UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Instituto de Ciências Exatas
Departamento de Ciência da Computação

WEB COMPOSER: uma proposta de ferramenta web para autoria e execução de aplicações de TV Digital.

Diogo Cesar Souza Lopes

JUIZ DE FORA
Março, 2013

UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Instituto de Ciências Exatas
Departamento de Ciência da Computação

WEB COMPOSER: uma proposta de ferramenta web para autoria e execução de aplicações de TV Digital.

Diogo Cesar Souza Lopes

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Orientador: Marcelo Ferreira Moreno

JUIZ DE FORA
Março, 2013

WEB COMPOSER: uma proposta de ferramenta web para autoria e execução de aplicações de TV Digital.

Diogo Cesar Souza Lopes

MONOGRAFIA SUBMETIDADA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

_____

Prof. Marcelo Ferreira Moreno – Orientador

Dr. em Informática, PUC/RJ, 2008.

_____

Prof. José Maria Nazar David

Dr. em Eng. de Sis. e Comp., COPPE/UFRJ, 2004.

_____

Prof. Jairo Francisco de Souza

Dr. em Informática, PUC/RJ, 2012.

JUIZ DE FORA, MG

Março, 2013

RESUMO

Este projeto tem como objetivo principal a apresentação do desenvolvimento de um aplicativo web, denominado Web Composer NCL, que permite aos usuários a criação e execução de aplicações para a TV Digital, assim como possibilita a exportação do código NCL proveniente da criação do projeto.

ABSTRACT

This project's main objective is presenting the development of a web application, called Web Composer NCL, which allows users to create and run applications for digital TV, as well as enables the export of NCL code from the project creation.

AGRADECIMENTOS

Ao Orientador, Professor Marcelo Ferreira Moreno, pelo incentivo, ensinamentos e presteza no auxílio às atividades e discussões sobre o andamento desta Monografia.

A minha vó Maria da Conceição, pelo seu eterno carinho e amor incondicional.

A minha irmã Thainara Lopes, pelo companheirismo e amizade.

A minha namorada e companheira Daniela Leonel, pela paciência, tolerância e pelo fundamental apoio à conclusão deste projeto.

Aos meus pais, que mesmo ausentes são sempre onipresentes em minhas decisões.

A todos os meus parentes, amigos e aqueles que, direta ou indiretamente, colaboraram para que este trabalho conseguisse atingir aos objetivos propostos.

E, finalmente, a DEUS pela oportunidade e pelo privilégio que me foi dado.

SUMÁRIO

LISTA DE SIGLAS

Sigla	Descrição
ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
ARIB	Association of Radio Industries and Businesses
ATSC	Advanced Television Systems Committee
BD	Banco de dados
COFDM	Coded Orthogonal Frequency Division Multiplexing
DASE-DTV	Application Software Environment
DVB	Digital Video Broadcasting
DVB-T	Digital Video Broadcasting Terrestrial
EDTV	Enhanced Definition Television
GINGA CC	Ginga Commom Core
GINGA J	Ginga Java
GINGA NCL	Ginga Nested Context Language
HDTV	High Definition Television
IRD	Integrated Receiver Decoder
ISDB	Integrated Service Digital Broadcasting
ISDB-T	Integrated Services Digital Broadcasting Terrestrial
LAWS	Laboratory of Advanced Web Systems
LDTV	Low Definition Television
MHEG	Multimedia and Hypermedia Expert Group
MHP	Multimedia Home Plataform
MIT	Massachusetts Institute of Technology
MPEG-2	Moving Picture Experts Group
MPEG-4	Moving Picture Experts Group
NCL	Nested Context Language
NCM	Nested Context Model
PDA	Personal Digital Assistants
PUC	Pontifícia Universidade Católica
PUC - Rio	Pontifícia Universidade Católica do Rio de Janeiro
SBTVD	Sistema Brasileiro de TV digital
SDTV	Standard Definition Television
STB	Set Top Box
TV	Televisão
TVDI	TV Digital Interativa
UFPB	Universidade Federal da Paraíba
WEB	World Wide Web
XML	Extensible Markup Language

LISTA DE FIGURAS

1. INTRODUÇÃO

A televisão, um dos mais importantes meios de comunicação, vem se modernizando e propondo maior aproximação com seus telespectadores, assim como outras mídias. Nos anos 1970 cientistas japoneses iniciaram o desenvolvimento de uma TV de alta definição, hoje chamada de HDTV. O que antes era apenas um canal restrito a emitir informação, hoje passa a receber elementos de pessoas que assistem seus canais e estes se tornam ativos nesta comunicação.

A interatividade na TV digital possibilita novas abordagens para produção de notícias, vídeos, programas, entre outros. Para esta finalidade, diversos estudos têm sido realizados com o intuito de possibilitar a difusão e a ampliação da capacidade de criação de mídias para TV Digital.

Existem atualmente diversas ferramentas de autoria para linguagens de marcação baseadas no tempo, em especial, aquelas voltadas à concepção de documentos hipermídia (GUIMARÃES, 2007). Todas essas ferramentas possuem em comum a incapacidade de abstrair ou excluir completamente a autoria textual do processo de desenvolvimento, além de não permitirem o trabalho colaborativo entre usuários. Atualmente, não existe uma única abstração gráfica ou visual capaz de expressar por completo todas as especificidades de uma dessas linguagens.

Um exemplo de tais abstrações é a representação gráfica do paradigma de linha de tempo, comumente usado nesses tipos de ferramentas. Esse paradigma, entretanto, não é satisfatório ou conveniente para modelar a interação do usuário. Primeiro, porque é impossível precisar, ainda em tempo de autoria, o exato instante em que tal interação ocorrerá. Segundo, porque o suporte à interação do usuário resulta em múltiplas linhas do tempo e não em uma única. Na prática, as aplicações interativas devem ser tratadas como uma linha de tempo principal e diversas linhas secundárias, tantas quantas as interações de usuário determinar (COSTA E SOARES, 2007). Resultando assim, em um ambiente confuso ou difícil de representar visualmente.

Diante da ampla gama de possibilidades que a TV Digital proporciona à produção de documentos hipermídia e do conhecimento das dificuldades para a criação de aplicativos para a TVD, através das ferramentas de autoria existentes, o

presente trabalho se propôs a desenvolver um protótipo de uma ferramenta web para autoria de aplicações digitais, que foi intitulada de Web Composer.

O Web Composer tem como objetivo possibilitar ao usuário a criação de aplicativos por meio de uma interface simples e totalmente visual, eliminando a necessidade do conhecimento sobre a linguagem utilizada para autoria de documentos hipermídia, a NCL.

Desenvolvido sobre a plataforma web, estrutura diferente das ferramentas de autoria existentes, o Web Composer visa permitir o trabalho colaborativo entre os usuários, ampliando-se a capacidade de criação e a possibilidade do máximo aproveitamento da criatividade de cada um dos usuários envolvidos no projeto. Além disso, possibilita armazenar informações em banco de dados para garantir o reaproveitamento de projetos criados e mídias carregadas.

Enfim, o Web Composer NCL é um protótipo que suporta inicialmente um subconjunto da NCL para a interação simples entre as mídias, permite a carga e a reutilização de mídias de áudio, vídeo e imagem, além de possuir uma interface simples e sem expressões técnicas.

Com isso, a criação de aplicativos para a TV digital, através dessa ferramenta, possibilita aproximar os usuários do processo criativo, sem necessidade de conhecimento técnico, o que se constitui em algo inovador e desafiador.

2. NCL (NESTED CONTEXT LANGUAGE)

Considerando que o objetivo do Web Composer é permitir aos usuários a criação de aplicações para a TVD sem o conhecimento de NCL e possibilitar a exportação deste código, foi necessário para o desenvolvimento deste projeto o domínio sobre a estrutura e versões desta linguagem. Isto foi feito para que o código NCL exportado pelo Web Composer, quando executado pelo middleware (camada de software intermediária entre os sistemas operacionais e as aplicações), refletisse exatamente a aplicação que foi desenvolvida na ferramenta.

Criada no Laboratório TeleMídia da PUC-Rio, a linguagem NCL - Nested Context Language - é uma linguagem declarativa para autoria de documentos hipermídia baseados no modelo conceitual NCM -Nested Context Model.[1]

[1] Disponível em http://clube.ncl.org.br/node/32, acesso em 11/03/2013.

O modelo da linguagem NCL visa não apenas o suporte declarativo à interação do usuário, mas ao sincronismo espacial e temporal em sua forma mais geral, tratando a interação do usuário como um caso particular. NCL visa também o suporte declarativo a adaptações de conteúdo e de formas de apresentação de conteúdo; o suporte declarativo a múltiplos dispositivos de exibição, interligados através de redes residenciais (HAN – Home Area Networks) ou mesmo em área mais abrangente; e a edição/produção da aplicação em tempo de exibição, ou seja, ao vivo. Como esses são os focos da maioria das aplicações para TV digital, NCL se torna a opção preferencial no desenvolvimento da maioria de tais aplicações. Para os poucos casos particulares, como por exemplo, quando a geração dinâmica de conteúdo é necessária, NCL provê o suporte de sua linguagem de script Lua.

NCL define como objetos de mídia são estruturados e relacionados, no tempo e espaço. Como uma linguagem de cola, NCL não restringe ou prescreve os tipos de conteúdo dos objetos de mídia. Nesse sentido, podemos ter como objetos de mídia NCL: objetos perceptuais de imagem, de vídeo, de áudio, e de texto; objetos com código imperativo (Lua, entre outros); e objetos com código declarativo (XHTML, entre outros), incluindo objetos com código NCL aninhados. Quais objetos de mídia têm suporte depende apenas dos exibidores de objetos de mídia que estão acoplados ao exibidor (player) NCL. Essa definição depende do Sistema de TV Digital onde o Ginga será utilizado. Por exemplo, no caso do ISDB-TB, objetos com código imperativo podem ser Java (XLet) ou Lua (NCLua); objetos com código declarativo podem ser XHTML (com as funcionalidades mínimas definidas pelas Normas do Sistema) ou outras aplicações NCL embutidas.

Lua é a linguagem de script de NCL. Lua é uma linguagem de programação imperativa eficiente, rápida e leve, projetada para estender aplicações. Lua combina uma sintaxe simples para programação imperativa com construções poderosa para descrição de dados baseadas em tabelas associativas e em semântica extensível. Lua é tipada dinamicamente, é interpretada e tem gerenciamento automático de memória, com coleta de lixo incremental. Essas características fazem de Lua uma linguagem ideal para configuração, automação (scripting) e prototipagem rápida (geração rápida de aplicações).

2.1 Versões da NCL

O número de versão de uma especificação NCL consiste em um número principal e outro secundário, separados por um ponto. Os números são representados como uma cadeia de caracteres formada por números decimais, na qual os zeros à esquerda são suprimidos. O número de versão corrente é 3.0 (NCL 3.0).[2]

A primeira versão da NCL (Nested Context Language) foi especificada através de um XML DTD (Document Type Definition). A partir de sua segunda versão, chamada de NCL 2.0, NCL passou a ser especificado através de um XML Schema. Seguindo as recentes tendências de desenvolvimento, NCL foi projetada de forma modular, permitindo a combinação de seus módulos em diferentes perfis de linguagem.

Além da estrutura modular já mencionada, a NCL 2.0 introduziu funcionalidades novas, tais como:

- Definição de conectores hipermídia e bases de conectores;
- Uso de conectores hipermídia para autoria de elos (links);
- Definição de portas e mapas para nós de composição;
- Definição de templates de nós de composição hipermídia e bases de templates de nós de Composição e o uso destes para autoria de nós de composição;
- Refino da especificação do documento com alternativas de conteúdo através do uso do elemento <switch>, que agrupa um conjunto de nós alternativos;
- Refino da especificação do documento com alternativas de apresentação através do elemento <descriptorSwitch>, que agrupa um conjunto de descritores alternativos;
- Uso de um novo modelo de leiaute espacial.

NCL 2.1 trouxe algumas melhorias com relação à versão anterior: um módulo para definir funções de custo associadas à duração de objetos de mídia foi

[2] Disponível em http://www.ncl.org.br/pt-br/versoes, acesso em 11/03/2013.

introduzido; um módulo que descreve a seleção de regras para os elementos <switch> e <descriptorSwitch> foi definido; além de melhorias em nos módulos da linguagem, destacando-se o XTemplate.

NCL 2.2 incorporou algumas pequenas mudanças com relação à versão anterior, relacionadas à definição dos elementos da linguagem, introduzindo uma nova abordagem para definição dos módulos e perfis NCL.

NCL 2.3 introduziu dois novos módulos para o suporte ao reuso de bases e entidades, refinando a definição de alguns elementos de forma a suportar as novas funcionalidades.

NCL 2.4 reviu e refinou o suporte a reuso introduzido na versão 2.3, e a especificação dos elementos <switch> e <descriptorSwitch>. Essa versão também dividiu o módulo Timing introduzido por NCL 2.1, criando um novo módulo para encapsular questões relacionadas com operações de escala de tempo (computação de tempo elástico usando funções de custo temporal) em documentos hipermídia.

A edição NCL 3.0 reviu algumas funcionalidades contidas na NCL 2.4. NCL 3.0 é mais específica no tocante a alguns valores de atributos. Essa nova versão introduz duas novas funcionalidades: navegação através do uso de teclas e funcionalidades de animação. Adicionalmente, NCL 3.0 fez modificações profundas na funcionalidade de template de nó de composição. NCL 3.0 também reestruturou a especificação de conectores hipermídia de forma a possuir uma notação mais concisa. Relacionamentos entre objetos imperativos NCL e outros objetos NCL, também são definidos na versão 3.0, como também o comportamento dos exibidores NCL para objetos imperativos.

A partir da versão 3.0, novas versões da NCL devem ser publicadas de acordo com a seguinte política de versionamento. Se exibidores (players) NCL compatíveis com versões mais antigas ainda puderem receber um documento com base na especificação revisada, com relação a correções de erro, por motivos operacionais, a nova versão da NCL deve obrigatoriamente ser publicada com o número secundário atualizado. Se exibidores (players) NCL compatíveis com versões mais antigas não puderem receber um documento baseado nas especificações revisadas, o número principal deve obrigatoriamente ser atualizado.

Um módulo específico da versão x.y é especificado em:

http://www.ncl.org.br/NCLx.y/modules/moduleName, onde o número de versão é escrito imediatamente após NCL. De modo similar, um perfil específico da versão x.y é especificado em http://www.ncl.org.br/NCLx.y/profiles/profileName.

2.2 Estrutura do código NCL

Assim como qualquer arquivo XML, toda aplicação NCL deve apresentar um cabeçalho XML como primeira linha do arquivo: (SOARES E BARBOSA, 2012).

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

A estrutura básica de uma aplicação NCL é formada pelo elemento <ncl>, e por seus elementos filhos <head> (cabeçalho) e <body> (corpo). O elemento <ncl> possui os atributos *id* e *xmlns*, que identificam a aplicação e o perfil de linguagem utilizado, respectivamente, conforme o seguinte formato:

```
<ncl id="qualquer_cadeia_de_caracteres" xmlns="http://www.ncl.org.br/NCL3.0/nomePerfil">
```

O atributo *id* de um elemento <ncl> é obrigatório e pode ter como valor qualquer cadeia de caracteres que comece com uma letra ou sublinhado ("_") e que contenha apenas letras, dígitos, "." e "_".

O nome do perfil, no caminho do URI do namespace, também é obrigatório e deve ser "EDTVProfile" ou "BDTVProfile", para indicar o perfil Enhanced DTV ou Basic DTV, respectivamente.

O elemento <head> contém bases de elementos referenciados pelo corpo da aplicação NCL (definido no elemento <body>), como as regiões, os descritores, as transições, os conectores e as regras. Também é no elemento <head> que se definem os documentos que podem ser reutilizados pelo documento atual, bem como os metadados que auxiliam na descrição do documento como um todo.

O elemento <body> contém os elementos que definem o conteúdo da aplicação propriamente dita, tais como objetos de mídia, elos, contextos e objetos

switch. Os elementos, atributos e conteúdos que definem a estrutura de documentos NCL no perfil EDTV estão sumarizados na Tabela abaixo.

Elementos	Atributos	Conteúdo
Ncl	id, title, xmlns	(head?, body?)
Head		(importedDocumentBase?, ruleBase?, transitionBase?, regionBase*, descriptorBase?, connectorBase?, meta*, metadata*)
body	Id	(port | property | media | context | switch | link | meta | metadata)*

Tabela 1: Elementos, Atributos e Conteúdo (Elementos Filhos) que Definem a Estrutura de Documentos NCL no Perfil EDTV.

É recomendado que os elementos filhos do elemento <head> sejam declarados na ordem indicada na tabela e ilustrada pela Listagem abaixo. Já os elementos do <body> podem ser definidos em qualquer ordem.

```
<head>
<importedDocumentBase>
<!-- apliações NCL importadas e referenciadas por esta -->
</importedDocumentBase>
<ruleBase>
<!-- regras para adaptar o conteúdo da aplicação e sua forma de apresentação -->
</ruleBase>
<transitionBase>
<!-- efeitos de transição para apresentação dos objetos de mídia -->
</transitionBase>
<regionBase >
<!-- áreas de exibição destinadas aos objetos de mídia -->
</regionBase>
<descriptorBase>
<!-- configurações de apresentação dos objetos de mídias -->
</descriptorBase>
```

```
<connectorBase>
<!-- comportamento dos elos de relacionamento entre objetos -->
</connectorBase>
<meta/> <!-- dados descritivos simples -->
<metadata>
<!-- dados descritivos estruturados -->
</metadata>
</head>
```

Os elementos filhos dos elementos <head> e <body> são definidos em outros módulos, conforme indicado na Tabela abaixo. Os módulos são indicados aqui para facilitar a consulta a outros documentos de especificação da NCL (p. ex., ABNT, NBR, 15606-2, 2011 e ITU-T, H.761, 2011).

Pai	Elemento(lista parcial)	Módulo
head	importedDocumentBase	Import
	ruleBase	TestRule
	transitionBase	TransitionBase
	regionBase region	Layout
	descriptorBase descriptor	Descriptor
	escriptorSwitch	DescriptorControl
	connectorBase	ConnectorBase (e outros)
	meta metadata	Metainformation
body	port	CompositeNodeInterface
	media	Media
	area	MediaContentAnchor
	property	PropertyAnchor
	context	Context
	switch	ContentControl
	switchPort	SwitchInterface
	link	Linking

Tabela 2: Módulos que Definem os Elementos da NCL no Perfil EDTV, Filhos dos Elementos <head> e <body>.

3. ESTRUTURA NECESSÁRIA À EXECUÇÃO DE APLICAÇÕES HIPERMÍDIA.

Para o desenvolvimento do módulo de criação dos aplicativos para TVD e do módulo de exportação do código NCL dos projetos do Web Composer, foi necessário o entendimento da estrutura básica necessária à execução de aplicações hipermídia.

Programas de TV Digital interativa podem ser entendidos como aplicações hipermídia / multimídia. Nesse cenário, sistemas hipermídia irão se constituir em uma das ferramentas mais importantes a serem dominadas. Sistemas de autoria hipermídia são o suporte para a geração de informação, não se restringindo apenas à concepção dos conteúdos em si, mas incluindo também a concepção de como eles devem ser apresentados. Sistemas de exibição hipermídia (núcleo dos chamados *middleware* para TV) são os responsáveis pela apresentação especificada. Todos esses sistemas têm por base alguma linguagem de especificação.

Segundo (RODRIGUES, 2006), conteúdos para TV Digital interativa são usualmente concebidos usando uma linguagem declarativa (aplicações essas que para serem exibidas têm o suporte do chamado *middleware* declarativo), ou uma linguagem imperativa (a linguagem Java predomina e, nesse caso, as aplicações têm o suporte do chamado *middleware* procedural). Além do suporte à criação de conteúdos, o *middleware* tem a função de “virtualizar” os aparelhos de televisão dos diferentes fabricantes, definindo para os que produzem conteúdo uma visão única de plataforma.

Esse papel confere ao *middleware* grande importância, pois é ele quem regula as relações entre duas indústrias estratégicas para o país: a de produção de conteúdo e a de fabricação de aparelhos receptores.

3.1 Sistemas utilizados

Conforme comentando anteriormente, existem na literatura dois tipos de visões: Uma declarativa, onde a programação é mais de alto nível e o programador passa uma sequência de tarefas a serem executadas, sem se preocupar em como o interpretador, compilador ou máquina virtual irão programar estas tarefas. Como exemplo desta visão tem-se o *middleware* Ginga-NCL. A segunda visão é a procedural, onde o programador deve ser mais experiente e trabalhar a baixo nível,

informando cada passo a ser executado pelo computador. Como exemplos têm o **Ginga-J** e o **Java TV**.

3.2 Ginga

Ginga é uma camada intermediária (middleware) para o desenvolvimento de conteúdo interativo para TV Digital. O uso deste middleware permite que o desenvolvimento das aplicações seja independente da plataforma de hardware utilizada nos receptores de TV (Set-Top-Box).

Resultado de anos de pesquisas lideradas pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB), Ginga reúne um conjunto de tecnologias e inovações brasileiras que o tornam a especificação de middleware mais avançada e, ao mesmo tempo, mais adequada à realidade do país.

O middleware Ginga se encaixa entre as aplicações e os outros módulos de hardware que compõem o sistema brasileiro de TVD. Na FIGURA 6 tem-se uma visão estrutural dos blocos que compõem todo o sistema.

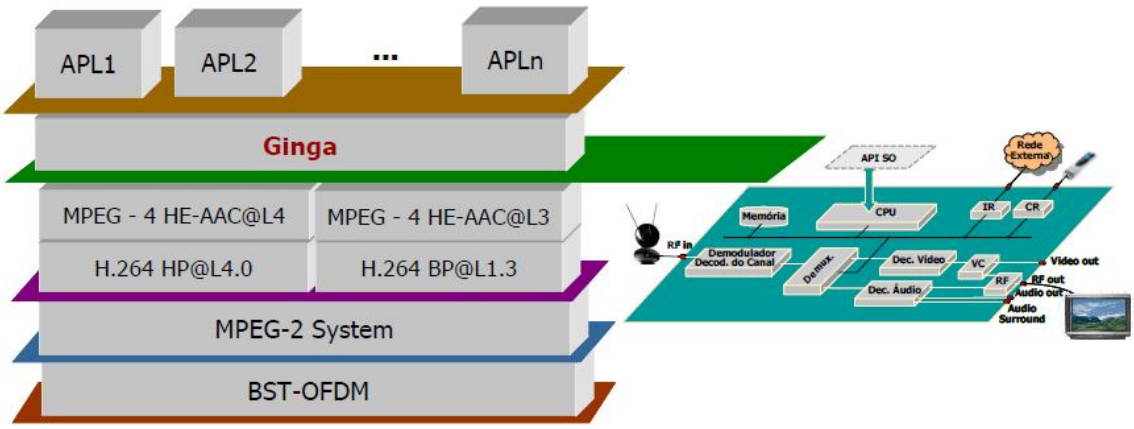


Figura 1 – Padrões de referência do sistema brasileiro de TV Digital terrestre.

[3]

[3] Fonte: SOARES E BARBOSA, 2012.

O *middleware* Ginga pode ser subdividido em três sistemas principais: **Ginga-CC**, **Ginga-J** e **Ginga-NCL**.

O **Ginga-CC** (*Ginga Common-Core*) é o núcleo do sistema e dá suporte para o uso de ambientes declarativos (Ginga-NCL) e ambientes procedurais (Ginga-J).

Dependendo das funcionalidades requeridas na aplicação inicial, um paradigma (declarativo ou procedural) será mais bem adequado que o outro. As linguagens declarativas são de mais alto nível e por isso mais fáceis de usar não dependendo de um *expert* em programação. Porém, uma linguagem declarativa tem um foco específico.

Quando o foco da aplicação é mais dinâmico, o uso de uma linguagem procedural é mais vantajoso. Uma aplicação não necessita ser puramente declarativa ou puramente procedural. Nos sistemas de TVD os dois tipos de aplicação coexistirão e os receptores darão suporte aos dois tipos em seu *middleware*.

Ginga-J foi desenvolvido pela UFPB (Universidade Federal da Paraíba) para dar suporte à execução de aplicações baseadas em linguagem Java, com facilidades voltadas para o desenvolvimento de conteúdo para TV Digital.

O **Ginga-NCL**, desenvolvido pela PUC-RIO (Pontifícia Universidade do Rio de Janeiro) tem como função, o suporte de apresentação de conteúdos hipermídia escritos na linguagem NCL, com facilidades voltadas a interatividade, sincronismo espaço-temporal de objetos de mídia, adaptabilidade e suporte a múltiplos dispositivos. O NCL possui o *script* LUA como sua linguagem de *script*.

A figura a seguir representa a estrutura básica do *middleware* Ginga.

Figura 2 – Estrutura do Ginga.[4]

3.3 Ginga-NCL

Diferente dos outros middlewares utilizados no sistema de TV Digital, o middleware brasileiro Ginga-NCL tem seu ambiente declarativo baseado em uma linguagem XML (Extensible Markup Language), chamado de NCL (Nested Context Language). Essa linguagem foca em como os objetos são estruturados e relacionados no tempo e espaço.

3.3.1 Estrutura básica do NCL

Um documento NCL é um arquivo escrito em XML. Todo documento NCL possui a seguinte estrutura:

- Um cabeçalho de arquivo (linhas 1 e 2);
- Uma seção do cabeçalho do programa (seção head, linhas 3 a 13), onde se definem as regiões, os descritores, os conectores e as regras utilizadas pelo programa;
- O corpo do programa (seção body, linhas 14 a 17), onde se definem os contextos, nós de mídia, elos e outros elementos que definem o conteúdo e a estrutura do programa;
- Pelo menos uma porta que indica por onde o programa começa a ser exibido (portpInicio, linha 15);
- A conclusão do documento (linha 18).

[4] Fonte: NETO, 2013

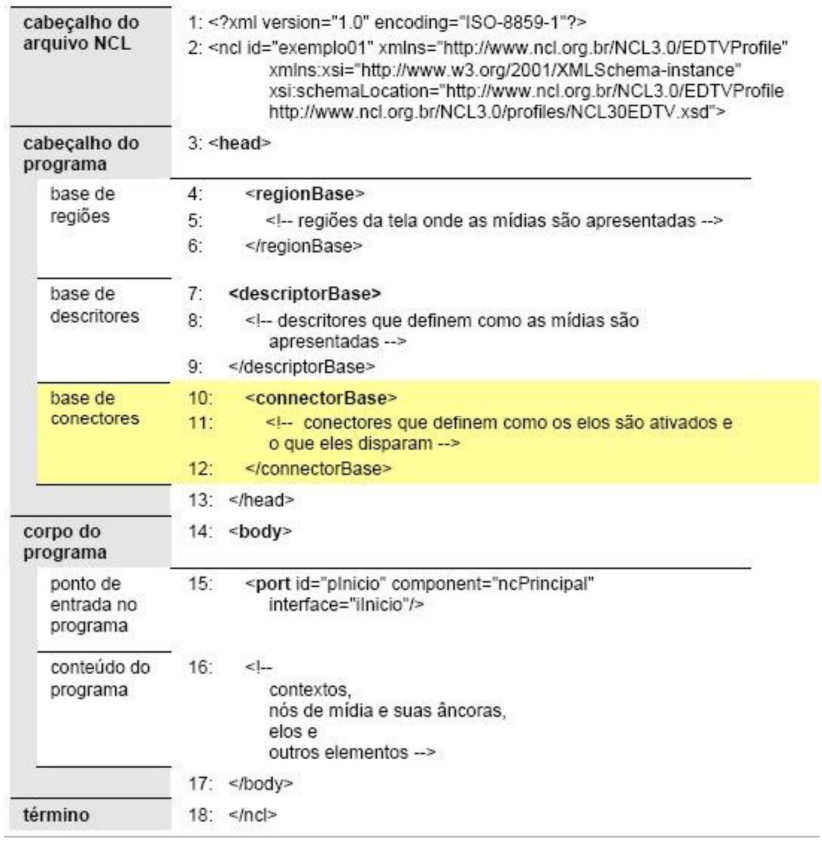


Figura 3 - Estrutura de um programa NCL.[5]

3.3.2 Regiões

Uma região nada mais é do que uma área na tela onde será exibido um determinado nó de mídia. Estas regiões podem ser aninhadas (regiões dentro de regiões), tornando a estrutura organizada. Todas as regiões devem ser definidas no cabeçalho do programa (regionBase). A seguir tem-se um exemplo de definição de uma região: (SOARES E BARBOSA, 2012).

```
<region id="rgTV" width="1920" height="1080">
<region id="rgVideo1" left="448" top="156" width="1024" height="768" />
</region>
```

Os atributos height, width, left e top definem a altura, largura, coordenada esquerda e coordenada superior da região. O atributo id dá um nome único a esta região, nome que depois será referenciado, por exemplo, nos descritores de mídia

[5] Fonte: NETO, 2013

associados a esta região. Pode-se ainda definir os atributos background, que atribui uma cor de fundo, e zIndex, que indica quais regiões aparecerão sobre ela no caso de janelas sobrepostas.

3.3.3 Descritores

Um descritor define como será apresentado um nó de mídia, incluindo em que região ele aparecerá. Os descritores devem ser definidos no cabeçalho do programa (descriptorBase). A seguir tem-se o exemplo de um descritor:

```
<descriptor id="dVideo1" region="rgVideo1" />
```

O atributo id, como nas regiões, fornece um nome único ao descritor, que será referenciado quando da criação de um nó de mídia relacionado a este descritor. O atributo region associa uma região a este descritor. Além destes atributos, podem-se definir os atributos do player, que diz qual a ferramenta de apresentação será executada para mostrar nós de mídia associadas a este descritor, e explicitDur, que diz qual será a duração temporal da apresentação dos nós de mídia relacionados a este descritor.

3.3.4 Portas

Portas servem para garantir um acesso externo ao conteúdo de um contexto.

Sendo assim, para que um elo aponte para um nó interno de um contexto, este nó deve apontar para uma porta que leve ao nó interno desejado.

Pode-se visualizar todo o body do documento como um grande contexto, assim precisa-se de uma porta de entrada que aponte para o primeiro nó de mídia ou contexto a ser apresentado quando da execução do documento. Um exemplo de definição de porta segue abaixo:

```
<port id="pInicio" component="video1" />
```

O atributo id novamente atribui a um nome único, pelo qual esta porta será referenciada. O atributo component diz a qual nó de mídia ou contexto esta porta

esta associada. Há ainda o atributo interface que indica a qual porta esta porta deve ser relacionada, no caso de component ser um nó de contexto, ou a qual âncora ela deve ser relacionada, caso component seja um nó de mídia.

3.3.5 Contextos

Os contextos servem para estruturar o documento NCL. Dessa forma eles podem ser aninhados, procurando sempre refletir a estrutura do documento e tornar a organização do programa mais intuitiva. Define-se contexto da seguinte forma:

```
<context id="ctxNome">
...
</context>
```

O atributo id, como nos demais itens, define o nome único pelo qual o contexto será referenciado. Além deste tem-se os atributos descriptor, que identifica qual descritor definirá a apresentação do contexto, e refer, que faz referência a outro contexto já definido, do qual este contexto herdará tudo menos o atributo id.

3.3.6 Nós de mídia

Um nó de mídia caracteriza o objeto de conteúdo propriamente dito (vídeo, áudio, texto, etc..). O nó de mídia deve identificar o arquivo com o conteúdo da mídia, além do descritor usado para regular a apresentação deste objeto de mídia.

```
<media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo1"/>
```

O atributo type define que tipo de mídia se trata. O atributo id da um nome único ao nó de mídia. O atributo src indica onde está localizado o arquivo fonte deste nó de mídia, e o atributo descriptor indica qual descritor será usado para a execução daquele objeto. Outro atributo que pode ser utilizado é o refer, que referencia outro nó de mídia do qual este nó usará todos os atributos, menos o id.

3.4 Subconjunto da NCL suportado no Web Composer

O subconjunto da NCL utilizado no Web Composer se restringe ao sincronismo de mídia sem interatividade e com reuso apenas de relação.

Este subconjunto permite introduzir vários objetos de mídia sincronizados no tempo, possibilitando a criação de aplicações hipermídia básicas, contudo avançadas e desafiadoras, por se tratar de um ambiente de autoria baseado na plataforma Web.

Para a definição de todos os objetos, com seus identificadores e atributos, a NCL faz uso dos elementos apresentados no item acima (3.3 Estrutura básica do NCL). Através do uso destes objetos, o Web Composer faz a composição da aplicação sincronizada no tempo. É possível portanto, definir através de seus componentes:

- Mídias utilizadas no projeto através do componente: <media id="" src="">;

- Delimitar trechos (no tempo ou no espaço) do conteúdo do seu objeto de mídia pai: <area id="" begin="">;

- Definir os espaços de exibição através dos elementos <property>, filhos dos elementos <media>: <property name="" value="">. Os atributos *name* podem ser definidos como: “left”, “top” “width”, “height”, que definem a área de apresentação em relação à tela total da área de exibição da aplicação;

- Configurar os relacionamento através do elemento: <link id="" xconnector="">. Um relacionamento em NCL é especificado referindo-se a uma relação, dada pelo valor URI do atributo *xconnector* do elemento <link> e pelos atores que exercerão os papéis da relação, dados pelos elementos <bind>. Um elemento <bind> especifica o papel da relação através do atributo <role> e a interface que exercerá o papel, dada pelos atributos *component,* que selecionam um objeto a ser relacionado, e *interface,* que seleciona uma interface desse objeto. (SOARES e BARBOSA, 2012).

Enfim, o Web Composer trabalha com um subconjunto da NCL que suporta a criação de aplicações básicas para TVD. Entretanto, vale ressaltar que é o início de um projeto inovador, o qual abre portas para um trabalho de grande expressão

colaborativa, que permite o rompimento de barreiras que dificultam o processo criativo. O item a seguir exemplifica esta afirmação.

3.5 Ferramentas de autoria existentes

Existem diversas ferramentas de autoria para a TV Digital, como: Composer, NCL Eclipse, GriNS, SMOX Pad, entre outras.

Embora sejam ferramentas muito robustas, as mesmas limitam a sua utilização a um público restrito, devido a exigência do conhecimento do código NCL.

Neste projeto estão apresentadas as ferramentas Composer e NCL Eclipse e um quadro comparativo entre as ferramentas acima citadas. Desta maneira será possível comparar as ferramentas desktop existentes e o Web Composer.

A diferença mais significativa entre as ferramentas atuais, que são aplicações desktop e o Web Composer, que foi desenvolvida na plataforma web, é a possibilidade do trabalho colaborativo entre seus usuários, proporcionando maior produtividade na criação de aplicações para a televisão digital.

3.5.1 NCL Composer

NCL Composer é uma ferramenta de autoria flexível e multiplataforma desenvolvida para auxiliar na criação de aplicações para a TV Digital Interativa (TVDI) em NCL (Nested Context Language).[6]

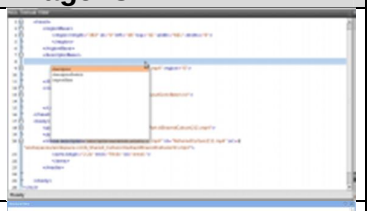
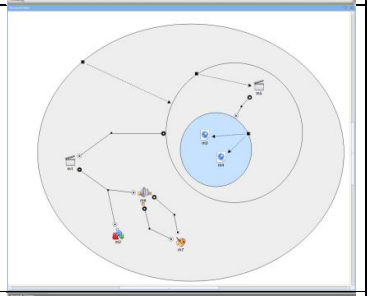
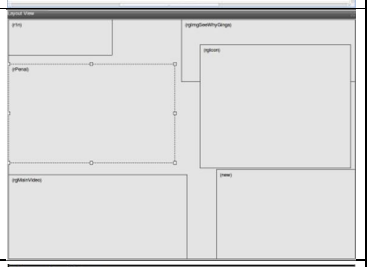
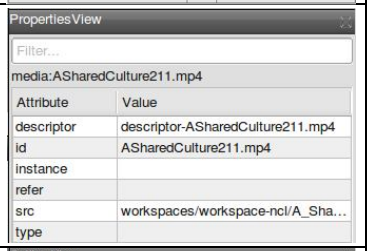
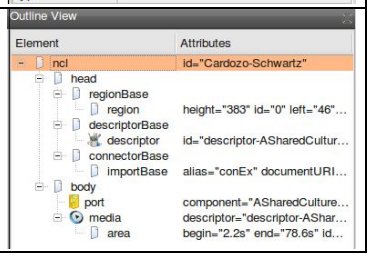
Nesta ferramenta, as abstrações são definidas em diversos tipos de visões que permitem simular um tipo específico de edição (estrutural, temporal, leiaute e textual). Essas visões funcionam de maneira sincronizada, a fim de oferecer um ambiente integrado de autoria.

Problemas de representação e edição de objetos de mídia, relacionamentos de sincronismo entre objetos, dentre eles os relacionamentos interativos, e edição ao vivo são tratados.

[6] Disponível em: Em http://composer.telemidia.puc-rio.br/?id=pt-br%2Fstart, acesso em 19/03/2013).

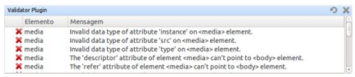
Em resumo, o sistema visa facilitar e agilizar a criação de aplicações voltadas para TV digital abstraindo do autor toda, ou pelo menos parte da complexidade de se programar em NCL através desse ambiente de autoria.[7]

Abaixo é apresentada uma tabela que demonstra todos os plug-ins especificados acima com a descrição e imagem de cada um deles:

Plugin	Tipo	Desenv.	Descrição	Imagens[8]
Visão Textual	ncl	Laboratório Telemídia	Permite a interação com o conteúdo textual.	
Visão Estrutural	ncl	Laboratório Telemídia	A Visão Estrutural permite a interação visual com a estrutura lógica do documento. Em NCL, a estrutura lógica é representada por objetos de mídia, contextos e links.	
Visão de Leiaute	ncl	Laboratório Telemídia	A Visão de Leiaute permite ao usuário interagir visualmente com regiões nas quais os objetos de mídia são apresentados.	
Visão de Propriedades	ncl	Laboratório Telemídia	A Visão de Propriedades permite que o usuário tenha uma forma fácil para acessar (e modificar) o conteúdo das propriedades dos elementos selecionados.	
Visão de Outline	ncl	Laboratório Telemídia	A Visão de Outline apresenta a estrutura do documento em forma de árvore. Esta visão é excelente para navegar entre os elementos de um documento.	

[7] Fonte:http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/composer2/onecommunity?page_um=0, acesso em 19/03/2013

[8] Fonte: http://www.ncl.org.br/en/autoria

Plugin de Validação	ncl	Laboratório Laws	O plugin Validador valida um documento NCL sempre que o modelo é modificado, dando um feedback para os usuários sobre os erros encontrados no documento que está sendo editado.[1]	

Na figura a seguir um exemplo da tela do composer, com todas as suas visões de edição sendo exibidas.

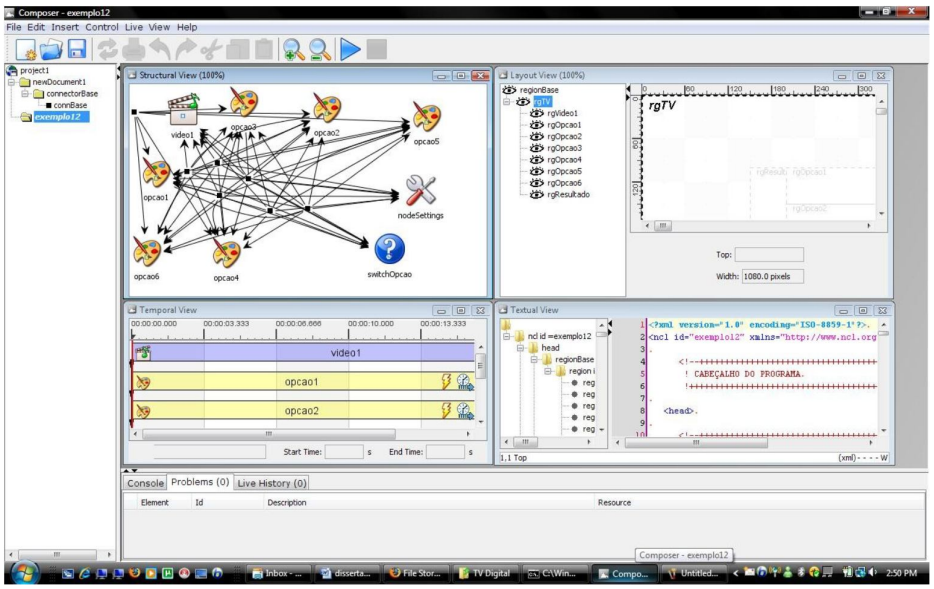


Figura 3 – Ferramenta de autoria Composer.[9]

3.5.2 NCL Eclipse

Assim como as demais ferramentas, o NCL Eclipse tem o objetivo de agilizar o desenvolvimento de aplicações para TV digital Interativa em NCL. Desenvolvido como um plug-in para o Eclipse, ele permite que todas as facilidades deste conhecido ambiente sejam reutilizadas, facilitando sua integração com outras ferramentas de desenvolvimento.

Ele nasceu com o intuito principal de tornar o trabalho de autores de apresentações interativas em NCL mais rápido e menos propenso a erros. Baseia-se

[9] Fonte: http://www.ncl.org.br/en/autoria

em princípios claros de produtividade, facilidade e integração no desenvolvimento de conteúdo interativo para TV digital.[10] Dentre as principais características implementadas pelo NCL Eclipse estão: coloração de *tags* e atributos XML, auto-formatação do código XML, validação do documento NCL, sugestão de código NCL de forma contextual (*autocomplete*), navegação no documento como uma árvore, execução do documento NCL e uma visão de Leiaute. A Figura abaixo mostra a visão geral da arquitetura do NCL Eclipse:

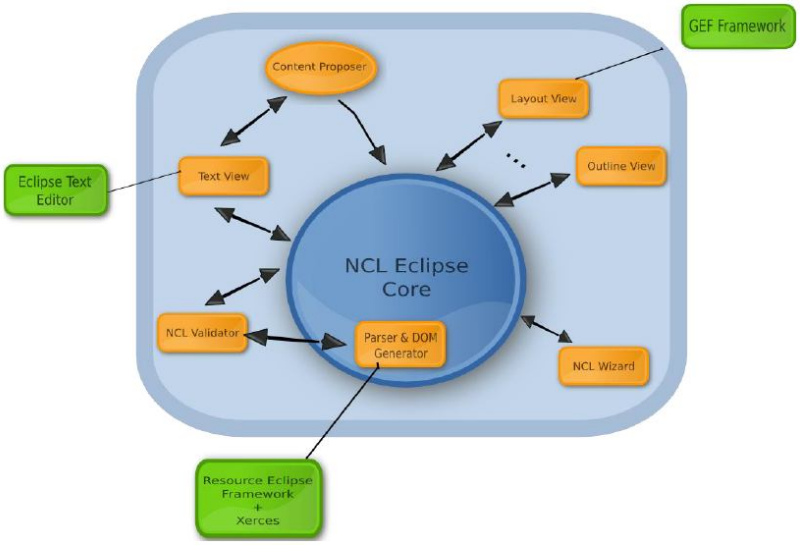


Figura 4 – Visão geral da arquitetura – Eclipse NCL.[11]

Assim como no Composer, o NCL eclipse possui módulos que auxiliam o usuário na criação de projetos. Na tabela abaixo estão descritos esses módulos e suas funcionalidades.

Módulo	**Descrição**
NCL Wizard	Um fluxo passo-a-passo que auxilia o usuário na realização de uma tarefa. É comum sua utilização para a criação de novos projetos ou documentos. Os *wizards* são organizados em categorias, sendo que tanto as categorias quanto eles próprios são definidos pelos *plug-ins* que fazem parte do ambiente.

[10] Em: http://laws.deinf.ufma.br/ncleclipse/pt-br:start#.UUk2Uhccfy0, acesso em 19/03/2013.
[11] Fonte: http://www.ncl.org.br/en/autoria

Validação Sintática e Semântica	Identifica e marca erros em documentos NCL em tempo de autoria, possibilitando assim que, caso exista erro no documento, o autor possa identificá-lo e consertá-lo, sem a necessidade de executar a aplicação.
Coloração sintática	Desenvolvida para diferenciar os elementos XML, seus atributos e os valores dos atributos, bem como alguns conteúdos textuais e comentários.
Visão Outline (*Outline View*)	Permite que se navegue pelo documento NCL como uma árvore. O evento de clicar sobre um nó da árvore leva o editor a selecionar a linha daquele nó.
Folding	Permite esconder a parte do código em que não se está interessada no momento, mostrando apenas o que é mais relevante.
Sugestão de código Contextual	Envolve o processo de o programa predizer uma palavra ou frase que o usuário deseje inserir no texto sem que seja necessário digitá-la por inteiro.
Execução de um documento NCL	Permite a execução de um documento NCL, através do comando Run do Eclipse.
Visão de Leiaute	Exemplificar a maneira como novas visões podem ser adicionadas ao ambiente de autoria, sendo possível criar regiões nas quais as mídias deverão aparecer. É possível, por exemplo, redimensionar e aninhar as regiões de forma visual.

Abaixo estão apresentadas algumas imagens do ambiente da ferramenta NCL Eclipse:

```
<?xml version="1.0" encoding="ISO-8859-1"?>          Instrução de processsamento

<!--                                                 Comentários
    Generated by NCL Eclipse
-->

<ncl id="new_ncl_file"                               Start tag
    xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

    <head>
        <connectorBase>
            <importBase alias="conn"
                documentURI="connectorBase.ncl"/>
        </connectorBase>
    </head>

    <body>
        <media id="media01" src="teste.mpg" type="video/mpeg"/>
        <media id="media02" src="teste2.mpg" type="video/mpeg"/>
        <link xconnector="conn#onBeginStart">
            <bind component="media01" role="onBegin"/>
            <bind component="media02" role="start"/>
        </link>
    </body>

</ncl>                                               End tag
```

Figura 5 – Partições de um documento NCL definidas pelo NCL Eclipse.[12]

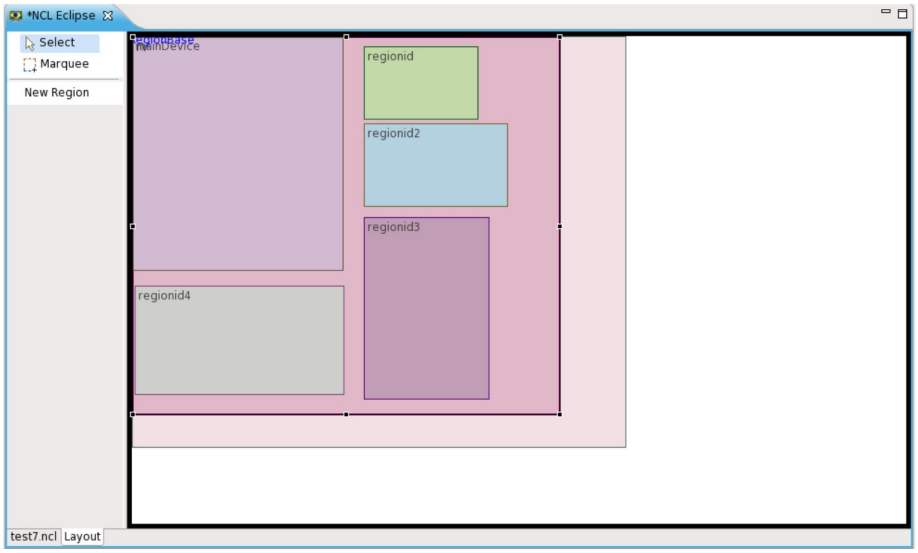


Figura 6 – Visão de Leiaute do NCL Eclipse.[13]

[12] Fonte: http://www.ncl.org.br/en/autoria
[13] Fonte: http://www.ncl.org.br/en/autoria

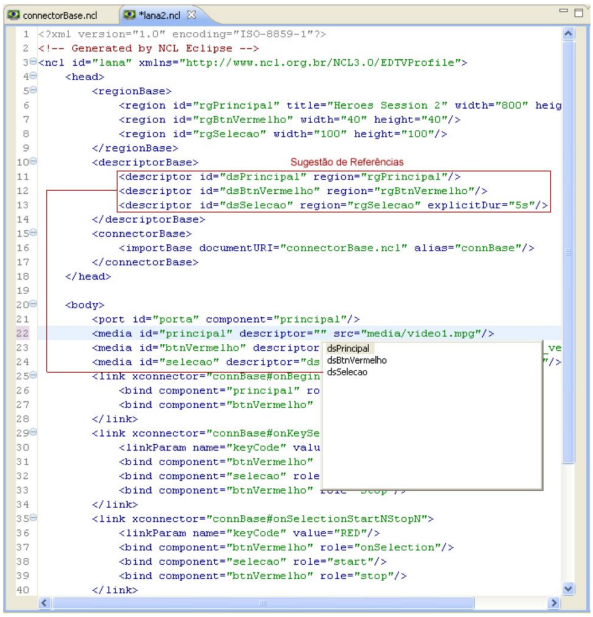


Figura 7 – Sugestão de código contextual do NCL Eclipse.[14]

3.5.3 Quadro comparativo

Abaixo está apresentado um quadro comparativo entre as ferramentas de autoria citadas, especificando as características de cada uma delas.

Todas as ferramentas, em diferentes níveis de funcionalidades, dão suporte à autoria de forma textual. Isso nos leva a concluir que por mais intuitiva que uma abstração gráfica para linguagens hipermídia possa ser, pelo menos até o momento, ela não acaba totalmente com a necessidade da autoria baseada diretamente no texto, seja pelo fato do ambiente visual não representar todas as funcionalidades da linguagem hipermídia textual ou, simplesmente, porque existem usuários que se sentem mais a vontade com a autoria textual.

Nome	Foco	Múltiplas Visões	Licença	Sugestão de código	Coloração sintática	Multi-plataforma
Composer	Visual	Sim	GPLv2	Não	Sim	Sim
NCL Eclipse	Textual	Sim	GPLv2	Sim (contextual)	Sim	Sim
GriNS	Visual	Sim	Comercial	Não	Sim (fraca)	Sim
SMOX Pad	Textual	Não	Freeware	Sim (não-contextual	Sim	Não

[14] Fonte: http://www.ncl.org.br/en/autoria

4. WEB COMPOSER NCL

O propósito deste projeto foi o desenvolvimento de um aplicativo Web que possibilitasse aos usuários criar aplicações para a TV Digital de maneira rápida, fácil e sem qualquer necessidade de conhecer e programar em NCL, além de permitir o trabalho colaborativo entre seus usuários.

O Web Composer foi desenvolvido em módulos, sendo: módulo de criação de aplicações, o módulo de execução e o módulo de exportação do código NCL referente ao projeto criado.

4.1 Requisitos para o desenvolvimento do Web Composer

Devido as deficiências identificadas nas ferramentas de autorias atuais, para o alcance das metas deste projeto, foi necessário elencar e agrupar requisitos que permitissem o desenvolvimento do aplicativo, quais sejam:

- Armazenar informações em banco de dados para possibilitar o reaproveitamento de projetos criados e mídias carregadas;
- Atender inicialmente um subconjunto da NCL (item 3.4) para a interação simples entre as mídias;
- Permitir a carga e reutilização de mídias de áudio, vídeo e imagem;
- Permitir o trabalho colaborativo, ou seja, múltiplo acesso e compartilhamento dos projetos criados entre os usuários;
- Deve possuir um interface simples e amigável, sem expressões técnicas ou em idioma diferente do português;
- Possibilitar a criação, execução e exportação do código NCL.

Para tanto, foram utilizados as seguintes linguagens de programação, script, marcação, SQL e outras tecnologias:

- PHP (Utilizado para compor a estrutura da aplicação e realizar a comunicação com o banco de dados);
- Javascript (Linguagem de script utilizada para validação de formulários e para o desenvolvimento do módulo de execução);
- MySql (Empregado na armazenagem de informações pertinentes ao projeto no banco de dados);
- AJAX (Ferramenta utilizada para busca assíncrona de informações no banco de dados);

- CSS (Utilizado para a formatação de toda a aplicação, em especial o modelo de execução);
- HTML. (Linguagem utilizada para compor a estrutura do site da aplicação).

4.2 Desenvolvimento do projeto

Para realizar a implementação do projeto foram estabelecidos alguns parâmetros para que o objetivo do mesmo fosse alcançado. Foi necessário o estudo de ferramentas que possibilitassem o domínio sobre o controle da execução do vídeo, a sobreposição de imagens, vídeo e áudio, sobre o vídeo principal. Além disso foi necessário a implementação de uma interface de inserção de dados em um banco de dados (seção 4.2.4) que permitisse ao usuário realizar o upload de arquivos e realizar a integração dos mesmos (através de links), conforme está demonstrado na seção 4.4.

Assim, foi possível que em tempo de execução todos os dados fossem carregados e a animação fosse executada conforme configurada pelo usuário.

A seguir estão apresentados todos os módulos e recursos desenvolvidos e utilizados no presente trabalho.

4.2.1 Banco de dados – Modelagem

Através dos dados armazenados no banco de dados é possível realizar a execução do projeto desenhado pelos usuários, assim como gerar o código NCL do mesmo.

Como é possível observar na modelagem do banco de dados abaixo, um usuário pode criar quantos projetos desejar e esses projetos podem possuir a quantidade de mídias que for necessária, sendo que essas mídias possuem suas propriedades e podem possuir inúmeros links com outras mídias.

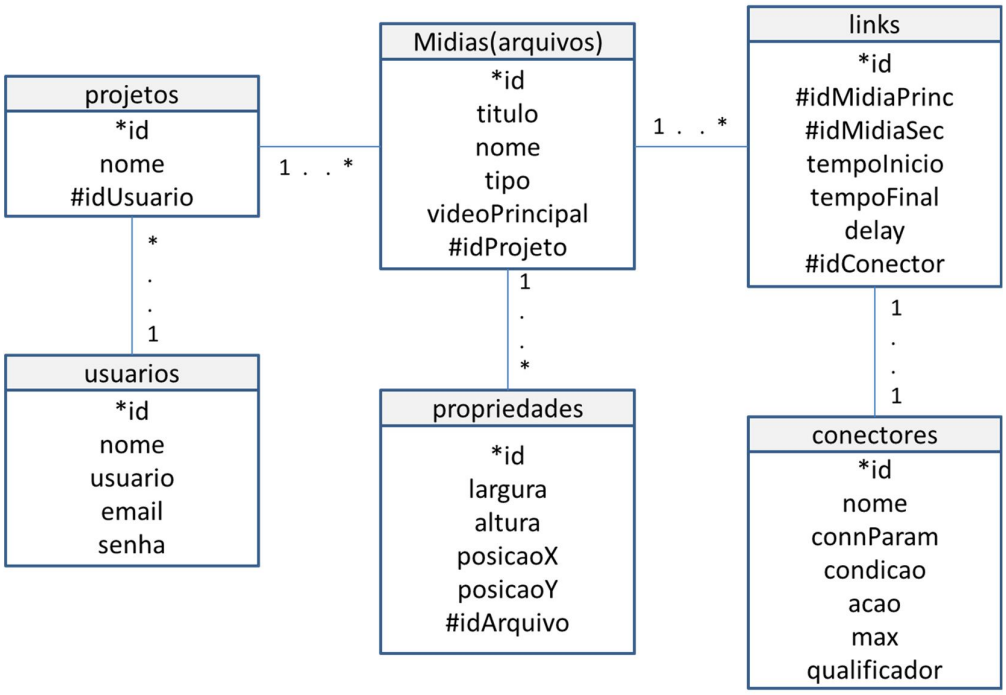


Figura 8 – Modelagem do banco de dados

A descrição de cada um dos campos das tabelas acima estão explicitadas na tabela abaixo:

Tabela	Campos	Tipo	Descrição
projetos	id	INT	Chave primária.
	nome	VARCHAR	Nome do projeto.
	idUsuario	INT	Especifica qual é o usuário dono do projeto.
usuarios	id	INT	Chave primária.
	nome	VARCHAR	Nome do usuário.
	usuario	VARCHAR	Login do usuário.
	email	VARCHAR	E-mail do usuário.
	senha	VARCHAR	Senha do usuário.
midias	id	INT	Chave primária
	titulo	VARCHAR	Nome dado à mídia.
	nome	VARCHAR	Nome MIME e extensão do arquivo.
	tipo	VARCHAR	Extensão do arquivo.
	videoPrincipal	VARCHAR	Campo booleano que define se a mídia é o arquivo principal do projeto.
	idProjeto	INT	Especifica qual projeto a mídia pertence.
propriedades	id	INT	Chave primária
	largura	INT	Largura do arquivo.

	altura	INT	Altura do arquivo.
	posicaoX	INT	Posição horizontal em que o arquivo irá aparecer.
	posicaoY	INT	Posição vertical em que o arquivo irá aparecer.
	idArquivo	INT	Especifica qual é o arquivo que está recebendo as propriedades.
links	id	INT	Chave primária
	idMidiaPrinc	INT	Identifica qual é a mídia principal.
	idMidiaSec	INT	Identifica a mídia que terá o elo com a mídia principal.
	tempoInicio	INT	Especifica em que momento a mídia aparecerá.
	tempoFim	INT	Especifica em que momento a mídia irá desaparecer.
	delay	INT	Especifica o tempo de atraso a partir do início da mídia principal.
	idConector	INT	Especifica qual é o tipo de conector do link.
conectores	id	INT	Chave primária
	nome	VARCHAR	Nome do conector.
	connParam	VARCHAR	Especifica o parâmetro do conector.
	condicao	VARCHAR	Especifica o papel de condição do conector.
	acao	VARCHAR	Especifica o papel de ação do conector.
	max	VARCHAR	Especifica o número de papéis que o papel pode exercer.
	qualificador	VARCHAR	Define a forma como as ações devem ser executadas.

4.2.2 Módulo de criação de aplicações

Este módulo permite ao usuário do Web Composer a utilização da ferramenta desde o seu cadastro até a execução e exportação do projeto criado.

O módulo de criação engloba as seguintes funcionalidades:

- Cadastro do usuário na aplicação: o usuário deve se cadastrar na aplicação informando o nome de usuário pretendido, e-mail e senha.

Arquivo: cadastroUsuario.php

```
<?php
//Recebe dados do formulário de cadastro.
$nome = $_POST['nome'];
```

```
$usuario = $_POST['usuario'];
$email = $_POST['email'];
$senha = $_POST['senha'];
        //Variáveis para reporte de erros durante o cadastro
$erro01 = "Erro 409.  Registro já existe na base de dados.";
$erro02 = "Erro 002.  Preencha todos os campos obrigatórios.";
$erro03 = "Erro 003.  Endereço de email inválido.";
        //Verifica se algum campo está em branco
$email = "$email";
if (!($nome) || !($email) || !($senha)|| !($usuario)){
   echo('<div id="erro">');
   echo"$erro02";
   echo'<br /><br /><input type="button" value="Voltar" onClick="history.go(-1)"><br
   />';
   exit();
}
        //Abre Conexao com o banco de dados
$conexao = mysql_pconnect("localhost","root","") or die (mysql_error());
$banco = mysql_select_db("repositorio");
        //Utilizando o  mysql_real_escape_string voce se protege o seu código contra
SQL Injection.
$nome = mysql_real_escape_string($nome);
$usuario = mysql_real_escape_string($usuario);
$email = mysql_real_escape_string($email);
$senha = mysql_real_escape_string($senha);
$email = $_REQUEST["email"];

        //Verifica se já existe o e-mail cadastrado no banco de dados
if (mysql_num_rows(mysql_query("SELECT email FROM usuarios WHERE email =
'$email'")) != 0) {
   echo('<div id="erro">');
   echo "$erro01";
   echo("<script>document.f.nome.focus()</script>");
```

```
  echo'<br /><input type="button" value="Voltar" onClick="history.go(-1)"><br />';
  exit();
}
        //Faz a inserção no usuário no banco de dados.
else {
  $insert = mysql_query("insert into usuarios (nome,email,usuario,senha) values
  ('{$nome}','{$email}','{$usuario}','{$senha}')");
  mysql_close($conexao);
}
        //Exibe a mensagem de gravação em banco de dados se não tiver ocorrido
erro.
if($insert) {
  echo('<div id="erro">');
  print "<img src=\"load.gif\"/> Carregando...</div>";
  echo "Dados gravados com sucesso!";
  echo ("<script>alert('Dados gravados com sucesso! Faça seu login!');</script>");
  echo("<script>window.location=' index.php?id=login';</script>");
  echo("<script>document.f.nome.focus()</script>");
        //Exibe mensagem de erro se tiver ocorrido algum problema
}else {
  print "Erro ao Cadastrar!";
  echo'<br /><input type="button" value="Voltar" onClick="history.go(-1)">';
}
        ?>
```

- “Log in” do usuário na aplicação: o usuário realiza o “log in” no sistema utilizando os dados cadastrados.

```
        Arquivo: login.php (formulário):

        <form name="login" method="post" action="valida.php">
        <table cellspacing='2' cellpadding='2' border='1' bordercolor='ddd'
width="500px">
```

```
                <tr style='background-color: #ddd'>
                    <td colspan="3">Login</td>
                </tr>
                <tr>
                    <td>Usuário:</td>
                    <td><input type="text" name="usuario" maxlength="50" /></td>
                </tr>
                <tr>
                    <td>Senha:</td>
                    <td><input type="password" name="senha" maxlength="50" /></td>
                </tr>
                <tr>
                    <td><input type="button" value="Entrar" onclick="validar();"/></td>
                    <td></td></tr>
            </table>
        </form>
        Arquivo: valida.php.

        <?php
        // Inclui o arquivo com o sistema de segurança
        include("seguranca.php");

        // Verifica se um formulário foi enviado
        if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        // Salva duas variáveis com o que foi digitado no formulário
        // Detalhe: faz uma verificação com isset() pra saber se o campo foi
preenchido
        $usuario = (isset($_POST['usuario'])) ? $_POST['usuario'] : '';
        $senha = (isset($_POST['senha'])) ? $_POST['senha'] : '';
        // Utiliza uma função criada no seguranca.php pra validar os dados digitados
        if (validaUsuario($usuario, $senha) == true) {
        // O usuário e a senha digitados foram validados, manda pra página interna
        header("Location: index.php?id=inicio");
```

```
        } else {
        // O usuário e/ou a senha são inválidos, manda de volta pro form de login
        // Para alterar o endereço da página de login, verifique o arquivo
seguranca.php
        expulsaVisitante();
        }
        }
        ?>
```

- Criar novo projeto: após realizar o “log in” no sistema o usuário poderá criar novos projetos.

```
        Arquivo: novoProjeto.php (formulário)

        <body>
            Criar novo projeto.<br>
            Digite o nome que quer dar ao projeto e clique em Iniciar.<br /><br />
            <form name="proj" action="index.php?id=novoProjetoUpload"
method="POST">
            <table border="1" width="1" cellspacing="1" cellpadding="1" style="width:
400px">
                <tr style="background-color: #DDDDDD">
                    <td colspan="2" align="center">Novo projeto:</td>
                </tr>
                <tr>
                    <td>Nome:</td>
                    <td><input type="text" name="nomeProjeto" value="" size="40"
/></td>
                </tr>
                <tr>
                    <td colspan="2" align="center"><input type="button" value="Iniciar"
onclick="validar()"/></td>
                </tr>
```

```
            </table>
            </form>
            <br />
        </body>
```

Arquivo insereProjeto.php

```
        <?php
        include("conexao.php");
        include("seguranca.php"); // Inclui o arquivo com o sistema de segurança
        protegePagina(); // Chama a função que protege a página
        $idUsuario = $_SESSION['usuarioID'];
        $nome = $_POST['nomeProjeto'];

        // Query de consulta SQL
        $query = "INSERT INTO projetos (nome, idUsuario) VALUES ('".$nome."',
'".$idUsuario."')";

        // Executa a query
        $inserir = mysql_query($query);

        // Verifica se ocorreu algum erro na inserção do projeto.
        if ($inserir) {
        echo "Projeto criado com sucesso.<br /> Faça agora o Upload dos arquivos
que deseja trabalhar!";
        } else {
        echo "Não foi possível inserir a notícia, tente novamente.";
        // Se houver,exibe dados sobre o erro:
        echo "<br />Dados sobre o erro:" . mysql_error();}
        ?>
```

- Fazer carga (upload) de arquivos: após a criação do projeto é permitido ao usuário a inserção de mídias a serem utilizadas no mesmo.

Arquivo: uploadMidias.php (formulário).

```
<form method="post" action="index.php?id=novoUpload"
enctype="multipart/form-data" name="formUpload">
    <table border="1" width="1" cellspacing="1" cellpadding="1" style="width:
500px">
        <tr style="background-color: #DDDDDD">
         <td colspan="2" align="center">Upload de arquivos:</td>
        </tr>
        <tr>
            <td>Arquivo:</td>
            <td><input type="file" name="arquivo" /></td>
        </tr>
        <tr>
            <td>Nome do arquivo</td>
            <td><input type="text" name="titulo" /></td>
        </tr>
        <tr>
         <td colspan="2" align="center"><input type="submit"  value="Enviar"
/></td>
        </tr>

    </table>
     <input type="hidden" name="idProjeto" value="<?php echo
$idProjeto;?>">
</form>
```

Arquivo: upload-arquivo.php

```
<?php
include("conexao.php");
```

```
        include("seguranca.php"); // Inclui o arquivo com o sistema de segurança
        protegePagina(); // Chama a função que protege a página
        $idProjeto = $_POST['idProjeto'];
        $_UP['pasta'] = 'img/'; // Pasta onde o arquivo vai ser salvo
        // Tamanho máximo do arquivo (em Bytes)
        $_UP['tamanho'] = 1024 * 1024 * 1024 * 1024 * 10; // 10MB
// Array com as extensões permitidas
        $_UP['extensoes'] = array('wmv', 'mp3', 'mp4', 'jpg', 'jpeg', 'png', 'gif');
        // Renomeia o arquivo? (Se true, o arquivo será salvo como .jpg e um nome
único)
        $_UP['renomeia'] = false;
        // Array com os tipos de erros de upload do PHP
        $_UP['erros'][0] = 'Não houve erro';
        $_UP['erros'][1] = 'O arquivo no upload é maior do que o limite do PHP';
        $_UP['erros'][2] = 'O arquivo ultrapassa o limite de tamanho especifiado no
HTML';
        $_UP['erros'][3] = 'O upload do arquivo foi feito parcialmente';
        $_UP['erros'][4] = 'Não foi feito o upload do arquivo';

        // Verifica se houve algum erro com o upload. Se sim, exibe a mensagem do
erro
        if ($_FILES['arquivo']['error'] != 0) {
        die("Não foi possível fazer o upload, erro:<br />" .
$_UP['erros'][$_FILES['arquivo']['error']]);
        exit; // Para a execução do script
        }

        // Caso script chegue a esse ponto, não houve erro com o upload e o PHP
pode continuar
        // Faz a verificação da extensão do arquivo
        $ref = explode('.', $_FILES['arquivo']['name']);
        $extensao = strtolower(end($ref));
        if (array_search($extensao, $_UP['extensoes']) === false) {
```

```
        echo "Por favor, envie arquivos com as seguintes extensões: jpg, png ou gif";
        }

        // Faz a verificação do tamanho do arquivo
        else if ($_UP['tamanho'] < $_FILES['arquivo']['size']) {
        echo "O arquivo enviado é muito grande, envie arquivos de até 10Mb.";
        }

        // O arquivo passou em todas as verificações, hora de tentar movê-lo para a
pasta
        else {
        // Primeiro verifica se deve trocar o nome do arquivo
        if ($_UP['renomeia'] == true) {
        // Cria um nome baseado no UNIX TIMESTAMP atual e com extensão .jpg
        $nome_final = time().'.jpg';
        } else {
        // Mantém o nome original do arquivo
        $nome_final = $_FILES['arquivo']['name'];
        }

        // Depois verifica se é possível mover o arquivo para a pasta escolhida
        if (move_uploaded_file($_FILES['arquivo']['tmp_name'], $_UP['pasta'] .
$nome_final)) {
        // Upload efetuado com sucesso, exibe uma mensagem e um link para o
arquivo
        $caminho = $_UP['pasta'] . $nome_final;
        $titulo = $_POST['titulo'];
        //Armazena no banco de dados
        $con=mysql_connect ($srv, $usr, $snh) or print mysql_error();
        $db=mysql_select_db ($banco, $con) or print mysql_error();
        $sql="insert into arquivos (titulo,nome,caminho,tipo,idProjeto) values
('$titulo','$nome_final','$caminho','$extensao','$idProjeto')";
        $exec=mysql_query ($sql, $con) or print mysql_error();
```

```
        mysql_close ($con);
        echo "Upload efetuado com sucesso!<br />";
        } else {
        // Não foi possível fazer o upload, provavelmente a pasta está incorreta
        echo "Não foi possível enviar o arquivo, tente novamente";
        }
        }
        echo '<br />';
        echo '<input type="button" value="Voltar" onClick="history.go(-1)">';
        echo '<br /><br />';
        echo "Ir para a página de projeto: <a href='index.php?id=inicio' style='color:
#000'>Projetos</a>";
        ?>
```

- Definir as propriedades dos arquivos: após a inserção das mídias o usuário deve configurar as propriedades das mesmas, sendo: altura, largura, posição vertical e horizontal, onde as mesmas deverão aparecer durante a aplicação interativa.

```
        Arquivo: configPropriedades.php (formulário).

        <form name='propArquivo' action='index.php?id=atualizaProp' method='POST'
>
            <input type='hidden' name='idProjeto' value='".$idProjeto."'>
            <tr>
                <td>Id:</td>
                <td><input type='text' name='idArquivo' value='".$escrever['id']."' size='2'
readonly='readonly' /></td>
                <td>Nome:</td>
                <td><input type='text' name='nomeArquivo' value='".$escrever['titulo']."'
size='20' readonly='readonly'/></td>
            </tr>
            <tr>
                <td>Largura:</td>
```

```
                <td><input type='text' name='largura' value='".$largura."' size='2'
/>%</td>
                <td>Altura:</td>
                <td><input type='text' name='altura' value='".$altura."' size='2' />%</td>
            </tr>
            <tr>
                <td>Coord X:</td>
                <td><input type='text' name='coordX' value='".$coordX."' size='2'
/>%</td>
                <td>Coord Y:</td>
                <td><input type='text' name='coordY' value='".$coordY."' size='2'
/>%</td>
            </tr>
            <tr style='background-color: #DDDDDD'>
                <td colspan='4' align='center'>
                    <input type='submit' value='Salvar' name='Salvar' />
                </td>
            </tr>
            </form>
```

- Configurar os links entre as mídias: o usuário deve configurar o momento em que as interações entre as mídias irá acontecer (tempo inicial e final), assim como deve escolher qual o tipo de interação a ser realizada (no início ou no fim de alguma ação).

```
        Arquivo: formLinks.php (formulário).

        <form name='insereLinks' action='index.php?id=insereLinks' method='POST'>
        <table cellspacing='2' cellpadding='2' border='1' bordercolor='DDDDDD'
width='800px'>
            <tr style='background-color: #DDDDDD'>
                <td colspan='4' align='center'>Inserir link</td>
            </tr>
```

```
            <tr>
                <input type='hidden' name='idProjeto' value="<?php echo $idProjeto;?>"
>
                <input type='hidden' name='idArquivo' value="<?php echo $idArquivo;?>"
>
                <td>Arquivo principal:</td>
                <td><?php echo $tituloArqPrincipal['titulo'];?></td>
                <td>Arquivo secundário:</td>
                <td>
                <select id='comboArquivos' name='comboArquivos' onchange=''>
                        <option >Selecione o arquivo...</option>
                        <?php
                        while ($escrever = mysql_fetch_assoc($arquivos)) {
                            echo "<option value={$escrever['A']}>{$escrever['B']}</option>";
                        }
                        ?>
                    </select>
                </td>
            </tr>
            <tr>
                <td>Conector:
                </td>
                <td><select id='comboConectores' name='comboConectores'
onchange=''>
                        <option >Selecione o conector...</option>
                        <?php
                        while ($listaConectores = mysql_fetch_assoc($conectores)) {
                            echo "<option
value={$listaConectores['id']}>{$listaConectores['nome']}</option>";
                        }
                        ?>
                    </select></td>
                <td>Delay:</td>
```

```
            <td><input type="text" name="delay" value="" size="2"/>s</td>
        </tr>
        <tr>
            <td>Tempo inicial:</td>
            <td><input type="text" name="tinicial" value="" size="2"/>s</td>
            <td>Tempo final:</td>
            <td><input type="text" name="tfinal" value="" size="2"/>s</td>
        </tr>
            <tr>
                <td colspan="4" align ="center"><input type="submit" value="Salvar"
name="inserir" /></td>
            </tr>
    </table>
    </form>
```

Portanto, neste módulo é permitido ao usuário fazer o upload das mídias que deseja inserir em seu projeto, configurar o tamanho e a localização onde as mesmas irão aparecer durante a execução da aplicação e definir o momento no qual as mídias farão a interação umas com as outras.

Conforme será apresentado nas próximas seções, após a configuração do projeto, o usuário poderá testar a sua execução e exportar o código NCL da mesma.

4.2.3 Módulo de execução de aplicações

Pode-se considerar este módulo como o alicerce de todo o projeto. Nele foram desenvolvidas soluções que permitem a sobreposição de mídias, incluindo a execução de um vídeo sobre outro, de uma imagem sobre um vídeo e a execução um áudio em paralelo a um vídeo.

Estas soluções, até então não abordadas em outros aplicativos Web, abrem um grande horizonte para o teste de aplicações para a TV Digital. Por ser o primeiro aplicativo Web a oferecer tais possibilidades, a partir deste protótipo poderão ser desenvolvidas outras soluções que permitirão aos usuários a criação de aplicações que contemplem toda a gama de opções oferecidas pela NCL.

Pioneiro na execução/teste de aplicativos para a TVD em ambiente Web, este módulo permite simular perfeitamente a execução dos aplicativos na TVDI, através do middleware Ginga.

Como demonstrado nós códigos abaixo, a execução da aplicação é realizada no lado do cliente. Todos os dados do projeto são trazidos do banco de dados e carregados no browser do usuário, o que permite o sincronismo perfeito das mídias em tempo de execução.

Este módulo é composto de:

- Busca das informações do banco de dados e conversão dos dados em javascript (para a execução da aplicação no lado do cliente).

```
        //Busca o arquivo principal do projeto para carregar o vídeo.
        $query = mysql_query("SELECT caminho FROM arquivos WHERE
videoPrincipal = 1 and idProjeto = $idProjeto");
        $caminhoVideo = mysql_fetch_array($query);
        //Busca os demais dados do projeto
        $queryDados = mysql_query("SELECT arquivos.id, arquivos.caminho,
arquivos.tipo, propriedades.largura,
        propriedades.altura, propriedades.posicaoX, propriedades.posicaoY,
links.tempoInicio, links.tempoFinal, links.delay FROM arquivos JOIN propriedades
ON (arquivos.id = propriedades.idarquivo AND arquivos.idprojeto = $idProjeto) JOIN
links ON (arquivos.id = links.idArqSecundario) JOIN conectores ON (conectores.id =
links.idconector)");

        //Verifica se o usuário definiu qual será o vídeo principal do projeto
        if (empty($caminhoVideo)) {
            echo "<script>alert('Atenção: Primeiro você deve configurar as
propriedades do projeto!');
                history.go(-1);
                </script>";
        }
```

```
        //Percorre o array do banco de dados e armazena os tipos de dados em
arrays específicos.
            while ($escrever = mysql_fetch_array($queryDados)) {
                $idArq[] =         $escrever['id'];
                $caminho[] =    $escrever['caminho'];
                $tipo[] =       $escrever['tipo'];
                $largura[] =    $escrever['largura'];
                $altura[] =     $escrever['altura'];
                $coordX[] =     $escrever['posicaoX'];
                $coordY[] =     $escrever['posicaoY'];
                $tInicio[] =    $escrever['tempoInicio'];
                $tFinal[] =     $escrever['tempoFinal'];
                $delay[] =      $escrever['delay'];
            }

        // Prepara os arrays PHP para converter em arrays Javascript.
                $arrayId =          implode("|", $idArq);
                $arrayCaminho =     implode("|", $caminho);
                $arrayTipo =        implode("|", $tipo);
                $arrayLargura =     implode("|", $largura);
                $arrayAltura =      implode("|", $altura);
                $arrayCoordX=       implode("|", $coordX);
                $arrayCoordY=       implode("|", $coordY);
                $arrayInicio =      implode("|", $tInicio);
                $arrayFinal =       implode("|", $tFinal);
                $arrayDelay =       implode("|", $delay);
        ?>
        //recebe a string com elementos separados, vindos do PHP
        array_id =          "<?php echo $arrayId;  ?>";
        array_caminho =     "<?php echo $arrayCaminho;  ?>";
        array_tipo =        "<?php echo $arrayTipo;     ?>";
        array_largura =     "<?php echo $arrayLargura;  ?>";
        array_altura =      "<?php echo $arrayAltura;  ?>";
```

```
array_coordX =      "<?php echo $arrayCoordX;   ?>";
array_coordY =      "<?php echo $arrayCoordY;   ?>";
array_tInicio =     "<?php echo $arrayInicio;   ?>";
array_tFinal =      "<?php echo $arrayFinal;    ?>";
array_delay =       "<?php echo $arrayDelay;    ?>";

//transforma esta string em um array próprio do Javascript
idArq =         array_id.split("|");
caminho =   array_caminho.split("|");
tipo =      array_tipo.split("|");
largura =   array_largura.split("|");
altura =    array_altura.split("|");
coordX =    array_coordX.split("|");
coordY =    array_coordY.split("|");
tInicio =   array_tInicio.split("|");
tFinal =    array_tFinal.split("|");
delay =     array_delay.split("|");

//variáveis que definem a largura e altura do módulo de execução.
larguraVideo = 480;
alturaVideo = 360;
```

- Configuração do ambiente de execução: a sobreposição das mídias é realizada através da utilização de CSS. A solução encontrada foi a sobreposição de divs ( ), sendo as mesmas configuradas de acordo com o arquivo a ser sobreposto ao vídeo.

Abaixo estão as funções em javascript que são chamadas em tempo de execução para a exibição ou exclusão das mídias. Nelas são configuradas as classes CSS, que são: (divAudio, divImagem e divVideo).

As classes em CSS foram definidas conforme código abaixo. Pode-se observar que as mídias são posicionadas de forma absoluta, enquanto a classe que define a região de execução da aplicação (.palco) foi definida de maneira relativa, artifício que possibilita a sobreposição das mídias utilizando CSS.

```
        function
criarDivImagemDados(idNome,caminho,altura,largura,posicaoX,posicaoY){
                newdivImagem = document.createElement('div');
                newdivImagem.setAttribute('id',idNome);
                newdivImagem.setAttribute('class','divImagem');
                newdivImagem.innerHTML = '<input type="image" height="'+altura+'"
width="'+largura+'" src="'+caminho+'" style="top: '+posicaoY+' ;left:'+posicaoX+' ;">';
                document.getElementById('palco').appendChild(newdivImagem);
                //iCount++;
        }

        function escondeDivImagemDados(idNome){
                var pai = document.getElementById('palco');
                var filho = document.getElementById(idNome);
                pai.removeChild(filho);
        }

        function criarDivVideoDados(idNome, caminho, largura, altura){
                newdivVideo = document.createElement('div');
                newdivVideo.setAttribute('id',idNome);
                newdivVideo.innerHTML = '<video height="'+altura+'" width="'+largura+'"
autoplay="autoplay" class="divVideo"><source src="'+caminho+'"
type="video/mp4"></video>';
                document.getElementById('palco').appendChild(newdivVideo);
                iCount++;
        }

        function escondeDivVideo(idNome){
                var pai = document.getElementById('palco');
                var filho = document.getElementById(idNome);
                pai.removeChild(filho);
        }
```

```
        function criarDivAudio(idNome,caminho){
                newdivAudio = document.createElement('div');
                newdivAudio.innerHTML = '<audio controls="" autoplay="autoplay"
class="divAudio" id="'+idNome+'"><source src="'+caminho+'"></audio>';
                document.getElementById('palco').appendChild(newdivAudio);
                //iCount++;
        }

        function criarDivAudio(){
                newdivAudio = document.createElement('div');
                newdivAudio.innerHTML = '<audio controls="" autoplay="autoplay"
class="divAudio" id="audioTeste"><source src="mediasVideo/choro.mp4"></audio>';
                document.getElementById('palco').appendChild(newdivAudio);
                //iCount++;
        }

        function escondeDivAudio(idNome){
                var pai = document.getElementById('palco');
                var filho = document.getElementById(idNome);
                pai.removeChild(filho);
        }
```

```
        .divImagem, .divAudio, .divVideo {
position: absolute;
z-index: 2;
        }

        .palco {
                position: relative;
        }
```

-Executar a aplicação: a execução da aplicação inicia quando o usuário clica sobre a região onde está posicionado o vídeo principal. Ao acionar a aplicação, a função javascript “setInterval()” chama recursivamente a função testaTempo(), que verifica a cada segundo se existe alguma mídia a ser sobreposta ao vídeo.

```
        <body onload="clickPage();">

        <div id="tudo">
                <div id="conteudo">
                        <div class="palco" id="palco">
                           <video id="videoTeste" class="video-js vjs-default-skin"
preload="auto" autoplay="autoplay" width="480" height="360" poster="" data-
setup="{}">
                              <source src="<?php echo
$caminhoVideo['caminho'];?>" type='video/mp4' />
                           <!-- <track kind="captions" src="captions.vtt" srclang="en"
label="English" /> -->
                           </video>
                        </div>
                </div>
        </div>

        <script>
        var palco = document.getElementById('palco');
        palco.onclick = clickPage;

        function clickPage() {
              setInterval( function(){showTempoAtual()
                 testaTempo(tempoCerto);
              }, 1000);
        }

        var myPlayer = _V_("videoTeste");
```

```
        var myAudio = document.getElementById("audioTeste");

        function showTempoAtual() {
        document.getElementById('tempoAtual').innerHTML = getTempoAtual();
        }

        function getTempoAtual(){
                var tempoAtual = myPlayer.currentTime();
                tempoCerto = tempoAtual.toFixed(0);
               return tempoCerto;
        }

        function testaTempo(a){
            //varre o array só pra mostrar que tá tudo ok
            var segundoAtual = a;
            for (i in tInicio){
                if (segundoAtual > 0){
                if ( (tInicio[i] == segundoAtual) || (delay[i] == segundoAtual) ){
                    if((tipo[i] == "png") || (tipo[i] == "jpg") || (tipo[i] == "gif")){
criarDivImagemDados(idArq[i],caminho[i],((alturaVideo*altura[i])/100),((larguraVideo*l
argura[i])/100),coordX[i],coordY[i]);
                    }
                    if( tipo[i] == "mp4"){
                        criarDivVideoDados( idArq[i],caminho[i],
((larguraVideo*largura[i])/100),((alturaVideo*altura[i])/100) );
                    }
                    if( tipo[i] == "mp3"){
                        criarDivAudio(idArq[i],caminho[i]);
                    }
                }

                if ( (tFinal[i] == segundoAtual) || (delay[i] == segundoAtual) ){
                    if( (tipo[i] == "png") || (tipo[i] == "jpg") || (tipo[i] == "gif") ){
```

```
                escondeDivImagemDados(idArq[i]);
            }
            if(tipo[i] == "mp4"){
                escondeDivVideo(idArq[i]);
            }
            if(tipo[i] == "mp3"){
                escondeDivAudio(idArq[i]);
            }
        }
        }
    }
}
</script>
</body>
```

Como é possível observar, este módulo possui um conjunto de ferramentas que foram integradas para alcançar o objetivo de simular e testar a execução de uma aplicação da TVDI em ambiente WEB.

Pode-se considerar que este módulo é a base fundamental para futuros projetos que venham a trabalhar com a TVDI em ambiente Web.

4.2.4 Módulo de exportação do código NCL

Este módulo permite que os usuários exportem o código NCL proveniente da criação e configuração do projeto de aplicação para a TV Digital.

Tão importante quanto ao módulo de execução e teste da aplicação, este módulo proporciona algo inovador para a autoria de aplicações para a TVD em ambiente web, pois possibilita que usuário obtenha o código NCL sem a necessidade de possuir qualquer conhecimento a respeito do mesmo.

Este módulo busca todas as informações do banco de dados referente ao projeto criado e gera o código NCL que descreve toda a aplicação.

Abaixo o código executado neste módulo:

```
<?php
include("seguranca.php"); // Inclui o arquivo com o sistema de segurança
protegePagina(); // Chama a função que protege a página
$idProjeto = $_GET['idProjeto'];

//Busca nome do projeto.
$queryNome = mysql_query("SELECT nome FROM projetos WHERE id =
$idProjeto");
$nomeProjeto = mysql_fetch_array($queryNome);

//Busca conectores
$queryConectores = mysql_query("SELECT * FROM conectores");

//Busca o arquivo principal do projeto para carregar o vídeo.
$queryMidiaPrinc = mysql_query("SELECT nome,id,caminho FROM arquivos
WHERE videoPrincipal = 1 and idProjeto = $idProjeto");
$mediaPrinc = mysql_fetch_array($queryMidiaPrinc);

//Verifica se o arquivo principal vou definido pelo usuário
if (empty($mediaPrinc)) {
   echo "<script>alert('Atenção: Primeiro você deve configurar as
propriedades do projeto!');
      history.go(-1);
      </script>";
}

//Busca proriedades das mídias
$queryPropriedades = mysql_query("SELECT * FROM propriedades JOIN
arquivos ON ( propriedades.idArquivo = arquivos.id
AND arquivos.idProjeto = $idProjeto )" );

//Busca lista de links
```

```
        $queryLinks = mysql_query("SELECT links.id as linkId, links.tempoInicio,
links.tempoFinal, links.delay, arquivos.nome AS arquivo, conectores.nome,
conectores.acao, conectores.condicao AS conector, arquivos.id
        FROM links JOIN arquivos ON ( links.idArqSecundario = arquivos.id AND
arquivos.idProjeto = $idProjeto) JOIN conectores ON (links.idConector =
conectores.id)");

        //variáveis para composição do NCL - dados estáticos
        $cabecalho = '< ? xml version="1.0" encoding="ISO-8859-1" ? ><br />';
        $cabecalho .='< ncl id="'.$nomeProjeto['nome'].'"
xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile" >';
        $cabecalho .='<br /><br />< head ><br />< connectorBase >';
        while ($escrever = mysql_fetch_array($queryConectores)){
            $cabecalho .='<br /><  causalConnector id="'.$escrever['nome'].'">';
            $cabecalho .='<br />   < connectorParam
name="'.$escrever['connParam'].'">';
            $cabecalho .='<br />   < simpleCondition
role="'.$escrever['condicao'].'"/ >';
            $cabecalho .='<br />   < simpleAction
role="'.$escrever['acao'].'"/ >';
        }
        $cabecalho .='<br /><  /connectorBase ><br /><  /head >';

        //midias de entrada
        $cabecalho .='<br /><br /><  body >';
        $cabecalho .='<br /><  port id ="entry"
component="'.$mediaPrinc['id'].$mediaPrinc['nome'].'">';

        //midias (propriedades)
        while ($escrever2 = mysql_fetch_array($queryPropriedades)){
            $cabecalho .='<br /><  media
id="'.$escrever2['idArquivo'].$escrever2['nome'].'" src="'.$escrever2['caminho'].'"/ > ';
            if ($escrever2['tipo'] <> 'mp3' ){
```

```
                $cabecalho .='<br />   < property name="width"
value="'.$escrever2['largura'].'"/ >';
                $cabecalho .='<br />   < property name="heigth"
value="'.$escrever2['altura'].'"/ >';
                $cabecalho .='<br />   < property name="left"
value="'.$escrever2['posicaoX'].'"/ >';
                $cabecalho .='<br />   < property name="top"
value="'.$escrever2['posicaoY'].'"/ >';
                $cabecalho .='<br />   < property name="zIndex"
value="3"/ >';
            }
        }
        $cabecalho .='<br /><br />';

        //links
        while ($escrever3 = mysql_fetch_array($queryLinks)){
            $cabecalho .='<br />< link id="'.$escrever3['linkId'].'"
xconnector="'.$escrever3['nome'].'" >';
            $cabecalho .='<br />   < bind
role="'.$escrever3['conector'].'" component="'.$escrever3['id'].$escrever3['arquivo'].'"/
>';
            $cabecalho .='<br />   < bind role="'.$escrever3['acao'].'"
component="'.$escrever3['id'].'" >';
            $cabecalho .='<br />   < /bind>';
            $cabecalho .='<br />< /link>';
        }
        $cabecalho .='<br />< /body><br />< /ncl>';
        echo $cabecalho; //Imprime código NCL
        ?>
```

4.2.5 Controle da execução do vídeo

Para o controle efetivo de execução de vídeo foi utilizado o HTML Vídeo Player (VIDEO JS).

Através dele, utilizando comandos de Javascript, foi possível realizar o controle do tempo de execução do vídeo principal e assim realizar a interação com as demais mídias.

Inicialmente foi necessário obter o tempo atual do vídeo:

```
- Obter o tempo atual do vídeo:
function getTempoAtual(){
        var tempoAtual = myPlayer.currentTime();
        tempoCerto = tempoAtual.toFixed(1);
        return tempoCerto;
}
```

```
- Obter o tempo total do vídeo:
function getTempoTotal(){
        var tempoTotal = myPlayer.duration();
        tempoDuracao = tempoAtual.toFixed(2);
      return tempoDuracao;}
```

Com a utilização do método setInterval (nativa do Javascript), a cada segundo a função showTempoAtual é chamada, enquanto o vídeo principal estiver em execução.

Por sua vez, a função showTempoAtual faz com que a “tempoAtual” receba dinamicamente o valor retornado da função getTempoAtual().

```
setInterval( function(){showTempoAtual()}, 100);


function showTempoAtual() {
document.getElementById('tempoAtual').innerHTML = getTempoAtual();
}
```

4.2.6 Ferramenta auxiliar – HTML Vídeo Player

Na implementação do aplicativo foram utilizadas, em conjunto, todas as tecnologias acima apresentadas para que o objetivo do projeto fosse alcançado.

Além dessas, foram utilizadas outras tecnologias para que a execução do vídeo fosse possível, sendo uma delas o VIDEOJS(HTML Vídeo Player).

O VIDEO JS é uma API para execução de vídeo que utiliza HTML/CSS para exibição e marcação, além de javascript e flash para a interpretação dos dados para a execução do vídeo.

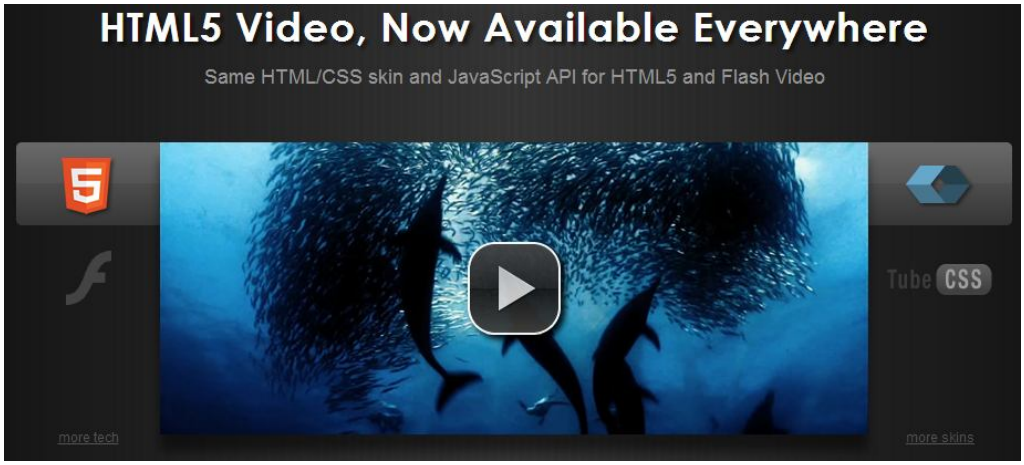


Figura 9 - VIDEOJS

Esta API é customizável, o que permitiu realizar todas as configurações necessárias para a adaptação do player para o Web Composer.

5. TUTORIAL PARA CRIAÇÃO DE PROJETOS

Como o objetivo do Web Composer NCL é criar, executar e gerar o código NCL dos projetos de maneira simples e rápida, sua interface foi criada para que o usuário, de maneira intuitiva, perceba quais dados devem ser preenchidos.

Primeiramente deve-se acessar a ferramenta:

Figura 10 – Web Composer NCL.

Para criar uma aplicação para TV Digital através do Web Composer NCL, os seguintes passos devem ser seguidos:

1. Fazer cadastro no sistema.

O usuário deverá informar seus dados para realizar seu cadastro e poder acessar a ferramenta.

Figura 11 – Cadastro no sistema Web Composer NCL.

2. Realizar Log In.

Após realizar o cadastro no sistema o usuário deve fazer o log in para iniciar a criação dos seus projetos.

Figura 12 – Log In no aplicativo.

3. Crie um projeto.

O primeiro passo a ser realizado, após o “log in” na ferramenta, é a criação de um projeto, dando um nome para o mesmo.

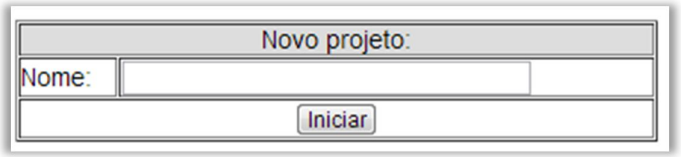


Figura 13 - Criar projeto.

4. Visualizar o projeto criado e clicar em “Itens”.

Depois de criado o projeto, o usuário deve primeiramente fazer o Upload dos arquivos que deseja utilizar no projeto. Para isto deve clicar no menu “Itens”.

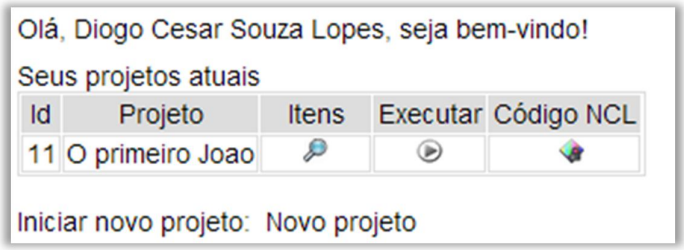


Figura 14 – Visualizar projetos.

5. Clicar em “Upload” para fazer upload das mídias a serem utilizadas.

Após clicar no menu “Itens” o usuário terá a opção “Fazer Upload”. A tela abaixo será aberta. Neste momento o usuário deve escolher o(s) arquivo(s) desejado(s) e enviar os mesmo para o servidor do aplicativo.

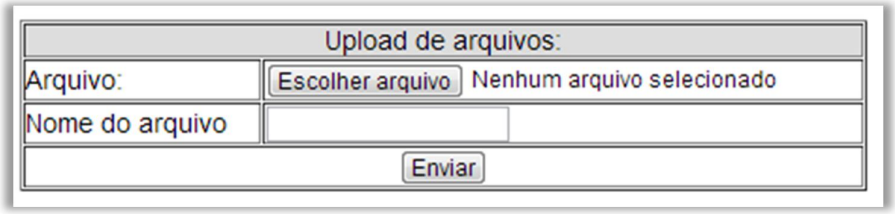


Figura 15 – Upload de arquivos.

6. Após o upload das mídias, clique em “Configurações” para Configurar as mídias (propriedades) e definir qual será o vídeo principal do projeto.

Basta selecionar um dos arquivos de vídeo enviados pelo usuário no combo “Vídeo principal” para definir qual será a mídia principal do projeto.

Figura 16 – Configuração das mídias.

7. Definir o relacionamento das mídias (em que momento as mídias irão interagir com o vídeo principal).

Após ser definido qual será a mídia principal o usuário terá a opção de configurar os links, os quais serão realizados entre os demais arquivos e a mídia principal.

Para isto ele deverá clicar no item “Links”, que aparecerá à frente da mídia principal.

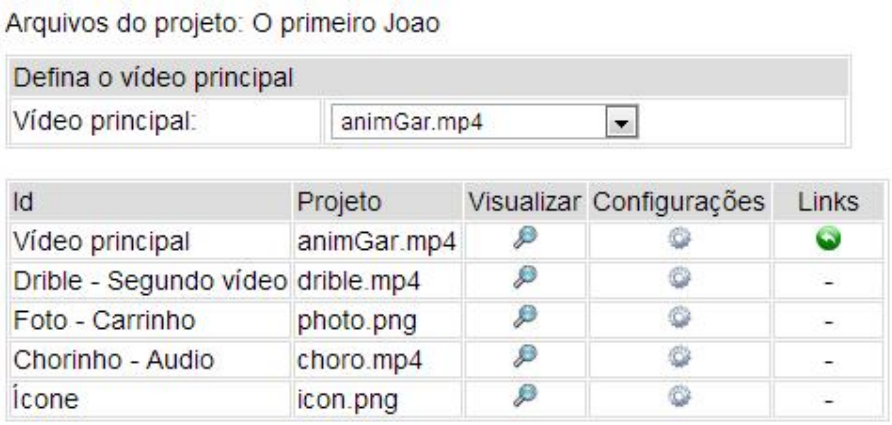


Figura 17 – Definição dos links.

Após clicar em “Links” a tela abaixo será aberta. O usuário poderá definir qual média irá interagir com a mídia principal e em qual momento ele quer que a interação ocorra.

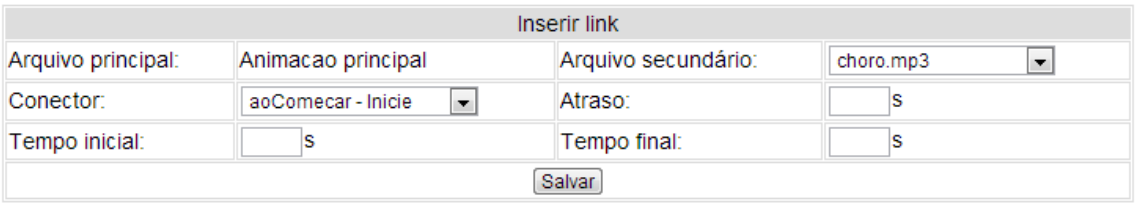


Figura 18 – Inserção de links.

8. Testar o projeto.

Após a upload e configuração das interação das mídias o usuário poderá testar seu projeto clicando em “Executar”.

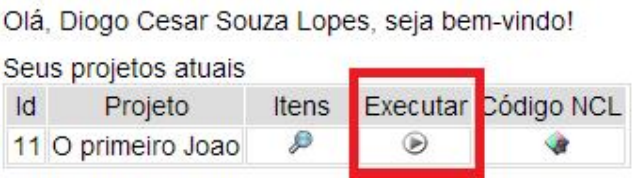


Figura 19 – Testar o projeto.

9. Exportar o NCL gerado a partir das interações criadas.

O usuário também poderá exportar o código NCL do projeto criado clicando em “Código NCL”.

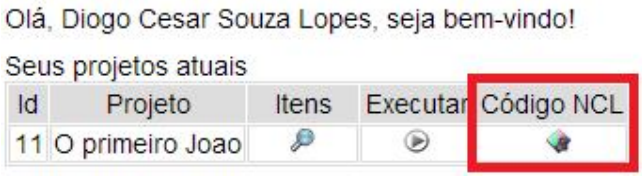


Figura 20 – Exportar o código NCL do projeto.

6. WEB COMPOSER NCL – EXEMPLO PRÁTICO.

O exemplo prático deste trabalho, explicitado e demonstrado abaixo, baseia-se no exemplo exposto no Livro "Programando em NCL 3.0", 2ª edição (SOARES E BARBOSA, 2012).

6.1 O primeiro João

O Primeiro João é baseado em um vídeo, uma animação de mesmo nome produzida por André Castelão, que por sua vez foi baseado nas crônicas de Mané Garrincha, escritas por Gerson Soares. A animação conta por que Garrincha passou a chamar de João todos os seus marcadores. A história se passa em um campo de pelada, onde Mané, ainda pré-adolescente, pobre, já fazia sua história. Ao ser marcado impiedosamente, Garrincha arrasa, com dribles desconcertantes, o time adversário e, principalmente, seu marcador, o pobre primeiro João; dribles esses que seriam repetidos na gloriosa carreira do ídolo brasileiro (SOARES E BARBOSA, 2012).

6.2 Sincronismo de mídia

Durante o vídeo da animação *O Primeiro João*, Garrincha dá o seu famoso drible em que leva os marcadores para a ponta, finge que vai avançar e retorna, por diversas vezes, até que finalmente realiza o drible e passa. Esse drible foi aplicado pelo Mané por diversas vezes em jogos oficiais, inclusive na Copa do Mundo pela seleção brasileira. Em outro trecho do vídeo, mais à frente, aparece o Garrincha e seu marcador, caído no chão, após um drible desconcertante. Cena idêntica aconteceu em um jogo oficial do Botafogo e Vasco, anos depois (SOARES E BARBOSA, 2012).

A aplicação ilustra como é fácil, em NCL, introduzir vários objetos de mídia sincronizados no tempo. O mesmo foi realizado na aplicação objetivo deste projeto.

Utilizamos no exemplo prático:

1. uma música de fundo (um chorinho), que deverá começar assim que terminar a apresentação inicial do vídeo e começar a animação propriamente dita;
2. um outro objeto de vídeo, que deverá ser exibido em paralelo e sincronizado com o famoso “drible do vaivém” do Mané, retratado na animação; e ainda:
3. uma outra imagem, uma foto, que deverá ser exibida junto com a cena do marcador caído no chão.

O novo vídeo acrescentado é uma réplica, uma clonagem do drible real aplicado por Garrincha, realizado por garotos da Comunidade Beira Rio, no Rio de Janeiro, produzido pelo Ponto de Cultura CIDS-VG, daquela comunidade. Ele mostra garotos descalços em um campo de várzea, como era Garrincha na época em que aplicou o drible no primeiro João. O mesmo Ponto de Cultura é também responsável pela foto que retrata exatamente a mesma situação ocorrida na animação, quando o marcador do Garrincha vai ao chão, foto representada pelos mesmos garotos atores (SOARES E BARBOSA, 2012).

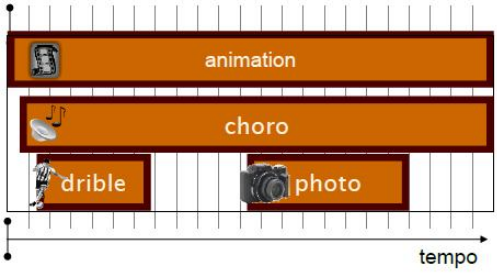


Figura 21 – Visão temporal – Exemplo prático.[15]

Para definir todos esses objetos de mídia, a NCL faz uso de seu elemento <media>, como ilustrado abaixo:

[15] Fonte: SOARES E BARBOSA, 2012

```
<media id="animation" src="../media/animGar.mp4"/>
<media id="choro" src="../media/choro.mp3"/>
<media id="drible" src="../media/drible.mp4"/>
<media id="photo" src="../media/photo.png"/>
```

Todo elemento <media> tem um identificador, definido pelo atributo *id*, para que possa ser posteriormente referenciado. Ele tem também um atributo denominado *src*, que contém um URL para a localização do conteúdo. Em todos os casos de nosso exemplo, os conteúdos estão no diretório “/media”, filho do mesmo diretório onde está definida a aplicação. É usual, em aplicações para TV digital, o atributo *src* referenciar um fluxo elementar MPEG System.

Importante observar que todo elemento NCL deve começar com o caractere “<”. Quando o elemento não tem nenhum conteúdo nem elementos filhos, ele deve terminar sempre com “/>”. Caso contrário, a definição dos atributos do elemento termina com o caractere “>”.

Após a definição de seu conteúdo ou elementos filhos, o elemento termina repetindo seu nome envolvido com os caracteres “</” na frente e o caractere “>” atrás, como visto a seguir, na redefinição do elemento <media id=“animation”.../> da Listagem acima que, na Listagem abaixo representa o vídeo da animação com dois elementos filhos aninhados, denominados <area>.

```
<media id="animation" src="../media/animGar.mp4"
descriptor="screenDesc">
    <area id="segDrible" begin="12s"/>
    <area id="segPhoto" begin="41s"/>
</media>
```

Os elementos <area> delimitam trechos (no tempo ou espaço) do conteúdo do seu objeto de mídia pai. No caso, são delimitados o instante em que, na animação, o Garrincha inicia o drible de vaivém e o instante, após outro drible, em que o marcador aparece caído no chão. O primeiro instante é delimitado por meio do atributo *begin* igual a 12 segundos, e o segundo pelo atributo *begin* igual a 41

segundos. Como não foi especificado o valor do atributo *end*, ele é assumido, por default, como tendo o mesmo valor do final do vídeo da animação, em ambos os casos. Note que todo elemento <area> também possui um identificador (atributo *id*).

Nosso próximo passo no projeto da aplicação é definir em que posição da tela os vários objetos de mídia serão exibidos. A figura abaixo apresenta a visão de leiaute desejada. São definidas regiões para exibição do vídeo da animação ("screenReg"), em tela cheia, e uma região para exibição do vídeo do drible e da foto ("frameReg").

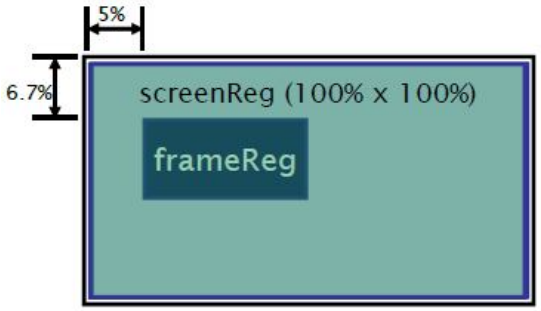


Figura 22 – Visão de layout – Exemplo prático.[16]

A definição dos espaços de exibição pode ser realizada por meio de elementos <property>, filhos dos elementos <media> que representam cada objeto de mídia da aplicação, como ilustrado abaixo:

```
<media id="animation" src="../media/animGar.mp4">
    <area id="segDrible" begin="11.5s"/>
    <area id="segPhoto" begin="41s"/>
    <property name="width" value="100%"/>
    <property name="heigth" value="100%"/>
    <property name="zIndex" value="2"/>
</media>
<media id="choro" src="../media/choro.mp3"/>
    <media id="drible" src="../media/drible.mp4">
    <property name="left" value="5%"/>
```

[16] Fonte: SOARES E BARBOSA, 2012

```
        <property name="top" value="6.7%"/>
        <property name="width" value="18.5%"/>
        <property name="heigth" value="18.5%"/>
        <property name="zIndex" value="3"/>
    </media>
    <media id="photo" src="../media/photo.png">
        <property name="left" value="5%"/>
        <property name="top" value="6.7%"/>
        <property name="width" value="18.5%"/>
        <property name="heigth" value="18.5%"/>
        <property name="zIndex" value="3"/>
        <property name="explicitDur" value="5s"/>
    </media>
```

Todo elemento <property> possui um identificador válido no escopo do objeto de mídia, definidos por seu atributo *name*. Por exemplo, “left”, “top” “width”, “height”, “right”, e “bottom” são propriedades que definem a área de apresentação em relação à tela total do dispositivo de exibição. Essas propriedades podem ser definidas de forma absoluta ou como uma porcentagem, sempre com relação à tela total do dispositivo de exibição. A propriedade “zIndex” especifica como fica a sobreposição de regiões. Uma região de maior valor para *zIndex* se sobrepõe àquela de menor valor. Assim, no exemplo, são definidas uma região para exibição do vídeo da animação em tela cheia, e uma região para exibição do vídeo do drible e da foto se sobrepondo ao vídeo da animação.

Elementos <property> podem ser usados para a definição do valor de qualquer propriedade de um objeto de mídia, e não apenas aquelas que definem espaços de exibição. Por exemplo, para o objeto de mídia representando a foto, uma duração de 5s pode ser associada ao elemento <media> correspondente, por meio de sua propriedade “explicitDur”.

Finalmente, podemos agora definir os relacionamentos de sincronização entre os vários objetos de mídia. Em NCL, o elemento <body> agrupa todos os objetos de um documento e seus relacionamentos.

O elemento <body> é único em um documento NCL e pode ter um ou mais elementos <port> como filho. O elemento <port> indica por onde pode começar uma exibição do documento. No nosso exemplo, como ilustrado abaixo, a exibição começa sempre pela apresentação do vídeo da animação.

```
<body>
    <port id="entry" component="animation"/>
    <!—definição dos diversos elementos de <media> do documento -->
    <!—definição dos diversos relacionamentos, elementos <link> do documento -->
</body>
```

Relacionamentos são definidos por elementos <link>, para nosso exemplo de *O Primeiro João*. O relacionamento ilustrado define que, ao ser iniciada a exibição do vídeo da animação, o áudio com o chorinho também deve ser iniciado, mas 5 segundos depois (como determinamos no enunciado do exemplo, esse objeto de mídia é apresentado após os créditos iniciais do vídeo da animação, que duram 5 s).

```
<link id="lMusic" xconnector="onBeginStart_delay">
    <bind role="onBegin" component="animation"/>
    <bind role="start" component="choro">
        <bindParam name="delay" value="5s"/>
    </bind>
</link>
```

Um relacionamento em NCL é especificado referindo-se a uma relação, dada pelo valor URI do atributo *xconnector* do elemento <link>, e pelos atores que exercerão os papéis da relação, dados pelos elementos <bind>. Um elemento <bind> especifica o papel da relação, através de seu atributo <role>, e a interface que exercerá o papel, dada pelos atributos *component*, que seleciona um objeto a ser relacionado, e *interface*, que seleciona uma interface desse objeto. Por enquanto, as únicas interfaces que definimos foram por meio de elementos <area>.

Quando não especificada, o valor default para a *interface* assume uma área contendo todo o conteúdo do objeto.

No relacionamento definido pelo elemento <link> “lMusic”, a relação referenciada é definida pelo elemento <causalConnector>, filho do elemento <connectorBase>, o papel “onBegin” do conector é associado ao componente “animation”, que é associado ao vídeo da animação, definindo que, ao começar a exibição do vídeo, deve-se dar partida (*role* = “start”) à exibição do chorinho, mas 5 segundos a partir do início do vídeo.

```
<connectorBase>
<causalConnector id="onBeginStart_delay">
<connectorParam name="delay"/>
<simpleCondition role="onBegin"/>
<simpleAction role="start" delay="$delay" max="unbounded" qualifier="par"/>
</causalConnector>
</connectorBase>
```

A relação é causal, onde a condição é dada pelo papel “onBegin”, e a ação é “start”, com o parâmetro “delay” (retardo) a ser definido pelo relacionamento (o elemento <link>), quando da sua especificação. O atributo *max*=“unbounded” definido na ação especifica que um número ilimitado de atores pode exercer esse papel. Quando existe mais de um ator para o papel, o atributo *qualifier* define a forma como as ações devem ser executadas, se em paralelo ou em sequência.

Abaixo apresentamos a visão estrutural da aplicação com todos os relacionamentos entre os vários objetos de mídia definidos:

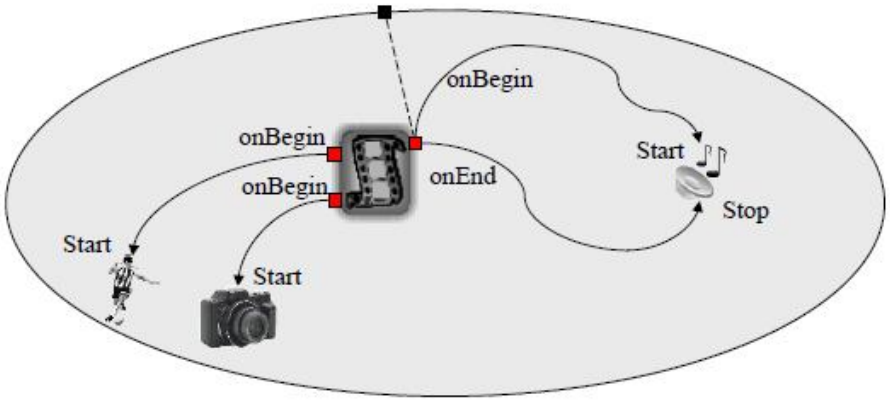


Figura 23 – Visão estrutural – Exemplo prático.[17]

Bases de conectores são definidas como elemento filho do elemento <head>, que define, como veremos, as partes reusáveis de uma aplicação. No caso, um mesmo conector pode ser usado por mais de um elemento <link>. Abaixo a apresentação da definição completa do elemento <head> da aplicação exemplo:

```
<head>
<connectorBase>
<causalConnector id="onBeginStart_delay">
<connectorParam name="delay"/>
<simpleCondition role="onBegin"/>
<simpleAction role="start" delay="$delay" max="unbounded" qualifier="par"/>
</causalConnector>
<causalConnector id="onBeginStart">
<simpleCondition role="onBegin"/>
<simpleAction role="start" max="unbounded" qualifier="par"/>
</causalConnector>
<causalConnector id="onEndStop">
<simpleCondition role="onEnd"/>
<simpleAction role="stop" max="unbounded" qualifier="par"/>
</causalConnector>
```

[17] Fonte: SOARES E BARBOSA, 2012

```
</connectorBase>
</head>
```

Podemos agora definir todo o elemento <body> e seus filhos, conforme apresentado abaixo:

```
<body>
<port id="entry" component="animation"/>
<media id="animation" src="../media/animGar.mp4">
<area id="segDrible" begin="11.5s"/>
<area id="segPhoto" begin="41s"/>
<property name="width" value="100%"/>
<property name="heigth" value="100%"/>
<property name="zIndex" value="2"/>
</media>
<media id="choro" src="../media/choro.mp3"/>
<media id="drible" src="../media/drible.mp4">
<property name="left" value="5%"/>
<property name="top" value="6.7%"/>
<property name="width" value="18.5%"/>
<property name="heigth" value="18.5%"/>
<property name="zIndex" value="3"/>
</media>
<media id="photo" src="../media/photo.png">
<property name="left" value="5%"/>
<property name="top" value="6.7%"/>
<property name="width" value="18.5%"/>
<property name="heigth" value="18.5%"/>
<property name="zIndex" value="3"/>
<property name="explicitDur" value="5s"/>
</media>
<link id="lMusic" xconnector="onBeginStart_delay">
<bind role="onBegin" component="animation"/>
```

```
<bind role="start" component="choro">
<bindParam name="delay" value="5s"/>
</bind>
</link>
<link id="lDrible" xconnector="onBeginStart">
<bind role="onBegin" component="animation" interface="segDrible"/>
<bind role="start" component="drible"/>
</link>
<link id="lPhoto" xconnector="onBeginStart">
<bind role="onBegin" component="animation" interface="segPhoto"/>
<bind role="start" component="photo"/>
</link>
<link id="lEnd" xconnector="onEndStop">
<bind role="onEnd" component="animation"/>
<bind role="stop" component="choro"/>
</link>
</body>
```

O elemento <link> “lDrible” define que, ao começar (*role* “onBegin”) o trecho do vídeo da animação correspondente ao drible de vaivém do Garrincha (*component* “animation” *interface* “segDrible”), o vídeo do drible deve ser iniciado (*role* “start”).

De forma análoga, o elemento <link> “lPhoto” define que, ao começar (*role* “onBegin”) o trecho do vídeo da animação correspondente ao momento em que o marcador cai no chão (*component* “animation” *interface* “segPhoto”), a foto do marcador deve ser exibida (*role* “start”).

Finalmente, o elemento <link> “lEnd” define que, ao findar (“onEnd”) o vídeo da animação, o chorinho deve parar (“stop”) de tocar.

Os elementos <head> e <body> devem ser declarados como filhos do elemento <ncl>, completando a definição do documento: a aplicação declarativa NCL.

Abaixo está ilustrada a aplicação completa. Note que toda aplicação NCL começa com a linha: <?xml version="1.0" encoding="ISO-8859-1"?> seguida da especificação do elemento <ncl>, onde o atributo *id* define um identificador para a

aplicação e o atributo *xmlns* define o espaço de nomes (namespace) onde estão definidos os esquemas do perfil NCL EDTV, para verificação, pelo parser XML, da validade da aplicação.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Exemplo de sincronismo sem a interação do usuário e com reúso apenas de relações-->
<ncl id="syncEx" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>
<connectorBase>
<causalConnector id="onBeginStart_delay">
<connectorParam name="delay"/>
<simpleCondition role="onBegin"/>
<simpleAction role="start" delay="$delay" max="unbounded" qualifier="par"/>
</causalConnector>
<causalConnector id="onBeginStart">
<simpleCondition role="onBegin"/>
<simpleAction role="start" max="unbounded" qualifier="par"/>
</causalConnector>
<causalConnector id="onEndStop">
<simpleCondition role="onEnd"/>
<simpleAction role="stop" max="unbounded" qualifier="par"/>
</causalConnector>
</connectorBase>
</head>
<body>
<port id="entry" component="animation"/>
<media id="animation" src="../media/animGar.mp4">
<area id="segDrible" begin="11.5s"/>
<area id="segPhoto" begin="41s"/>
<property name="width" value="100%"/>
<property name="heigth" value="100%"/>
```

```
<property name="zIndex" value="2"/>
</media>
<media id="choro" src="../media/choro.mp3"/>
<media id="drible" src="../media/drible.mp4">
<property name="left" value="5%"/>
<property name="top" value="6.7%"/>
<property name="width" value="18.5%"/>
<property name="heigth" value="18.5%"/>
<property name="zIndex" value="3"/>
</media>
<media id="photo" src="../media/photo.png">
<property name="left" value="5%"/>
<property name="top" value="6.7%"/>
<property name="width" value="18.5%"/>
<property name="heigth" value="18.5%"/>
<property name="zIndex" value="3"/>
<property name="explicitDur" value="5s"/>
</media>
<link id="lMusic" xconnector="onBeginStart_delay">
<bind role="onBegin" component="animation"/>
<bind role="start" component="choro">
<bindParam name="delay" value="5s"/>
</bind>
</link>
<link id="lDrible" xconnector="onBeginStart">
<bind role="onBegin" component="animation" interface="segDrible"/>
<bind role="start" component="drible"/>
</link>
<link id="lPhoto" xconnector="onBeginStart">
<bind role="onBegin" component="animation" interface="segPhoto"/>
<bind role="start" component="photo"/>
</link>
<link id="lEnd" xconnector="onEndStop">
```

```
<bind role="onEnd" component="animation"/>
<bind role="stop" component="choro"/>
</link>
</body> </ncl>
```

A Figura 24 ilustra dois momentos da exibição, o início do drible do Garrincha e o início da apresentação da foto, obtidos durante a execução da aplicação na implementação de referência do middleware Ginga-NCL.

Figura 24 – Cenas da aplicação – Exemplo prático.[18]

[18] Fonte: SOARES E BARBOSA, 2012

CONSIDERAÇÕES FINAIS

O objetivo desta monografia foi desenvolver e apresentar a ferramenta aqui denominada WEB Composer NCL, que permite aos usuários criar de maneira simples e rápida aplicativos para a TV Digital sem a necessidade do conhecimento da programação em NCL.

Como é possível observar na seção “3.4 - Ferramentas de autoria”, os softwares apresentados possuem diversos recursos e possibilitam aos usuários criarem aplicações para a TV Digital muito robustas, contudo exigem um alto nível de conhecimento dos utilizadores, o que limita o seu uso à públicos específicos.

O desafio proposto neste trabalho foi exatamente a eliminação destas barreiras na criação de aplicações para a TV Digital.

Para tanto, a metodologia empregada para a execução deste trabalho envolveu uma ampla análise dos ambientes de autoria (Seção 3.4), das ferramentas existentes para a manipulação e execução de vídeos (Seção 4.2.3) e das demais tecnologias necessárias (Seção 4.1) para execução em ambiente WEB.

Durante o andamento do projeto houve dificuldades em conseguir material que discorresse sobre o desenvolvimento de ferramentas de autoria voltadas para o ambiente WEB, exigindo assim que a criação fosse baseada em conceitos das ferramentas desktop, mas com fundamentos nos padrões WEB.

Superando as dificuldades encontradas, através da elaboração de módulos para a criação, execução e exportação do código NCL, foi possível construir um protótipo que expande para o mundo Web a possibilidade de criação de aplicativos para a TVDI.

Sendo assim, o WEB Composer NCL é um protótipo de uma ferramenta de autoria NCL para o ambiente WEB, limitado a criação de aplicações básicas para a TV Digital. A partir deste desenvolvimento abre-se um vasto campo para futuros trabalhos em torno da ampliação da capacidade de criação de aplicações mais robustas para a TV Digital.

BIBLIOGRAFIA

ABNT NBR 15606-2 (2010) – Associação Brasileira de Normas Técnicas, **"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", Sistema Brasileiro de TV Digital Terrestre, NBR 15606-2.

ABNT NBR 15606-5 (2010) – **Associação Brasileira de Normas Técnicas. "Televisão digital terrestre** — Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações", Sistema Brasileiro de TV Digital Terrestre, NBR 15606-5.

ABNT/CEET (2010) "**Televisão digital terrestre** - Codificação de dados e especificações de transmissão para transmissão digital – Parte 4: Ginga-J – Ambiente para a execução de aplicações procedurais".

ANATEL, **Agência Nacional de Telecomunicações**. "TV Digital", Brasília, 2001.

NETO, C.S.S. **Clube NCL Dados** – Construindo programas audiovisuais interativos utilizando a NCL 3.0 e a ferramenta Composer. Disponível em: http://www.ncl.org.br/documentos/TutorialNCL3.0-2ed.pdf. Acesso em: 11 mar. 2013.

Clube NCL – **TV Digital interativa no Brasil se faz com Ginga** – Fundamentos, Padrões, Autoria declarativa e Usabilidade. Disponível em: http://www.ncl.org.br/documentos/JAI2008.pdf. Acesso em: 11 mar. 2013.

CTIC – **Centro de pesquisa e desenvolvimento em tecnologias digitais para informação e comunicação**. Disponível em: http://www.ctic.rnp.br/. Acesso em: 11 mar. 2013.

DTV. **Site oficial da TV Digital brasileira**. Disponível em: http://www.dtv.org.br, Acesso em: 11 mar. 2013.

Eclipse.org (2008) **"Eclipse IDE"**. Disponível em: http://www.eclipse.org, Acesso em: 11 mar. 2013.

Eclipse.org GEF (2008) **"Graphical Editing Framework"**. Disponível em: http://www.eclipse.org/gef/, Acesso em: 11 mar. 2013.

GINGA. **GINGA DIGITAL TV MIDDLEWARE SPECIFICATION**. Disponível em http://www.ginga.org.br/. Acesso em: 11 mar. 2013

GINGA-NCL. GINGA-NCL - DECLARATIVE DTV MIDDLEWARE.
Disponível em: http://www.gingancl.org.br/. Acesso em: 11 mar. 2013

LABORATÓRIO TELEMÍDIA; **Building Interactive Audiovisual Programs Using NCL 3.0 and the Composer Tool** - 2nd edition (NCL 3.0, portuguese) - Revised. Disponível em: http://www.telemidia.puc-rio.br/?q=pt-br/node/542. Acesso em: 11 mar. 2013

MONTEZ, C.; BECKER, V. **TV Digital Interativa**: conceitos, desafios, perspectivas para o Brasil. 2º edição. Florianópolis: UFSC. 2005.

NCL. **NESTED CONTEXT LANGUAGE**. Disponível em: http://www.ncl.org.br/. Acesso em: 11 mar. 2013

Portal do software público brasileiro – **Ginga: a TV Digital no Software Público**. Disponível em: http://www.softwarepublico.gov.br/ginga-a-tv-digital-no softwarepublico. Acesso em: 11 mar. 2013

PUC – RIO - **Composer: um ambiente de autoria de documentos NCL para TV digital interativa**. Disponível em: http://www.telemidia.pucrio.br/sites/telemidia.puc-rio.br/files/2007_06_laiola.pdf Acesso em: Acesso em: 11 mar. 2013

SBTVD (2008). **Sistema Brasileiro de TV Digital:** Disponível em: http://sbtvd.cpqd.com.br/. Acesso em: 11 mar. 2013

SEBRAE – Aplicações para TV Digital Interativa - Disponível em: http://www.biblioteca.sebrae.com.br/bds/BDS.nsf/17466F805351BCA303256FD5004784DB/$File/NT000A612A.pdf. Acesso em: 09 nov. 2010.

SOARES, L.F.G.; BARBOSA, S.D.J. **Programando em NCL 3.0**: Desenvolvimento de Aplicações para o Middleware Ginga. 2.ed. Pontifícia Universidade do Rio de Janeiro. Rio de Janeiro, 2012.

UFRJ – **TV Digital:** A polêmica dos conversores. Disponível em: http://www.olharvirtual.ufrj.br/2006/imprimir.php?id_edicao=189&codigo=3.Acesso em: 11 mar. 2013.

UFF – **Middleware Ginga** – Disponível em: http://www.midiacom.uff.br/~debora/fsmm/trab-2008-2/middleware.pdf. Acesso em: 11 mar. 2013

VIDEO JS: **HTML5 Video Player**. Disponível em: http://videojs.com/. Acesso em: 11 mar. 2013