, ele será redirecionado para um roteiro de visita escolhido previamente pelos autores do programa.

Figura 4: Opção de interatividade entre roteiros turísticos.

O código abaixo define as opções para ativação (“Sim” ou “Não”) da interatividade para escolhas dos caminhos que se deseja percorrer no guia. Pelo controle remoto o usuário necessita fazer uso das teclas direcionais “para cima” e “para baixo” para trocar entre as duas opções de escolha. Na interatividade vocal, é facultado o uso das palavras “yes” / sim, “interaction” / interação ou “no” / não. Nesta especificação do diálogo, por desejo dos autores, apenas um dos elementos <link> recebe o elemento vocal <vncl>. A arquitetura neste exemplo não precisa amarrar mais de um diálogo de voz.

```
1.  <link xconnector="conn#onSelectionStopSetWithDur">
2.    <bind component="botaoSim" role="onSelection"/>
3.    <bind component="botaoNao" role="stop"/>
4.    <bind component="noSettingsInicio" interface="proxVdo" role="set">
5.      <bindParam name="val" value="ctxInteragir"/>
6.      <bindParam name="dur" value="0"/>
7.    </bind>
8.    <vncl>
9.      <prompt>
10.       You have the following options.<break size="medium"/>
11.       Yes <break size="small"/>
12.       or No <break size="small"/>
13.     </prompt>
14.     <choice text="Starting interaction">
15.       <grammar type="text/gsl">[yes interaction]</grammar>
16.       <object action="ENTER" index="1" />
17.     </choice>
18.     <choice text="No interaction.">
19.       <grammar type="text/gsl">[no]</grammar>
20.       <object action="ENTER" index="2"/>
21.     </choice>
22.     <nomatch>Sorry, please say again.</nomatch>
23.   </vncl>
24. </link>
25. <link xconnector="conn#onSelectionStopSetWithDur">
26.   <bind component="botaoNao" role="onSelection"/>
27.   <bind component="botaoSim" role="stop"/>
28.   <bind component="noSettingsInicio" interface="proxVdo" role="set">
29.     <bindParam name="val" value="ctxNaoInteragir"/>
30.     <bindParam name="dur" value="0"/>
31.   </bind>
32. </link>
```

O código abaixo permite ao usuário escolher o roteiro que deseja (“Copacabana” ou “Central do Brasil”). Assim como o diálogo passado, ele deveria navegar com as setas direcionais para fazer a escolha da opção desejada. Porém, o uso do diálogo vocal permite a seleção direta da opção desejada a partir das opções definidas na gramática da aplicação. Novamente foi definido o diálogo (elemento <vncl>) em somente um elemento <link>.

```
1.  <link xconnector="conn#onSelectionStopSetWithDur">
2.    <bind component="botaoCentral" role="onSelection"/>
3.    <bind component="botaoCopa" role="stop"/>
4.    <bind component="noSettingsI" interface="proxVdo" role="set">
5.    <bindParam name="val" value="C2"/>
6.      <bindParam name="dur" value="0"/>
7.    </bind>
8.    <vncl>
9.      <prompt>
10.       You have the following options.<break size="medium"/>
11.       Go to:<break size="small"/>
12.       Copacabana <break size="small"/>
13.       or Central do Brasil <break size="small"/>
14.     </prompt>
15.     <choice text="Copacabana">
16.       <grammar type="text/gsl">[beach copacabana]</grammar>
17.       <object action="ENTER" index="3" />
18.     </choice>
19.     <choice text="central do brazil">
20.       <grammar type="text/gsl">[downtown central do brazil
21.     train station subway]</grammar>
22.       <object action="ENTER" index="4"/>
23.     </choice>
24.     <nomatch>Sorry, please say again.</nomatch>
25.     <noinput>Please, say something.</noinput>
26.   </vncl>
27. </link>
28. <link xconnector="conn#onSelectionStopSetWithDur">
29.   <bind component="botaoCopa" role="onSelection"/>
30.   <bind component="botaoCentral" role="stop"/>
31.   <bind component="noSettingsI" interface="proxVdo" role="set">
32.     <bindParam name="val" value="C3"/>
33. <bindParam name="dur" value="0"/>
34.   </bind>
35. </link>
```

O documento VoiceXML gerado contempla o diálogo vocal para a interação desejada, especificado no código abaixo.

```
1.  <var name="button"></var>
2.  <var name="index"></var>
3.  <menu id="menu" name="menu">
4.    <prompt>
5.      You have the following options.<break size="medium"/>
6.      Go to:               <break size="small"/>
7.      Copacabana           <break size="small"/>
8.      or Central do Brasil <break size="small"/>
9.     </prompt>
10.    <choice next="#form1">
11.        <grammar type="text/gsl">[beach copacabana]</grammar>
12.    </choice>
13.    <choice next="#form2">
14.        <grammar type="text/gsl">[downtown central do brazil
15.     train station subway]</grammar>
16.    </choice>
17.    <nomatch>Sorry, please say again.</nomatch>
18. </menu>
19. <form id="form1">
20.   <block>Starting Interaction
21.     <assign expr="3" name="button"></assign>
22.     <assign expr="1" name="index"></assign>
23.     <object next="http://ip:9000/event" namelist="button index"/>
24.   </block>
25. </form>
```

6. CONCLUSÃO

Este trabalho demonstrou a possibilidade de utilização de elementos da linguagem VoiceXML para estender a linguagem NCL e possibilitar a definição de diálogos de voz como mais uma forma de interatividade na TV Digital brasileira. A extensão da NCL foi completamente especificada. Dois estudos de caso foram utilizados para validar a arquitetura da solução. A interatividade vocal observada em ambas aplicações mostram a viabilidade da utilização dessa mídia para promover acessibilidade à TVDi. Ficou ainda patente a facilidade de utilização da extensão da NCL para a definição dos diálogos de voz por parte do desenvolvedor.

Uma limitação da proposta é a falta de especificação de comandos de edição em tempo de exibição, disponíveis para a NCL padrão. Isto deve ser incorporado também para os elementos da extensão.

Interatividade vocal no ambiente do Ginga-J não foi contemplada neste trabalho e deve ser abordada em trabalhos futuros. Pode-se considerar inicialmente a própria geração de documentos VoiceXML com a API TeouVi ou a utilização da Java Speech API para a construção dos diálogos.

Outros trabalhos futuros são a sistematização de testes de usabilidade e a definição de um guia de boas práticas para desenvolvimento de interfaces para aplicações de TV Digital com o uso da extensão vocal. Finalmente, o suporte a comandos vocais em objetos xHTML e NCLua também é foco de futura investigação.

7. REFERÊNCIAS

[1] ABNT NBR 15606-1:2007. *Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 1: Codificação de dados*, 2007.

[2] ABNT NBR 15606-2:2007. *Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 2: Ginga-NCL para receptores fixos e móveis - Linguagem de aplicação XML para codificação de aplicações.*, 2007.

[3] ABNT NBR 15606-4:2010. *Televisão digital terrestre - Codificação: transmissão para radiodifusão digital Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais.*

[4] L. A. M. C. Carvalho, A. P. Guimarães, and H. T. Macedo. Architectures for interactive vocal environment to the brazilian digital tv middleware. In *Proceedings EATIS (Euro American Conference on Telematics and Information Systems) 2008*, pages 105–112, 2008.

[5] L. A. M. C. Carvalho and H. T. Macedo. Geração automática de interações vocais na tvdi utilizando o ginga, ncl e voicexml. *X Workshop de Software Livre*, pages 115–120, Junho 2009.

[6] ECMA. *ECMAScript Language Specification.* Dec 1999. URL: http://ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

[7] K. Fujita, H. Kuwano, T. Tsuzuki, Y. Ono, and T. Ishihara. A new digital tv interface employing speech recognition. *International Conference on Consumer Electronics. ICCE.*, pages 356–357, 2003.

[8] IBGE. Censo demográfico 2000 - características gerais da população: Resultados da amostra. 2002.

[9] JVoiceXML. *The Open Source VoiceXML Interpreter.* 2008. URL: http://jvoicexml.sourceforge.net/.

[10] H. Kim and E. Hwang. Voiceepg: Speech interface for electronic program guide. In *Internet and Multimedia Systems and Applications*, 2003.

[11] J. A. Q. Ruiz and J. R. M. Sánchez. Design of a voicexml gateway. *Foufth Mexican International Conference on Computer Science*, 8-12:49–53, September 2003.

[12] F. Sant'Anna, R. Cerqueira, and L. F. G. Soares. Nclua - objetos imperativos lua na linguagem declarativa ncl. *XIV Simpósio Brasileiro de Sistemas Multimídia e Web - Webmedia 2008*, pages 83–90, 26 a 29 de outubro 2008.

[13] L. F. G. Soares and R. F. Rodrigues. Nested context model 3.0: Part 1 - ncm core. Technical Report 12, Departamento de Informática. PUC-Rio., Apr 2005. URL: http://ncl.org.br/documentos/ncm30-port.pdf.

[14] Telemídia. *Clube NCL.* 2010. URL: http://clube.ncl.org.br.

[15] Voxeo. *Prophecy.* URL: http://voxeo.com/prophecy/.

[16] W3C. *Voice Extensible Markup Language (VoiceXML) Version 2.0.* Mar. 2004. URL: http://w3.org/TR/voicexml20.

[17] W3C. *Extensible Markup Language (XML) 1.0.* Nov. 2008. URL: http://w3.org/TR/xml/.