UNIVERSIDADE DE BRASÍLIA - UNB
FACULDADE DE TECNOLOGIA - FT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LUACOMP: FERRAMENTA DE AUTORIA DE APLICAÇÕES PARA TV DIGITAL

PAULO JOSÉ DE SOUZA JÚNIOR

ORIENTADOR: Paulo Roberto de Lira Gondim

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM - 375/09

BRASÍLIA / DF: MARÇO/2009

UNIVERSIDADE DE BRASÍLIA - UNB
FACULDADE DE TECNOLOGIA - FT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LUACOMP: FERRAMENTA DE AUTORIA DE APLICAÇÕES PARA TV DIGITAL

PAULO JOSÉ DE SOUZA JÚNIOR

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA;

APROVADA POR:

_____

PROFESSOR PAULO ROBERTO DE LIRA GONDIM (DOUTOR)
(ORIENTADOR)

_____

PROFESSORA JULIANA FERNANDES CAMAPUM (DOUTOR)
(EXAMINADOR INTERNO)

_____

PROFESSOR GEORGES AMVAME NZE (DOUTOR)
(EXAMINADOR EXTERNO)

DATA: BRASÍLIA/DF, 41 DE MARÇO DE 2009.

FICHA CATALOGRÁFICA

SOUZA JÚNIOR, PAULO JOSÉ DE.

LuaComp: uma ferramenta de autoria de aplicações para TV digital. [Distrito Federal] 2009. viii, 143 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2009).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica.

1. Lua	4.Ginga-NCLua
2. LuaOnTV	5. Ferramenta de Autoria
3. LuaComp	6. TV Digital Interativa
I. ENE/FT/UnB.	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SOUZA JÚNIOR, P. J. LuaComp: uma ferramenta de autoria de aplicações para TV digital. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM 375/09, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 143 p.

CESSÃO DE DIREITOS

AUTOR: PAULO JOSÉ DE SOUZA JÚNIOR
TÍTULO: LuaComp: uma ferramenta de autoria de aplicações para TV digital.

GRAU: Mestre ANO: 2009.

_____

Paulo José de Souza Júnior
SCLN 316 Bl. D Aptº 200 - Asa Norte - Brasília
E-mail: paulojunior777@gmail.com

DEDICATÓRIA

Dedico esse trabalho inicialmente a JESUS, “o grande amigo que não abandona o amigo no meio do caminho”. Depois, não poderia esquecer o sacrifício que toda a minha família fez para que eu conseguisse este título tão importante. Também dedico à minha esposa, Anna Karine de Figueiredo Farias, ao meu brilhante filho, Matheus Figueiredo de Souza, e minha pequena princesa, a minha filha, Maria Isadora Figueiredo de Souza.

AGRADECIMENTOS

Ao meu orientador, Paulo Roberto de Lira Gondim, que me deu a oportunidade de sentar nos bancos escolares de uma universidade federal como a UnB.

Ao amigo inesquecível, João Henrique Allemand, que foi fundamental para a minha aprovação para seguir como aluno regular.

Ao amigo e exemplo de pai, Carlos Oscar Cruz, que está sempre me apoiando e torcendo pelo meu crescimento profissional e espiritual.

A amiga, prima, “tia” e carinhosa, Maria da Sallete Coutinho, que também tem me apoiado muito desde que cheguei à Brasília.

LUACOMP: UMA FERRAMENTA DE AUTORIA DE APLICAÇÕES PARA TV DIGITAL

RESUMO

Esta dissertação apresenta uma ferramenta de autoria para aplicações híbridas (declarativas e procedurais) em Ginga-NCLua para a TV Digital interativa do Brasil. A demora na disponibilização do Ginga-J, módulo do Ginga para códigos procedurais, foi a maior motivação para este trabalho. A necessidade de se criar aplicações procedurais com entrada e saída de dados, e as vantagens de se usar a linguagem Lua e o Ginga-NCLua (uma classe de objetos de mídia Lua), foram um importante fator de incentivo. O LuaComp é uma ferramenta de autoria que possibilita ao usuário a rápida criação de aplicações. O LuaComp explorando funcionalidades do LuaOnTV. O LuaOnTV é um *framework* para utilização dos componentes gráficos para entrada e saída de dados em aplicações interativas implementada sob o paradigma da programação orientada a objetos. O LuaComp, além de uma interface de fácil uso e com os principais recursos das mais destacadas ferramentas de autoria, *Delphi* e *Visual Basic*, também apresenta as visões como recursos utilizados na maioria das ferramentas de autoria para TV Digital. O LuaComp também se destaca por apresentar um resultado WYSIWYG e permitir a criação e utilização de *templates* em arquivos XML. O Luacomp implementa aplicações NCL com pouco sincronismo, mas o suficiente para sincronizar a aplicação NCLua com a programação e/ou vídeo principal. Em resumo, o sistema proposto visa agilizar a criação de aplicações voltadas para TVs Digitais interativas, abstraindo do autor toda, ou pelo menos parcialmente, a complexidade de se programar em NCL e Lua.

Palavras-chave: Ferramenta de autoria – Ginga – Lua – LuaComp –NCLua – TV Digital.

LUACOMP: UMA FERRAMENTA DE AUTORIA EM LINGUAGEM LUA

ABSTRACT

This work presents an authoring tool to create hybrid (declarative and procedural) applications of Ginga-NCLua for interactive digital TV of Brazil. The unavailability of Ginga-J middleware was the biggest motivation for this work. The demand of input/output applications, the availability of Ginga-NCLua (a class of Lua media objects) and the advantages of Lua language were an important incentive. The LuaComp is an authoring tool to makes possible a fast creation of applications, exploring LuaOnTV, a framework of graphical components to interactive applications of input and output of data implemented under the paradigm of object-oriented programming (OOP). The LuaComp is an interface of easy use with the most advanced authoring tool, Delphi and Visual Basic. In the same way that in the most authoring tool of interactive digital tv, in which it is based, in LuaComp, the abstractions are defined using views that allow to simulate a specific type of edition (structural, temporal, layout and textual). This tool also presents the visions of the authorship tools for digital TV. The LuaComp also presents a result WYSIWYG and allow create templates to archives XML. The Luacomp implements NCL applications with little synchronism but sufficient to synchronize the NCLua applications with the programming and/or video. In summary, the considered system tries to make easier the creation of interactive digital TV applications, abstracting from the author all, or at least some complexity of programming in NCL and Lua.

Keywords: Authory tools – Digital TV – Ginga – Lua – LuaOnTV – NCLua.

ÍNDICE

Item **Página**

1 - INTRODUÇÃO 18
1.1 - MOTIVAÇÃO 18
1.2 - OBJETIVOS 20
1.3 - METODOLOGIA 21
1.4 - ORGANIZAÇÃO DA DISSERTAÇÃO 21
2 - TV DIGITAL INTERATIVA 23
2.1 - INTRODUÇÃO 23
2.2 - TV DIGITAL 24
2.3 - SISTEMAS MUNDIAIS DE TV DIGITAL 26
2.4 - SISTEMA BRASILEIRO DE TV DIGITAL 28
2.5 - LINGUAGENS NOS PADRÕES DE TV DIGITAL 31
2.6 - *MIDDLEWARE* 34
2.7 - *MIDDLEWARE* GINGA 37
2.8 - INTERATIVIDADE 42
2.9 - CENÁRIOS DE APLICAÇÕES INTERATIVAS PARA TV DIGITAL 46
2.10 - USABILIDADE NA TV DIGITAL INTERATIVA 52
2.11 - CONCLUSÃO 60
3 - FERRAMENTAS DE AUTORIA 61
3.1 - INTRODUÇÃO 61
3.1.1 - *Delphi* 64
3.1.2 - *Visual Basic* 67
3.1.3 - *NetBeans* 68
3.1.4. - Eclipse 70
3.1.5 - *Dreamweaver* e *FrontPage* 72
3.1.6 - *Macromedia Flash* 74
3.1.7 - GRiNS 75
3.1.7 - *LimSee2* 76

3.1.8 - JAME AUTHOR 77
3.1.9 - *Cardinal Studio* 80
3.1.10 - *AltiComposer* 82
3.1.11 - *Composer* 84
3.2 CONCLUSÃO E ANÁLISE COMPARATIVA 89
4 - AUTORIA EM NCLUA PARA TV DIGITAL INTERATIVA 92
4.1 - INTRODUÇÃO 92
4.2 - LINGUAGEM LUA 92
4.3 - REQUISITOS PARA O LUACOMP 94
4.4 - LUAONTV 95
4.5 - LUACOMP 105
4.5.1 - Arquitetura LuaComp 106
4.6 - CONCLUSÃO 111
5 - UTILIZANDO E COMPARANDO O LUACOMP 112
5.1 - INTERFACE GRÁFICA 112
5.2 - VISÕES 117
5.2.1 - Visão de leiaute 117
5.2.2 - Visão estrutural 118
5.2.3 - Visão textual 119
5.2.4 - Visão temporal 120
5.3 - ARQUITETURA DO LUACOMP 120
5.4 - ESTUDO DE CASO 131
5.5 - ANÁLISE COMPARATIVA 131
5.6 - CONCLUSÃO 133
6 - CONCLUSÕES 135
6.1 - TRABALHO FUTURO 136
REFERÊNCIAS BIBLIOGRÁFICAS 137

ÍNDICE DE TABELAS

Tabela **Página**

Tabela 2.1 - Padrões e camadas da TV Digital.....27

Tabela 2.2 - Melhores características.....28

Tabela 2.3 - TV interativa x não interativa.....43

Tabela 2.4 - Tipos de STB.....44

Tabela 2.5 - MHP – *Multimedia Home Plataform*.....45

Tabela 3.1 - Tabela de comparação entre ferramentas de autoria.....90

Tabela 4.1 – Tabela de métodos dos componentes LuaOnTV.....105

Tabela 6.1 - Tabela comparativa entre ferramentas de autoria para TV Digital.....133

ÍNDICE DE FIGURAS

Figura **Página**

Figura 2.1 – A nova televisão 23
Figura 2.2 – Sinal analógico X Sinal digital 25
Figura 2.3 – *Middleware* 26
Figura 2.4 – Sistemas de televisão digital *broadcast* 27
Figura 2.5 – *Benchmark* 33
Figura 2.6 – Arquitetura de *middleware* 35
Figura 2.7 – Evolução e convergência dos *middlewares* até o GEM 36
Figura 2.8 – GEM como subconjunto dos outros *middlewares* 37
Figura 2.9 – Arquitetura Ginga 38
Figura 2.10 – Arquitetura Ginga 40
Figura 2.11 – Primeiro programa interativo para TV – CBS 1953 42
Figura 2.12 – Tipos de interatividades 46
Figura 2.13 – Tipos de aplicações 47
Figura 2.14 – EPG 47
Figura 2.15 – VOD 48
Figura 2.16 – Aplicações interativas 51
Figura 2.17 – Aplicações veiculadas na SKY do Brasil 54
Figura 2.18 – Exemplos da BBC 55
Figura 2.19 – Adaptação de conteúdo 55
Figura 2.20 – Aplicação 16:9 em tela 4:3 56
Figura 2.21 – Aplicação 4:3 em tela 16:9 56
Figura 2.22 – Produção para *widescreen* 56
Figura 2.23 – Adaptação para *display* 4:3 57
Figura 2.24 – Fontes usadas pela BBC 57

Figura 2.25 - Distribuição de fontes em padrão SDTV 58
Figura 2.26 - Teclas de navegação ou funções 59
Figura 3.1 – Ambiente de janelas interativas 65
Figura 3.2 – Tela do *Visual Basic* 68
Figura 3.3 – Tela do *NetBeans* 69
Figura 3.4 – Tela do *NetBeans* 70
Figura 3.5 – Tela do Eclipse 71
Figura 3.6 – Tela inicial do *XletView* 72
Figura 3.7 – Ferramentas de componentes 73
Figura 3.8 – Ferramentas de componentes 73
Figura 3.9 – Tela inicial do *Flash* 74
Figura 3.10 – Definição da resolução da aplicação 78
Figura 3.11 – Caixa de diálogo da aplicação 80
Figura 3.12 – Planos *Alticomposer* 83
Figura 3.13 – Mecanismos de ajuda e mensagens de erro 85
Figura 3.14 – Visão de leiaute 86
Figura 3.15 – Visão estrutural 86
Figura 3.16 – Visão textual 87
Figura 3.17 – Visão temporal 88
Figura 3.18 – Link do *Composer* com o emulador Ginga NCL Player 88
Figura 4.1 – Lua x Java 94
Figura 4.2 – Diagrama de classes - LuaOnTV 100
Figura 4.3 – Interatividade SKY 107
Figura 4.4 – LuaComp – Visão estrutural 108
Figura 4.5 – LuaComp – Visão leiaute 108
Figura 4.6 – LuaComp – Visão textual 109
Figura 4.7 – Arquitetura LuaComp 110

Figura 5.1 – Componentes do LuaComp 1.0 112
Figura 5.2 – LuaComp 113
Figura 5.3 – Barra de menus LuaComp 113
Figura 5.4 –Visões estrutural, textual e leiaute - LuaComp 114
Figura 5.5 – Módulo de criação de projeto 115
Figura 5.6 – Imagem de fundo - LuaComp 115
Figura 5.7 – Inspetor de componentes - LuaComp 116
Figura 5.8 – Salvando uma página - LuaComp 117
Figura 5.9 – Emulador Ginga x LuaComp 118
Figura 5.10 – Visão estrutural 118
Figura 5.11 – Visão textual 119
Figura 5.12 – Visão temporal - *Composer* 120
Figura 5.13 – Diagrama de classes LuaComp 121
Figura 5.14 - Modelo para estudo de caso 122
Figura 5.15 - Janelas de Menus 123
Figura 5.16 - Janela do projeto da aplicação 123
Figura 5.17 - Janela do inspetor de componentes 124
Figura 5.18 - Janela de criação de projeto 124
Figura 5.19 - Janela de visão textual 125
Figura 5.20 - Janela de escolha de arquivos 125
Figura 5.21 - Janelas para implementação 126
Figura 5.22 - Inserindo imagens no painel da página 126
Figura 5.23 - Inspetor de componentes com seus atributos e eventos 127
Figura 5.24 - Inserindo botões 128
Figura 5.25 – Salvando uma página 129
Figura 5.26 – Resultado final WYSIWYG 130

LISTA DE SIGLAS E ABREVIATURAS

Abert	- Associação Brasileira de Emissoras de Rádio e Televisão
ABNT	- Associação Brasileira de Normas Técnicas
ACAP/OCAP	- *Advanced Common Application Platform*/
ADSL	- *Assymmetric Digital Subscriber Line*
AM	- *Amplitude Modulation*
Anatel	- Agência Nacional de Telecomunicações
API	- *Application Programming Interface*
ARIB	- *Association of Radio Industries and Businesses*
ATSC	- *Advanced Television Systems Committee*
AVM	- *ActionScript Virtual Machine*
BML	- *Broadcast Markup Language*
CA	- *Conditional Access*
CASE	- *Computer Aided Software Engineering*
CDC	- *Connected Device Configuration*
CDK	- Kit de Desenvolvimento de Componentes
COBOL	- *COmmon Business Oriented Language*
COFDM	- *Coded Orthogonal Frequency Division Multiplex*
Com-TV	- Comissão Assessora para Assuntos de Televisão
CPU	- Central Processing Unit
CSS	- *Cascading Style Sheets*
DAVIC	- *Digital Audio Video Council*
DIGEB	- *Digital Broadcasting Expert Group*
DLL	- *Dynamic-link Library*
DOM	- *Document Object Model*
DSM-CC	- *Digital storage media command and control*
DTG, U.K.	- *Digital Terrestrial Group from United Kingdom* -
DVB	- *Digital Video Broadcasting*
ECMA	- *European Computer Manufacturers Association*

ELG - *European Launching Group*

EPG - *Eletronic Program Guide*

ETSI - *European Telecommunications Standards Institute*

FEC - *Foward Error Control*

Finep - Financiadora de Estudos e Projetos

FP - *Foundation Profile*

FUNTEL - Fundo para o Desenvolvimento Tecnológico das Telecomunicações

GEM - *Global Executable MHP*

GRINS - *GRaphical INterface for creating and playing SMIL documents*

HAN - *Home Area Network*

HAVi - *Home Audio / Video Interoperability*

HDTV -*High Density Television*

HTML - *HyperText Markup Language*

HTTP - *Hypertext Transfer Protocol*

IDE - Ambiente Integrado de Desenvolvimento

IDTV - *Interactive Digital Television Systems*

IHC - Interface-Homem Computador

IMK - *Fraunhofer Institute for Media Communication*

INRIA - *Institute National de Recherche en Informatique*

ISDB - *Integrated Services Digital Broadcasting*

ISDTV - *International System Digital TV*

JMF - *Java Media Framework*

JPEG - *Joint Photographic Experts Group*

Kbytes - *Quilobytes*

LDTV - *Enhanced Definition Television*

LDTV - *Low Definition Television*

Mbit/s - *Megabits* por segundo

Mbytes - *Megabytes*

MDB - *Multimedia Digital Broadcast*

MHEG - *Multimedia and Hypermedia Expert Group*

MHP - *Multimedia Home Platform*

MHz - Megahertz

MPEG - *Moving Picture Expert Groups*

NCL - *Nested Context Language*

NCM - *Nested Context Model*

NTSC - *National Television Systems Committee*

OCAP - *OpenCable Application Platform*

PAM - *Pulse Amplitude Modulation*

PBP - *Personal Basis Profile*

PC - *Personal Computer*

PDA - *Personal Digital Assistant*

PHP - *Hypertext Preprocessor*

PLC - *Power Line Communications*

PNG - *Portable Network Graphics*

POO - Programação Orientada a Objetos

PVR - *Personal Vídeo Recorder*

PUC-RJ - Pontifícia Universidade Católica do Estado do Rio de Janeiro

QAM - *Quadrature Amplitude Modulation*

QPSK - *Quadrature Phase-Shift Keying*

RAD - Rápido Desenvolvimento de Aplicações

RTOS - *Real Time Operational* System

SBTVD - Sistema Brasileiro de TV Digital

SDK - *Software Development Kit*

SDTV - *Standard Definition Television*

SECAM - *Séquentiel Couleur avec Mémoire*

SET - Sociedade Brasileira de Engenharia de Televisão e Telecomunicações

SGML - *Standard Generalized Markup Language*

SMIL - *Synchronized Multimedia Integration Language*

SMPTE - *Society of Motion Picture and Televison Engineers*

STB - *settop box*

TS - *Transport Stream*

TV - Televisão

TVDI - TV Digital Interativa

UFPB - Universidade Federal da Paraíba

UIT - União Internacional de Telecomunicações

VCL - Biblioteca de Componentes Visuais

VOD - *Vídeo On Demand*

VSB - *Vestigial Side-Band*

WAM - *Web Adaptation and Multimedia*

WiFi - *Wireless Fidelity*

WYSIWYG - *What You See Is What You Get*

XHTML - *eXtensible Hypertext Markup Language*

XML - *eXtensible Markup Language*

1 INTRODUÇÃO

Com a implantação progressiva do sistema brasileiro de TV Digital, tem aumentado a demanda por novos serviços para a televisão brasileira. O desenvolvimento de ferramentas que aumentem a produtividade no meio da TV Digital tem despertado o interesse pelas ferramentas de autoria. Estas ferramentas visam agilizar e facilitar a criação de aplicações para esse sistema, possibilitando ao autor abstrair o máximo possível da complexidade de programar.

Este trabalho tem como objetivo principal apresentar o LuaComp, uma ferramenta de autoria para aplicações híbridas (declarativas e procedurais), utilizando uma linguagem de *script* brasileira, Lua [46]. O LuaComp pode ser considerada como a primeira ferramenta de autoria que gera aplicações em Ginga-NCLua.

Este capítulo apresenta as motivações e objetivos desta dissertação e também descreve a estrutura deste documento, que é focado em uma pesquisa para desenvolvimento de ambiente de autoria para aplicações de TV Digital brasileira.

1.1 MOTIVAÇÃO

A TV Digital, além de ser uma consequência da convergência tecnológica, é uma evolução natural de um nicho que movimenta muito dinheiro e informação no mundo. O surgimento de mais um novo veículo de interatividade traz novas perspectivas de mercado tanto para o *hardware* como para o *software*. O desenvolvimento de *softwares* (aplicativos, *middleware*, sistema operacional) que propiciem uma melhor interatividade do telespectador pode aproximar a funcionalidade de uma TV à de um PC, atraindo novos interesses tanto da sociedade como da indústria.

Dentro do processo de construção do Sistema Brasileiro de Televisão Digital – SBTVD, houve a possibilidade de desenvolvimento de um *middleware* brasileiro (Ginga), o qual tem sido bastante demorado, dificultando o desenvolvimento de aplicações. O *middleware* brasileiro Ginga [39] com os dois módulos, o Ginga-NCL (declarativo) e o Ginga-J (procedural), com base no contexto atual da implantação da TV Digital no Brasil, tem fomentado uma incerteza quanto ao fato dos conversores (*settop boxes*) terem o Ginga

completo (que vem sofrendo atrasos) ou terem apenas o Ginga-NCL (o qual já está disponível).

No evento *Free Software* 2008, o professor Luiz Fernando Soares da PUC-RJ disse:

> Tem duas coisas no movimento pelo Java que eu, particularmente, não gosto. Nada contra o Java. O Java é bom. Do Java eu gosto. O que está por trás desse movimento em torno do Java é que preocupa, porque pode ser ruim. Por isso, vou entrar agora em uma campanha muito grande pelo desenvolvimento de aplicações NCL/Lua, até para contrabalançar um pouco este movimento que vem do Java [46].

Um aspecto negativo para o desenvolvimento atual dos aplicativos é o atraso que a falta de um *middleware* capaz de dar suporte a soluções baseadas em Java está provocando no processo da interatividade. *"Podemos, sim fazer muita coisa só com NCL/Lua. Todo mundo sabe disso, mas parece que não sabe"*, afirma o professor Luiz Fernando Soares [46]. Dessa forma, a demanda por novas soluções procedurais, que substituam o Java, torna-se bastante clara e é uma das motivações desta dissertação.

Apesar deste processo aparentemente moroso de implantação, é importante ressaltar que as normas da Associação Brasileira de Normas Técnicas - ABNT - para o sistema brasileiro de TV Digital, tanto para o *hardware* como o *middleware*, já estão publicadas e disponíveis de forma gratuita, com exceção da norma sobre segurança (ABNT NBR 15605), cuja elaboração ainda não foi concluída, dificultando o desenvolvimento de aplicações que se fundamentam na segurança como: *T-Bank, T-Shopping* e outras aplicações de TV Digital interativa.

Apesar da demora na disponibilização do *middleware* e de receptores digitais para interatividade, fato que também ocorreu nos países precursores [39], o mercado de *hardware* e *software* tem impulsionado e valorizado ainda mais o esforço em pesquisa de modelos de aplicação que melhor se adaptem ao contexto nacional [36]. Apesar de o *middleware* brasileiro Ginga-J não estar disponível nos terminais de acesso, já existem ferramentas para teste do Ginga-NCL como emuladores e até ferramentas de autoria como o *Composer*.

Segundo Guimarães [1], apesar da disponibilização do *Composer* para autoria de aplicações declarativas com sincronismo de mídias, foi deixada uma lacuna para o desenvolvimento de rotinas para entrada e saída de dados. O LuaComp se dispõe a complementar esta lacuna. O Ginga-NCL puro, por ter uma estrutura muito parecida com o HTML, dificulta bastante o uso de variáveis e das estruturas de controle que existem nas linguagens procedurais.

Ao lado dos aspectos citados, a indefinição de modelos de *hardware* com relação à sua capacidade de processamento e armazenamento de aplicações tem impulsionado as pesquisas de modelos econômicos de aplicação. Algumas linguagens procedurais sistêmicas como o Java poderão ser preteridas por outras de *script* como o Lua ou HTML, devido ao seu alto consumo de memória principal.

Ao lado dessa indefinição, os parâmetros mínimos dos receptores digitais apresentados na norma ABNT 15604:2007 [60], como memória de 2 Mega bytes, limitam bastante as possibilidades funcionais e visuais das aplicações. Explorar cada vez mais o *software* em prol de um melhor rendimento e menor custo de processamento tem sido um esforço interminável. Ierusalimschy [61] lembra que a linguagem Lua tem se destacado mundialmente devido à sua leveza e consumo de memória, o que é ratificado por ser uma das mais usadas em *softwares* de jogos, que exigem boa memória e um melhor processamento de CPU.

Apesar da demora na comercialização dos receptores digitais com o Ginga, a demanda por aplicações com interatividade plena é iminente. Com relação às aplicações que demandam apenas sincronismo de mídias, o Ginga-NCL já atende perfeitamente aos requisitos funcionais. Com relação às aplicações com interatividade plena ou intermitente, o Ginga-J, com sua demora na liberação e também falta de norma ABNT, tem gerado descontentamento na comunidade e assinalado para a alternativa da linguagem Lua.

Por fim, uma motivação adicional deste trabalho está no uso de uma linguagem nacional, livre e com funcionalidades semelhantes as que o Ginga-J promete atender, para o desenvolvimento de uma ferramenta de autoria.

1.2 OBJETIVOS

O objetivo principal deste trabalho é apresentar um protótipo de ferramenta de autoria, LuaComp, para aplicações de TV Digital interativa utilizando linguagem Lua. Esta ferramenta utiliza a linguagem Lua como solução procedural para aplicações híbridas, ou seja, com código declarativo e procedural (imperativo). Aplicações híbridas são aplicações que misturam códigos declarativos e procedurais com recursos gráficos para entrada e saída de dados e sincronismo de mídias. Soares et al. [46] dizem que uma solução híbrida pode consistir em adicionar à linguagem declarativa algum suporte imperativo, assim, o autor da

aplicação usa a forma declarativa sempre que possível e lança mão da forma imperativa somente quando necessário.

Esta dissertação também objetiva disponibilizar um material de referência para consulta no desenvolvimento de aplicações ou de novas ferramentas de autoria para TV Digital interativa. Estes conceitos são fundamentais para a formação de desenvolvedores, considerando que o entendimento das limitações e peculiaridades das linguagens e das especialidades dos ambientes de TVD não deve ser ignorado no desenvolvimento de aplicações e ferramentas para a TV Digital.

1.3 METODOLOGIA

A metodologia adotada para este trabalho parte da descrição das principais características do sistema de TV Digital interativa e do estudo dos *middlewares*. Os tipos de linguagens declarativas e procedurais são também abordados apresentando suas vantagens e desvantagens para o desenvolvimento e aplicação. Os conceitos de usabilidade e os perfis dos tipos de aplicações são também apresentados como base para o desenvolvimento das aplicações e das ferramentas gráficas. Um estudo sobre as principais ferramentas de autoria é realizado.

Levando em consideração esses estudos, esta dissertação apresenta uma ferramenta de autoria, LuaComp, que gera aplicações interativas para TV Digital com modelo híbrido (declarativo e procedural) tomando por base a abordagem de orientação a objetos e a verificação prévia dos recursos e funcionalidades presentes em diferentes ferramentas de autoria.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação é dividida em seis capítulos.

O capítulo 2 apresenta a TV Digital interativa de forma a introduzir o sistema destacando as principais características do sistema de TV Digital, sua arquitetura e a importância da interatividade para a composição das aplicações. Esse capítulo apresenta o *middleware* brasileiro Ginga. Ademais, ele também aborda os tipos de aplicações e os padrões de usabilidade como importante referência para a construção de aplicações interativas.

O capítulo 3 disponibiliza os trabalhos relacionados a ferramentas de autoria, mostrando o desenvolvimento das ferramentas CASE. Esse capítulo também procura destacar a influência dos tipos de ambientes das aplicações como *Desktop* e *Web* até chegar às ferramentas de autoria para TV Digital. Há destaque para as ferramentas de autoria que tratam do sincronismo de mídia e da interatividade para entrada de saída de dados.

O capítulo 4 apresenta a linguagem Lua, o Ginga-NCLua e destaca o *framework* LuaOnTV desenvolvido pelo laboratório de TV Digital da UnB.

No capítulo 5 a interface do LuaComp e suas funcionalidades são apresentadas e discutidas e comparadas. Um estudo de caso também é abordado. E, por fim, o capítulo 6 apresenta conclusões da dissertação, destacando a importância da ferramenta para o atual contexto do SBTVD e indicando possíveis trabalhos futuros.

2 TV DIGITAL INTERATIVA

2.1 INTRODUÇÃO

Segundo Moreno [38],

> [...] os sistemas de TV Digital podem ser definidos, de forma resumida, como um conjunto de especificações que determinam as técnicas de codificação digital para transmitir o conteúdo de áudio, vídeo e dados, das emissoras (ou provedores de conteúdo) aos terminais de acesso dos telespectadores, e um conjunto de facilidades que dão suporte ao desenvolvimento de aplicações interativas.

A TV Digital também apresenta um conjunto de recursos que dão suporte ao desenvolvimento de aplicações interativas. Além da melhoria na qualidade da imagem, conforme a Figura 2.1, a possibilidade de envio e recebimento de dados e/ou aplicação é a grande novidade da TV Digital. A capacidade de transmissão de dados junto com vídeo e áudio, vinculados ou não à programação da operadora de sinal, possibilita o desenvolvimento de novos serviços e aplicações digitais. É a TV Digital interativa.

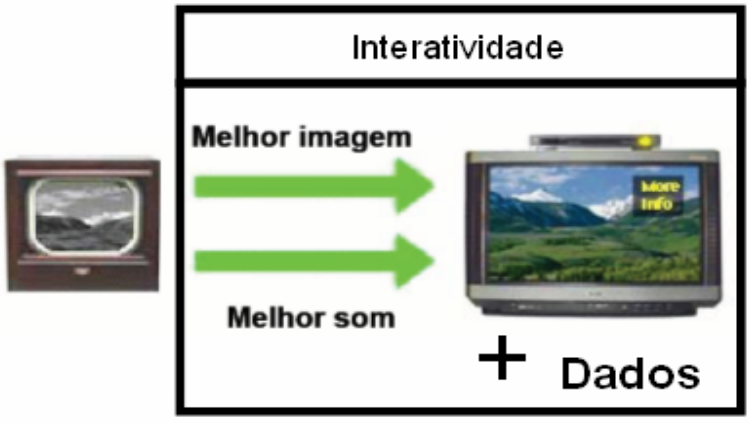


Figura 2.1 - A nova televisão
FONTE: Disponível em: www.ginga.org.br. Acesso em: 2 mar. 2009. [34]

O objetivo deste capítulo é destacar as principais características da TV Digital que delimitam o escopo para o desenvolvimento de aplicações interativas para esse sistema. Conhecer o limite para o tamanho das aplicações, a forma como as aplicações são enviadas, a importância da interatividade e os padrões de usabilidade, ajudam o desenvolvedor de aplicações para computadores a compreender as limitações e os principais detalhes de uma

aplicação para TV Digital. A história dessa TV e da implantação dos *middlewares* no mundo também ajudará a entender melhor os problemas que o sistema brasileiro de TV Digital está encontrando.

Este capítulo explorará as características da TV Digital, passando por suas características básicas, pelos sistemas de TV Digital no mundo, pela história dessa TV, pela arquitetura do sistema, pela interatividade e usabilidade, por exemplos de tipos de aplicações para TV Digital interativa e, finalmente, a evolução do *middleware* no mundo.

2.2 TV DIGITAL

A TV Digital apresenta uma melhor qualidade de serviços. Ela traz inúmeras vantagens com relação à TV analógica como: imagem com qualidade DVD ou superior; som com qualidade CD ou superior; transmissão em diversos idiomas e legendas; transmissão de dados junto com as aplicações; adaptação à área de cobertura e características locais; e serviços adicionais de telecomunicação como: comércio eletrônico; maior número de canais; resolução de problemas de interferência co-canal; maior número de canais; mais programação por emissora; *Personal Vídeo Recorder* – PVR – que permite gravação de programas pelo telespectador; personalização de conteúdo; e, principalmente, serviços interativos. A largura de banda é a mesma que a da TV analógica, ou seja, 6 MHz.

Lopes [64] afirma que o sistema de televisão digital apresenta como novidade a transmissão de sinal audiovisual digital em vez de analógico. Conforme a Figura 2.2, o sistema analógico apresenta a possibilidade de produzir fantasmas na imagem e, no sistema digital, para melhorar a robustez do sinal, são adicionados códigos corretores de erro aos *streams* ou feixes de *bits*.

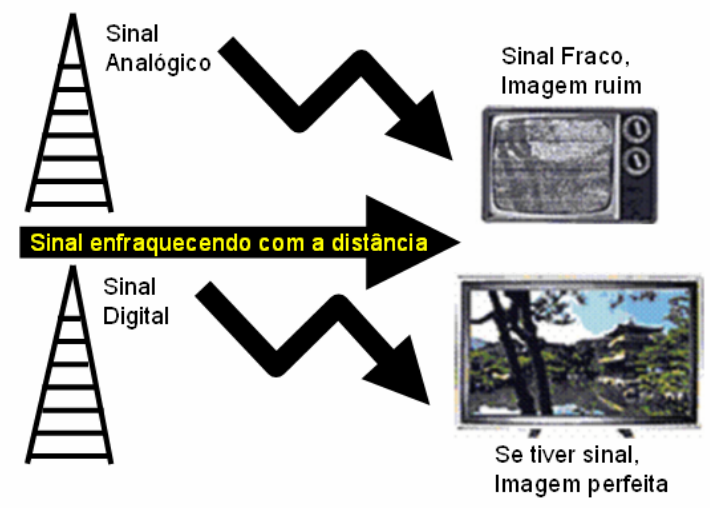


Figura 2.2 – Sinal analógico X Sinal digital
FONTE: Disponível em: www.ginga.org.br. Acesso em: 2 mar. 2009. [34]

Segundo a INATEL [63], a transmissão digital, todos os sistemas mundiais possuem uma estrutura similar e padronizada pela União Internacional de Telecomunicações – UIT. Os sinais de áudio e vídeo são digitalizados e multiplexados para depois formarem um fluxo de *bits* de programa. Um ou mais fluxos de programa são multiplexados para formar um feixe de transporte chamado *Transport Stream*. Essa multiplexação é feita no padrão MPEG-2. O feixe de transporte é colocado num codificador de canal e pode variar de 4 a 24 Mbit/s.

Mesmo com o grande esforço no desenvolvimento de técnicas de transmissão e codificação de sinais, a qualidade da imagem sempre foi um importante objetivo. Schwalb [35] comenta que pesquisas no desenvolvimento de novo formato foram o grande ponto inicial da televisão digital. O *High Definition TV* – HDTV - foi um importante marco tecnológico para impulsionar a grande demanda mundial por serviços digitais. O HDTV apresentava até o dobro da resolução espacial da televisão analógica com resoluções de 1080 ou 720 linhas horizontais, em formato de tela 16:9. Em 1981 a *Society of Motion Picture and Television Engineers* – SMPTE –, fez a primeira apresentação do HDTV.

De acordo com Schwalb [35], independente da evolução da qualidade dos vídeos e áudio, a principal novidade que diferencia a televisão analógica da digital é a interatividade. Este novo recurso tem uma influência muito grande da informática e principalmente da Internet. A interatividade aliada a uma excelente qualidade de imagem e recepção móvel transforma a TV Digital em TV Digital Interativa – TVDI. A interatividade tornou-se um recurso essencial para uma televisão digital de qualidade.

O recurso da TV Digital de transmitir dados possibilita aos terminais de acesso (receptores digitais ou *settop box)* disponibilizar diversos tipos de aplicações. Conforme a Figura 2.3, um *middleware* intermedia o acesso das aplicações aos recursos do *settop box*.

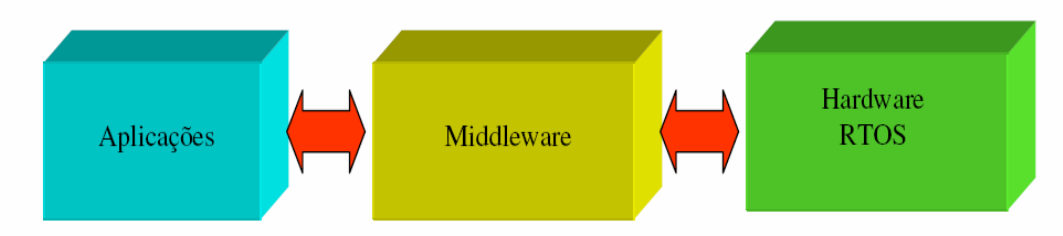


Figura 2.3 – *Middleware*

Não muito diferente deste contexto, a TV Digital, com o desenvolvimento de seus receptores e equipamentos, e a valorização de uma melhor interatividade, reforça a importância das aplicações e, consequentemente, o papel do *middleware* no desenvolvimento desta tecnologia. O conhecimento da TV Digital, como um meio complexo e com limitações de *hardware* e *software*, pode ajudar ao desenvolvedor de aplicações interativas a não projetar produtos inviáveis.

2.3 SISTEMAS MUNDIAIS DE TV DIGITAL

As diferentes redes de transmissão: terrestre, cabo, via satélite e via microondas influenciam bastante o modelo das TVs digitais no mundo. Para Morris e Smith-Chaigneau [36], as diferentes características destas redes foram fundamentais para a escolha do tipo de modulação. As modulações de sinal não podem ser as mesmas nestas redes. O sistema terrestre pode adotar o COFDM ou o 8-VSB. O sistema de TV a cabo adota a modulação QAM. A modulação adotada pelo sistema satélite é QPSK. A modulação é o último estágio da camada de transmissão. A modulação converte o sinal digital em sinal de radiofrequência. A modulação é utilizada para otimizar a transmissão dos dados nas redes carregando um máximo de dados em um mínimo de banda. Observe na Figura 2.4 os principais sistemas de TV Digital mundial: o americano (ATSC), europeu (DVB) e o japonês (ISDB). Segundo Tomé [78], os grandes centros econômicos como EUA, Europa e Japão adotaram sistemas de transmissão digital diferentes.

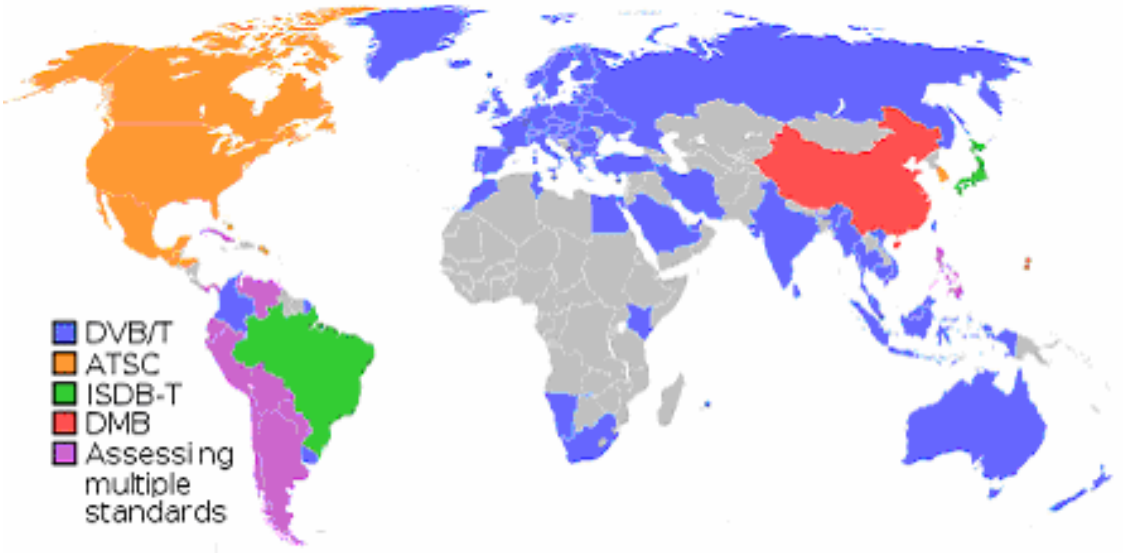


Figura 2.4 - Sistemas de televisão digital *broadcast*

A Tabela 2.1 apresenta a arquitetura da TV Digital divida em blocos ou camadas como: transmissão, transporte, compressão, *middleware* e aplicação. Essa tabela traz todos os elementos das principais TV digitais no mundo.

Aplicações	**APP1**	**APP2**	**APP3**	**APPn**
Middleware	**DASE/OCAP/ACAP**	**MHP**	**ARIB**	**GINGA**
Compressão - Áudio	**MPEG2 BC MPEG2 AAC DOLBY AC3**			
- Vídeo	**MPEG2-SDTV**		**MPEG2-HDTV**	
Transporte	**MPEG2**			
Transmissão / Modulação	**8-VSB**		**COFDM**	

Tabela 2.1 - Padrões e camadas da TV Digital
FONTE: Adaptado de FERNANDES et al., 2004. [37]

Com base na pesquisa em diversas referências, a Tabela 2.2 também apresenta os sistemas mundiais de TV Digital com os recursos que realizam um tratamento melhor, pois esses recursos são também implementados por todos os demais, com um grau de eficiência diferente.

SISTEMAS		MODULAÇÃO TERRESTRE	HDTV	MOBILIDADE	MULTI-PROGRAMAÇÃO	INTERA-TIVIDADE	ESPALHA-MENTO
Europeu	DVB	COFDM	X		X		Espacial
Americano	ATSC	VSB					
Japonês	ISDB	COFDM		X			Temporal
Brasileiro	ISDB	COFDM				X	

Tabela 2.2 - Melhores características

Com todas estas diferenças de focos dos sistemas mundiais de TV Digital descritas, conclui-se que as aplicações e *middlewares* devem estar de acordo com as principais características destacadas na Tabela 2.2. Uma ferramenta de autoria pode ter seus recursos limitados pelo *middleware* e/ou pelas normas regulatórias do sistema de TV Digital.

2.4 SISTEMA BRASILEIRO DE TV DIGITAL

A TV Digital no Brasil está sendo introduzida de uma forma não muito diferente da realizada nos outros países desenvolvidos. No Brasil, o processo está sendo coordenado pelo governo brasileiro e implantado junto com o desenvolvimento dos equipamentos e *softwares*. A TV Digital brasileira estará apta a disponibilizar serviços de alta qualidade de imagem junto com mobilidade, interatividade e multiprogramação. É importante ressaltar que é bastante salutar o fato do Brasil estar implantando o sistema de TV Digital depois de ter sido testado na maioria dos grandes centros. O sistema brasileiro poderá ser classificado como um dos mais modernos e flexíveis do mundo. A seguir, será apresentado todo o processo de implantação.

Segundo Denise Lopes [64], em 1991, o ministério das Comunicações do Brasil estabeleceu a Comissão Assessora para Assuntos de Televisão - Com-TV -, encarregada de estudar e analisar a TV de alta definição que estava sendo desenvolvida no mundo. Em 1994, a Associação Brasileira de Emissoras de Rádio e Televisão - ABERT - e a Sociedade Brasileira de Engenharia de Televisão e Telecomunicações - SET -, uniram-se formando o Grupo Técnico ABERT/SET de TV Digital. Este grupo era composto por profissionais que atuavam diretamente nas áreas de Televisão, Telecomunicações, Rádio e Multimídia.

A partir de 1998, começaram os testes com os modelos europeu e norte-americano, através de regulamentação da Agência Nacional de Telecomunicações - Anatel. Um ano depois, 1999, o sistema japonês passou a ser testado. Foi nomeada uma comissão que ficou

responsável pelos testes para que se pudesse escolher o melhor modelo a ser implantado no Brasil.

Em 2003, intensificaram-se as discussões em torno da TV Digital no Brasil, culminando com o Decreto nº 4.901, de 26 de novembro [40], que instituiu oficialmente o Sistema Brasileiro de Televisão Digital – SBTVD. Com este Decreto intensificaram-se as discussões e especulações em torno do padrão de TV Digital a ser adotado no Brasil.

Denise Lopes [64] disse que os resultados dos testes e as vantagens oferecidas pelo sistema japonês foram fundamentais para a escolha do novo padrão de TV Digital a ser adotado no Brasil. Houve muita discussão em torno desta escolha. Os europeus alegaram que a escolha brasileira pelo padrão japonês foi a mais cara, pois somente o Japão adota o sistema ISDB. Já os norte-americanos defenderam que a escolha do ATSC proporcionaria um aumento nas exportações de televisores, visto que o mercado americano se abriria aos televisores brasileiros.

Em junho de 2006, outro decreto foi assinado instituindo o padrão japonês como modelo a ser implantado no país. Segundo a Agência de Notícias do Planalto, em 26 de outubro daquele ano, citado por Denise Lopes [64], *"o cronograma de implantação prevê que até dezembro de 2009 o novo modelo vai estar funcionando em todas as capitais e, até dezembro de 2013 em todos os municípios brasileiros. O modelo analógico será tirado de funcionamento em junho de 2016"*.

Como forma de acreditar e valorizar a capacidade do brasileiro em desenvolver seu próprio padrão de TV Digital, a INATEL liderou um consórcio que integrava os subsistemas desenvolvidos pelas universidades e centros de pesquisa do país.

> "Um transmissor desenvolvido pelo consórcio MI–SBTVD foi alimentado por um vídeo com informações interativas desenvolvido pela UFSC. O vídeo foi recebido pelo protótipo do receptor do MI-SBTVD e entregue ao terminal de acesso da USP, onde se instalava o *Middleware* da UFPB. Através do controle remoto um telespectador pôde interagir com o programa de forma intuitiva [...] O que torna essa conquista ainda mais significativa é o fato de que todos esses resultados foram obtidos com menos de 15 meses de trabalho e com um orçamento relativamente muito modesto, se comparado aos gastos realizados pelos EUA, Europa ou Japão no desenvolvimento de seus respectivos padrões"[63].

O projeto de desenvolvimento do *middleware* brasileiro tem também a preocupação de ser aderente à GEM, possibilitando funcionar em *middlewares* de outros países, com algumas restrições, através de conversões automáticas. Logicamente as aplicações que forem

desenvolvidas exclusivamente para o mercado nacional terão mais facilidades em se beneficiar de todas as facilidades apresentadas pela linguagem NCL [16].

Com relação ao *middleware* brasileiro, o Ginga, não diferentemente dos outros *middlewares* já implantados no mundo, oferece solução tanto declarativa como procedural. As diversas formas de modelar a aplicação, seja declarativa, seja procedural, seja híbrida, apresentam resultados deferentes e que podem ser relevantes na concepção do produto.

O sistema brasileiro de TV Digital vem sendo implantado de forma paulatina, mas a demora na liberação do Ginga-J vem atrasando o desenvolvimento de aplicações para o ISDTV. Para os receptores digitais serem completos, é necessário que as especificações do Ginga-NCL e o Ginga-J sejam liberadas juntas. O Ginga-J foi programado pela Sun para ser liberado em meados de setembro de 2008 [39]. O fato é que o cronograma para o Ginga completo já começou a atrasar. A primeira previsão era a de que a norma ABNT do Ginga-J com as APIs JavaDTV fosse para junho de 2009. Com todos estes percalços, passou a existir uma data mágica para ter o Ginga completo nos conversores, apesar de ser programado para junho de 2009.

Hoje, com este imbróglio, o mercado e o próprio pai do Ginga já passam a defender o Ginga-NCLua como capaz de fazer muito no quesito interatividade. O medo é que no lançamento do Ginga faltem aplicações e um modelo de negócio que torne a interatividade atraente para a maioria dos *players* do ecossistema.

No site Convergência Digital, a Casa Civil também afirma que deixar a decisão na mão dos grandes fabricantes pode continuar atrasando a chegada da interatividade. Hoje existem defensores da ação do governo com recursos do Fundo para o Desenvolvimento Tecnológico das Telecomunicações - Funttel - e da Financiadora de Estudos e Projetos - Finep – para massificar a implantação do Ginga por meio de pequenas produtoras de conversores. Segundo Costa:

> [...] quando a TV Digital foi lançada, em dezembro do ano passado, o conversor custava R$ 1400,00 - preço elevado e fora da realidade nacional. Hoje, segundo o ministro, já há conversores vendidos por R$ 240,00, considerado por Costa, um preço acessível. Para o ministro, no entanto, não há um engajamento da indústria produtora em popularizar a tecnologia [59].

Com tudo isso, conclui-se que a demora na resolução do padrão a ser implantado no Brasil está servindo para análise da resposta do mercado, das operadoras e da população. Para Castro et al. [44], estas respostas estão servindo para elaborar um conjunto de objetivos que

atenda à realidade brasileira. A questão principal percorre não apenas a parte tecnológica, mas especialmente o social. A implantação da TV Digital não poderá afetar apenas a economia do país, mas a educação, a distribuição de renda e a inclusão digital. O ISDTV é um grande passo na história da televisão e já mostra o potencial dos pesquisadores brasileiros nas áreas de tecnologia, informação, telecomunicações e muitas outras áreas, possibilitando a melhoria das condições socioeconômicas brasileiras.

2.5 LINGUAGENS NOS PADRÕES DE TV DIGITAL

Os padrões atuais de TV Digital oferecem linguagens bem heterogêneas para a especificação da interatividade e do sincronismo nas aplicações. As aplicações para TV Digital interativa podem ser especificadas em dois grupos: declarativas e procedurais ou imperativas.

A maioria dos *middlewares* implantados no mundo, desde o MHEG em 1997, vem disponibilizando as linguagens declarativas e procedurais para serem escolhidas como alternativas solitárias ou híbridas para implementação dos códigos das aplicações interativas para a TV Digital. Apresentar as características das linguagens declarativas e procedurais ou imperativas pode facilitar as escolhas destas linguagens para o desenvolvimento das aplicações interativas.

No ambiente de TV Digital, é possível ter aplicações com vários tipos de comportamento: aplicação somente para organização de exibição de mídias que precisam ser organizadas no espaço da tela do aparelho televisor e sincronizadas temporalmente com outras mídias, e outra, que se assemelha aos programas de computador, com entrada e saída de dados gerados pelo usuário. O primeiro tipo se refere a aplicações declarativas e o segundo as imperativas ou procedurais. As aplicações de TV Digital são classificadas como modelos hipermídia e interativos.

Aplicações para TV Digital são casos particulares de aplicações multimídia/hipermídia porque podem manipular diversos tipos de objetos de mídia como imagem, texto, vídeo e áudio, permitindo a interatividade e o relacionamento sincronizado entre eles. Este tipo de sincronismo pode ser espacial e temporal. Segundo Robert Waker [6], aplicações hipermídia podem ser definidas como aplicações que misturam os conceitos de multimídia e hipertexto [6]. O hipertexto é uma ferramenta que permite a navegação e,

consequentemente, a interatividade não linear. Um documento hipermídia possui nós como todo documento hipertexto para acesso a mídias.

Atualmente, existem linguagens declarativas que trabalham com sincronismo como a - *Synchronized Multimedia Integration Language* - SMIL (SMIL, 2009) e a *Nested Context Language* - NCL. A SMIL é uma linguagem bastante intuitiva para o desenvolvimento de apresentações com sincronismo temporal de mídias. Segundo Soares et al. [28], a NCL é uma linguagem para autoria de documentos hipermídia baseada no modelo conceitual *Nested Context Model* – NCM. No modelo NCM um documento hipermídia é representado por um nó de composição, podendo este conter um conjunto de nós, que podem ser objetos de mídia ou outros nós de composição, recursivamente, e ainda elos relacionando esses nós.

As linguagens que permitem a implementação de funções não lineares como navegação entre páginas são declarativas e surgiram como linguagens de marcação. Outras linguagens, as imperativas ou procedurais, são mais rígidas, pois possuem estruturas de controle que predefinem o comportamento do usuário na aplicação. Na área da TV Digital tem-se como exemplo de uma linguagem imperativa o Java e como declarativa o XHTML. A utilização de linguagens procedurais ou imperativas é muito mais complexa que a utilização das declarativas. Outra diferença é que as linguagens imperativas foram feitas para programadores e as declarativas podem ser utilizadas por profissionais com pouquíssimo conhecimento de programação ou de construção de algoritmos, como *web designers* e curiosos da Internet.

A possibilidade de manipulação de vídeo, áudio, imagem e texto as tornam aplicações muitas vezes mais robustas. Apesar da menor complexidade no uso dos códigos das linguagens declarativas, os recursos utilizados podem ser mais complexos que os códigos procedurais. Guimarães [1] afirma que as aplicações desenvolvidas em XHTML, a princípio, favorecem a autoria ao permitir a especificação das aplicações através de marcações *eXtensible Markup Language* - XML. Apesar de toda esta capacidade, as linguagens de marcação são restritas apenas à formatação da apresentação e ao relacionamento entre os documentos, necessitando de outras linguagens para suprir suas deficiências quanto à interatividade e ao sincronismo. Aplicações mais elaboradas e que necessitam de sincronismo e outros tipos de interatividade, necessitam de adicionar outras linguagens procedurais sistêmicas ou de *script*. Essas linguagens suprem a necessidade de uso de estruturas de controle, variáveis e procedimentos que podem, eventualmente, acessar funções oferecidas pelo *middleware*. Essas linguagens também são imperativas.

Soares [25] diz que a principal linguagem procedural para implementação de aplicações para TV Digital é o Java. Produzir uma aplicação criada com base em um conjunto de classes oferecidas pelo *middleware*, que já esconde muito da complexidade, com conhecimento de programação orientada a objeto, não esconde a necessidade que o autor conheça as funcionalidades das classes do *middleware* e que possua conhecimentos de programação e orientação a objeto, como encapsulamento, herança, polimorfismo, etc.

As linguagens imperativas ou procedurais podem ser divididas em sistêmicas e de *scripts*. As linguagens sistêmicas produzem códigos pré-compilados e linkeditados, e com formatos de arquivos específicos da linguagem. As linguagens de *script* são escritas em arquivos texto e são reconhecidas através de sua extensão, exemplo de arquivos *Javascript*, *Python* e Lua. Elas são executadas por outra máquina de execução que as reconhecem como *ActionScript*. As linguagens de *scripts*, além de terem sido feitas para complementar as limitações das linguagens declarativas, exigem menos memória, como na Figura 2.5 a seguir.

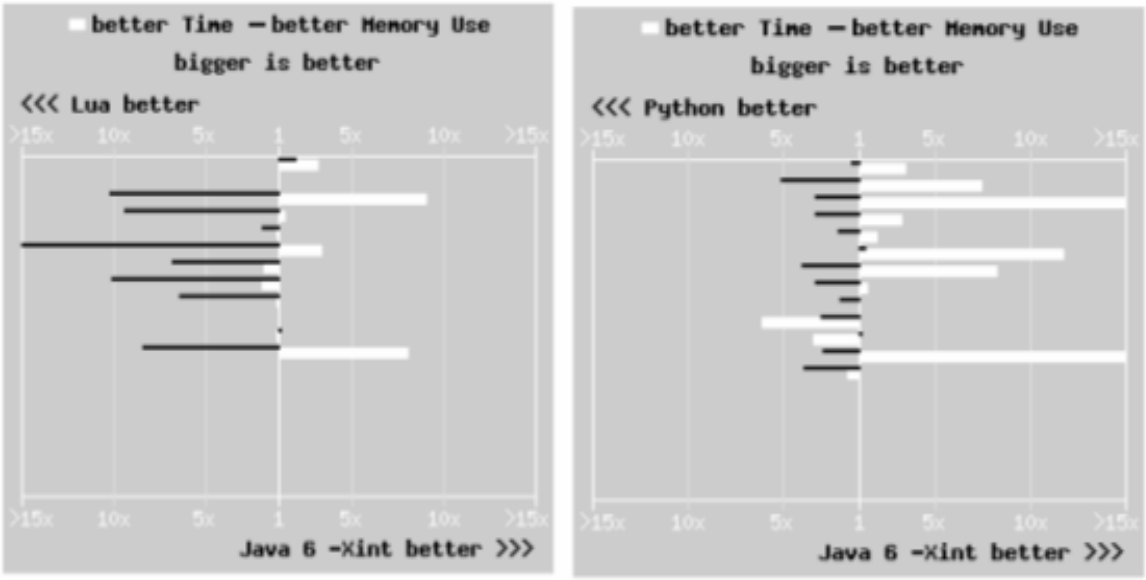


Figura 2.5 - *Benchmark*
FONTE: THE COMPUTER LANGUAGE BENCHMARKS GAME, 2008. [49]

O uso de ferramentas de autoria para gerar aplicações tanto declarativas como procedurais ou híbridas se faz, fundamentalmente, para ganho de tempo em produção e muito menos para possibilitar a implementação por leigos. A TV Digital, com sua complexidade e com suas aplicações nada menos difíceis de serem produzidas, não exime os produtores de

conteúdo de aplicações hipermídia de um conhecimento de programação. Falar que estas ferramentas substituirão programadores é repetir uma conversa que ainda não aconteceu.

Com esta complexidade toda, tanto das linguagens procedurais como das declarativas, se faz necessário o uso de ferramentas para autoria gráfica. As ferramentas de autoria, na maioria das vezes, devem responder e suportar os recursos oferecidos por estas linguagens e também criar abstrações para o desenvolvimento de aplicações interativas. Algumas ferramentas de autoria, que trabalham melhor com sincronismo temporal e espacial, são baseadas em linguagens declarativas e em contrapartida estas aplicações são limitadas quanto aos recursos de programação, por isso a valorização das soluções híbridas.

2.6 *MIDDLEWARE*

Não muito diferente desse contexto, a TV Digital, com o desenvolvimento de seus receptores, equipamentos e a valorização de uma melhor interatividade, reforça a importância das aplicações e, consequentemente, o papel do *middleware* no desenvolvimento dessa tecnologia. A possibilidade de um aparelho anexo à TV, como o receptor Digital *settop box*, carregar um *software* que explore todos os recursos de *hardware*, e tornar o telespectador mais ativo, possibilitando uma interatividade plena, reforça a importância do *middleware*. Independente do tipo de aplicação, o *middleware* deve fornecer mecanismos de abstração dos recursos e dos serviços de comunicação do receptor digital.

Para Márcio Moreno [38], o mundo da computação foi dividido inicialmente em *software* e *hardware*. Com o aumento das necessidades de se explorar cada vez mais o *hardware* sem a limitação de um sistema operacional, surgiu uma nova categoria chamada *middleware*. O *middleware* também não deixa de ser um *software*. O *middleware* é um neologismo, palavra ou expressão de criação recente, criado para designar camadas de *software* com novas funcionalidades, e como mediador entre o sistema operacional e as aplicações. O surgimento dos *middlewares* pode ser encarado como um importante paradigma para o desenvolvimento das aplicações. Observe na Figura 2.6 que o *middleware* é uma camada entre a aplicação e o *Real Time Operational* System – RTOS –, sistema operacional do receptor digital. O *middleware* é um *software* capaz de interpretar os aplicativos e traduzi-los na linguagem da plataforma em que reside, ou seja, fala como o sistema operacional.

Figura 2.6 - Arquitetura de *middleware*
FONTE: Adaptado de CALDER et al., 2000, p. 8 [42]

.

As aplicações para TV Digital podem ser transmitidas através dos canais de televisão para os receptores digitais. Fluxos de áudio e vídeo do canal também são enviados, porém de outra forma. O *middleware* tem também a função de perceber a chegada destas aplicações e apresentá-las ao telespectador no momento planejado.

Os *middlewares*, como os padrões mundiais de sistemas de TV Digital, também possuem diferentes características. Os *middlewares* podem ser divididos em proprietários e livres, conforme a Figura 2.7, quanto à forma de comercialização, e procedurais e declarativos, quanto à forma de programação, sem falar que os tipos de redes também o influenciam. Os principais *middlewares* proprietários são o *OpenTV*, NDS, Canal+, *PowerTV* e Microsoft. Os livres e mais conhecidos são o MHEG5, MHP, DASE, ACAP, e OCAP BML. Hoje, os principais *middlewares* "livres" ou não proprietários são o MHP (europeu), DASE/ACAP e OCAP (americano), o ARIB (japonês) e agora o Ginga (brasileiro).

Os principais *middlewares* no mundo são o MHP (europeu), ACAP/OCAP (americano) e o ARIB (japonês). Todos esses *softwares* começaram seu desenvolvimento de forma independente, mas hoje convergem para um modelo mundial, o GEM (*Globally Executable MHP*). Este padrão, apesar de fortalecer todos os *middlewares* no mercado, também comprova a importância de se considerar todas as experiências dos diversos *middlewares* em contextos de mercados diferentes. O Ginga, *middleware* brasileiro, da mesma forma que os outros, se divide em parte declarativa, Ginga-NCL e procedural, Ginga-J.

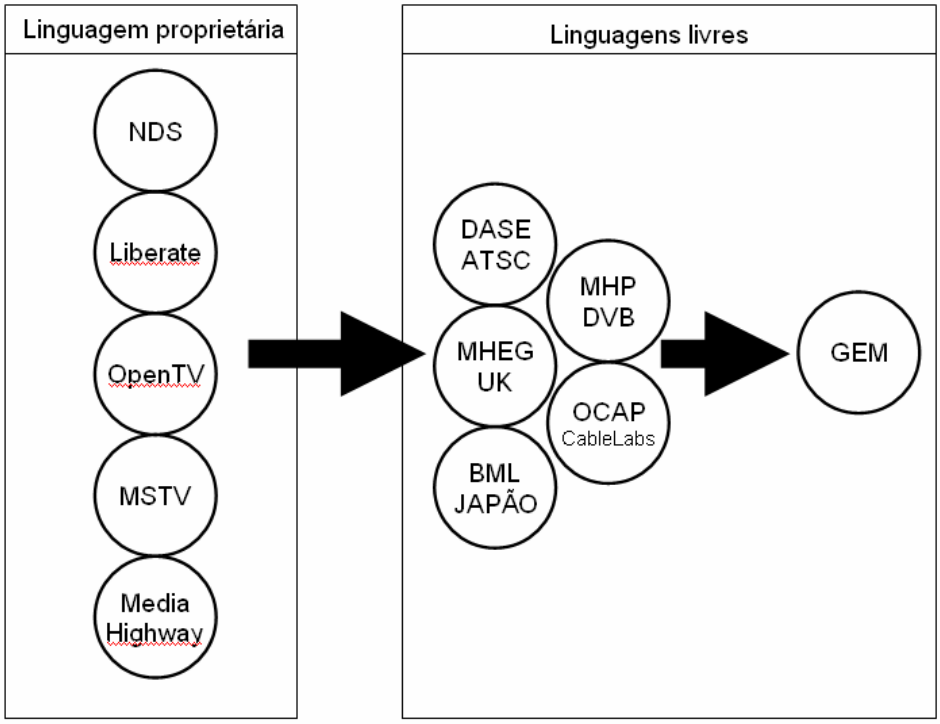


Figura 2.7 – Evolução e convergência dos *middlewares* até o GEM
FONTE: COELHO, 2005, p. 6. [67]

O GEM considera várias interoperabilidades entre os padrões como: interoperabilidade entre o DASE e OCAP, sistemas de transmissões, mudanças de modulação, mecanismos de distribuição, *Conditional Access* - CA - e operadores de redes. O GEM tornou-se o principal framework e foi publicado em 2003 com uma lista de especificações para MHP, DASE, OCAP, ACAP e ARIB B.23. Foi institucionalizado a cobrança de royalties para utilização do selo MHP e OCAP em STBs a partir de 2005. As aplicações MHP são identificadas como certificação quando apresenta a logo MHP. O GEM é a tentativa de harmonização dos *middlewares* existentes. Observe na Figura 2.8 que o GEM é compreendido com um subconjunto do MHP com funções gerais que devem ser implementadas por qualquer *middleware*.

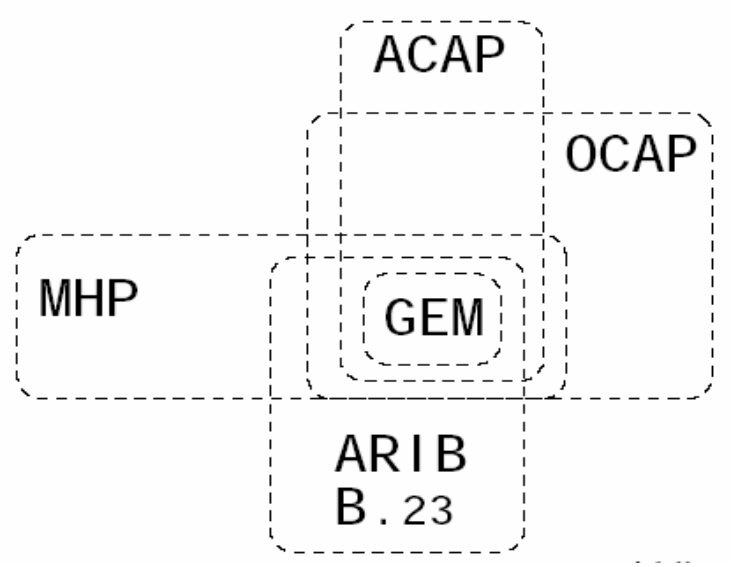


Figura 2.8 - GEM como subconjunto dos outros *middlewares*
FONTE: COELHO, 2005, p. 6. [67]

No caso do Ginga, como a proposta brasileira é de incentivo a popularização da TV Digital [76], a cobrança de *royalities* pelo uso das APIs GEM, estão inviabilizando o GINGA-J e consequentemente o GINGA. Com isso, o Ginga-NCLua tem se tornado uma solução e um nicho para o desenvolvimento de uma solução brasileira totalmente livre, pois Lua é uma linguagem brasileira e livre.

2.7 *MIDDLEWARE* GINGA

Como herança de todo o esforço da comunidade acadêmica brasileira desde 1998, e também como fruto de desenvolvimento de uma ferramenta de hipermídia iniciado há mais de 15 anos pela PUC-RJ, restou para o Brasil desenvolver seu próprio *middleware*, O Ginga.

O Ginga possibilitará a execução de aplicações desenvolvidas em linguagem procedural e/ou em linguagem declarativa. O Ginga basicamente incluirá uma máquina virtual para aplicações procedurais e um interpretador para aplicações declarativas. A possibilidade de aplicações híbridas, ou seja, de base declarativa chamando pedaços de códigos procedurais, e de base procedural chamando pedaços de códigos declarativos, é uma das novidades do *middleware* brasileiro. A Figura 2.9 apresenta uma arquitetura mais detalhada do Ginga.

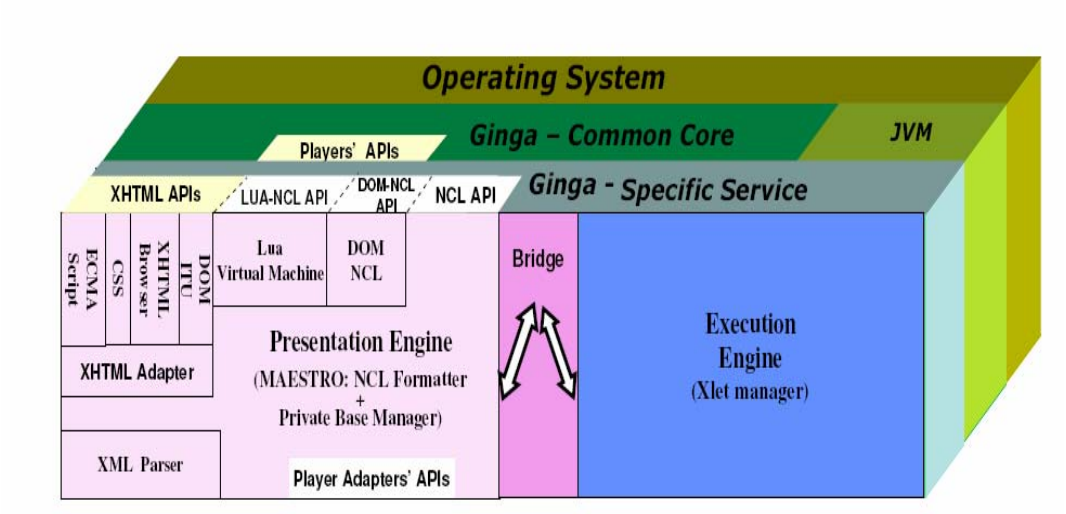


Figura 2.9 - Arquitetura Ginga
FONTE: SOUZA FILHO et al., 2006, p. 3. [17]

A programação declarativa é mais simples e exige uma formação menos especializada em programação. A primeira linguagem declarativa foi a SGML, que serviu de fundação para o desenvolvimento do HTML. Depois veio o XML, a linguagem SMIL que é padrão W3C e faz sincronismo temporal, e, finalmente, a NCL que faz sincronismo temporal e espacial.

A procedural é a maneira padrão de se programar e, diferentemente da declarativa, permite o uso das estruturas de controle e operações aritméticas e lógicas, exigindo uma formação mais especializada do programador em construção de algoritmos e programas. Exemplos de linguagem procedural são o Pascal, C, Java e outras.

Segundo Soares et al. [46], no Ginga, as aplicações procedurais devem ser desenvolvidas na linguagem Java ou na linguagem Lua. A escolha do Java pode ser justificada pelos benefícios advindos do alinhamento com outros padrões internacionais, o GEM. A linguagem Lua é uma opção de linguagem procedural em modelo de *script*. Nas aplicações declarativas será utilizada a linguagem NCL. A escolha do NCL se deve ao fato de ela ser uma linguagem extremamente flexível, possibilitando o desenvolvimento de aplicações bastante complexas, com sincronismo de mídias, mas ao mesmo tempo oferecendo um alto nível de abstração para os autores. Além disso, essa linguagem permite tratar as especificações declarativas constantes nas principais propostas de *middlewares* existentes no mundo, como objetos de um documento NCL, que pode possuir relacionamentos de sincronização com outros objetos do documento. O Ginga é subdividido em Ginga-NCL e Ginga-J.

Soares [16] diz que o Ginga-NCL, como linguagem de cola, permite o acesso a diversos tipos de objetos de conteúdo de mídia. O NCL não restringe nenhum conteúdo de mídia. Esses conteúdos podem ser imagens, vídeos, áudio, *scripts* ou arquivos de aplicação. Esses conteúdos são codificados e gerenciados por uma máquina NCL *Formater* e ficam armazenados em uma base privada. O gerenciador da base privada associa o documento NCL da base privada ao canal de TV. Em comparação com outros *middlewares* declarativos, um dos recursos de maior destaque é a capacidade de gerenciar sincronismo espacial junto com o temporal. Outra importante funcionalidade é a capacidade de implementar HAN, com a possibilidade de gerenciar multiusuário, multidispositivo e multiredes.

Para Soares et al. [46], o Ginga NCL possui dois módulos adaptadores que interagem com a máquina *engine* NCL: XHTML *Adapter* com *browser*, CSS e *ECMAScript*, e Lua *Adapter* [46]. Esses módulos também são encarados como objetos de mídia. O NCL também possui um *player* para algumas mídias executáveis. O Ginga NCL utiliza os programas Lua e Java também como nós de mídias. As linguagens Lua e Java são procedurais ou imperativas. O arquivo implementado em Lua é uma linguagem de *script* e o arquivo Java é uma linguagem de sistema.

O Ginga-NCL, por ser baseado em XML, separa conteúdo da estrutura. Suas *tags* orientam e descrevem como os objetos são estruturados e como se relacionam gerenciando eventos e ações no tempo e espaço. Existe disponível pela W3C, um *software* de sincronismo apenas temporal que é o SMIL. O sincronismo espacial ocorre quando se programa um objeto para movimentar-se por região específica tendo esse evento interpretado e respondido via programação Ginga-NCL. O Ginga-NCL também implementa eventos do DSM-CC conhecidos como NCL *streams events* e com o mesmo conceito dos *b-events* BML.

O Ginga-J, como um módulo procedural do *middleware* Ginga, tem como base a linguagem Java. Java serve também como base para as principais linguagens procedurais no mundo. Existe um padrão adotado mundialmente que é o GEM, no qual o Ginga-J também se adere na sua versão verde, como lembram Souza Filho et al. [17].

Na Figura 2.10, o Ginga-J é dividido nos módulos azul, amarelo e verde. O módulo azul contém a JMF 2.1.1 trata o sincronismo e faz integração de dispositivos com comunicação entre o receptor do sinal digital e qualquer dispositivo com uma interface compatível (Wi-Fi, Infravermelho, etc.), uma API de múltiplos usuários, que possibilita a interatividade com mais de um usuário simultaneamente, e uma API de Ponte NCL que

permite que as aplicações contenham aplicações NCL. O módulo amarelo disponibiliza API JMF 2.1, uma extensão para a API de aplicação do GEM, uma extensão para a API de canal de retorno do GEM e uma extensão para a API de Serviço de Informação do ISDB ARIB B.23. O módulo verde contempla pacotes *Sun* JavaTV, JMF 1.0, *Connected Device Configuration* - CDC -, *Foundation Profile* - FP -, *Personal Basis Profile* - PBP -, DAVIC, HAVi e DVB.

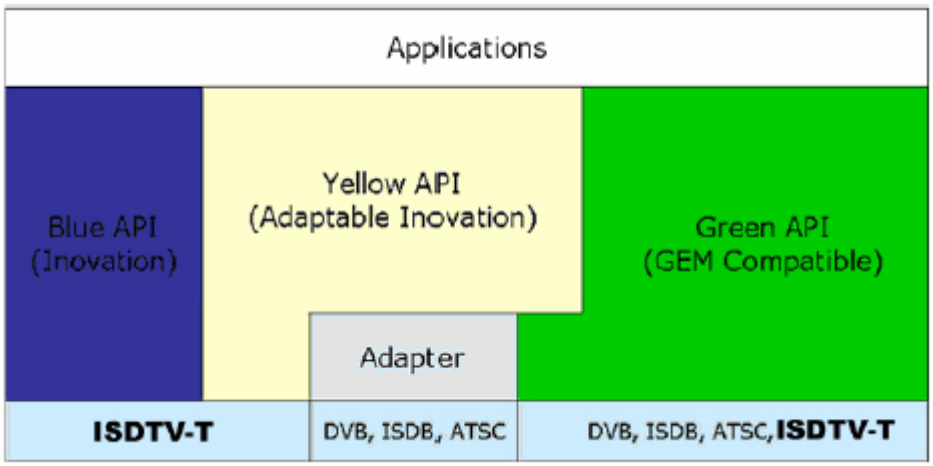


Figura 2.10 - Arquitetura Ginga
FONTE: SOUZA FILHO et al., 2006, p. 51. [17]

Na Figura 2.9, o Ginga *Common Core*, como complemento, aborda um grupo de funcionalidades comuns ao NCL e ao Java. Ele é composto por um decodificador de conteúdo para mídias PNG, JPEG, MPEG e outros formatos, uma função que obtém MPEG-2 *transport streams* via *carrousel* (DSM-CC) ou canal de retorno.

Como alternativa para implementação de aplicações procedurais, o Ginga possui uma máquina virtual Lua. O Ginga disponibiliza uma classe de objetos de mídia Lua, chamado de NCLua, que implementa a maioria das funcionalidades do Ginga-J. Em março de 2009, o GingaNClua está disponível antes do Ginga-J. O NCLua segue a sintaxe da linguagem Lua e foi uma API adaptada para funcionar embutido no Ginga-NCL e, consequentemente, transformando em NCLua [46].

Nesse início de 2009, o NCLua tem um conjunto de módulos disponíveis somente no emulador desenvolvido pelo laboratório Telemidia da PUC-RJ e esperado quando da disponibilização do Ginga nos receptores digitais. Os módulos são:

a) módulo *event*: permite que aplicações NCLua se comuniquem através de eventos;

b) módulo canvas: oferece uma API para desenhar componentes gráficos e imagens;

c) módulo *settings*: exporta uma tabela com variáveis definidas pelo autor do documento NCL e variáveis de ambiente reservadas;

d) módulo *persistent*: exporta uma tabela com variáveis persistentes.

O módulo *event* e o canvas foram os primeiros módulos a serem disponibilizados no emulador. Soares et al. [46] destaca que o NCLua pode tratar todos os eventos, inclusive os das mídias do código NCL, chamando o NCLua como um “tratador de eventos”. Com isto, conclui-se que os recursos disponibilizados pelo *Composer* para gerenciamento dos eventos podem ser resolvidos pelo NCLua. Todas as rotinas de tratamento do controle remoto, transmissões pelo canal de interatividade, e até sincronismo com o documento NCL, têm sua dinâmica resolvida pelo NCLua.

Para melhor explicar a relação do NCLua com o código NCL que o chama, segue a seguir uma sequência válida para os objetos NCLua elaborada por Soares et al. [46]. O formatador NCL identifica os elementos NCLua através da *tag* <media> incluída no código NCL.

a) O formatador NCL faz o pré-carregamento do NCLua.

b) O formatador envia ao NCLua os eventos correspondentes aos relacionamentos em que participa.

c) O NCLua passa a responder por todos os eventos programados para tratar.

d) O formatador pode destruir o NCLua quando quiser ou quando no estado *sleep*.

A existência desses diversos módulos Ginga, em conjunto com as *Bridges*, propicia implementações de aplicações híbridas tanto do NCL para Java como do Java para NCL, tornando o Ginga em um *middleware* bastante flexível e abrangente quanto às suas funcionalidades.

O projeto de desenvolvimento do *middleware* brasileiro tem também a preocupação de ser aderente à GEM, possibilitando funcionar em *middlewares* de outros países, com

algumas restrições, através de conversões automáticas. Logicamente que as aplicações que forem desenvolvidas exclusivamente para o mercado nacional terão mais facilidades em se beneficiar de todos os recursos apresentados pela linguagem NCL.

2.8 INTERATIVIDADE

Encarar a interatividade com uma coisa nova para a televisão é ignorar o passado. De acordo com Schwalb [35], a interatividade não é uma coisa nova na televisão, pois vem sendo explorada desde os anos de 1950 e com seu primeiro programa como o "*Dink Wink and You*", Figura 2.11. A BBC de Londres disponibilizou o serviço de teletexto em 1970 e em 1980, nos EUA, o Warner-Qube surgiu como o primeiro receptor de *Vídeo on Demand* - VOD. O conceito de TV interativa não é novo. Para se ter uma idéia da importância da interatividade, autores bibliográficos nem mais falam em TV Digital e, sim, em TV interativa, como Schwalb [35].

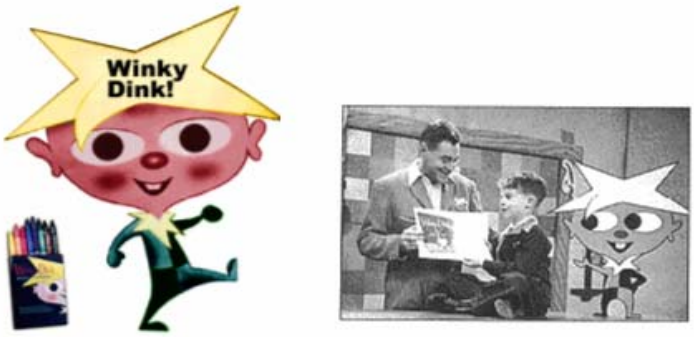


Figura 2.11 - Primeiro programa interativo para TV – CBS 1953
FONTE: MORRIS; SMITH CHAIGNEAU, 2005. [36]

A interatividade, no caso da TVDI, é a possibilidade de o telespectador trocar informações com o programa da televisão. Os novos receptores digitais estão também sendo projetados para gerenciar uma *Home Area Network – HAN*, conforme Lemos et al. [48]. Neste contexto, os PDAs também funcionam como dispositivo de interação com a TV. O controle remoto talvez seja o primeiro dispositivo de interatividade de uma televisão. Na Tabela 2.3, a diferença entre os tipos de TV sugere uma demanda de serviços muito maior.

TV não Interativa	TV Interativa
telespectador Passivo	Tespectador Ativo
SDTV	LDTV - EDTV - HDTV
Entreterimento	Vendas, Educacional e Comunicações
Propaganda dirigida a renda	Propaganda dirigida ao perfil
Modelo Unidirecional	Modelo bidirecional
Sinal Analógico	Sinal Analógico e Digital

Tabela 2.3 - TV interativa x não interativa
FONTE: CASTELARI; FERREIRA, 2007, p. 10. [27]

A interatividade é um recurso tão importante para a TV Digital que os terminais de acesso e *middlewares* são classificados de acordo com o grau de interatividade. Conforme a Tabela 2.4, os receptores digitais podem ser divididos em *Low-End* (básico), *Mid-Range* (intermediário) e *High-End* (avançado), como registra Piccolo [41]. O *settop box Low-End* funciona apenas como um receptor de TV Digital e suporta somente interatividade local e não possui canal de retorno. Os *settop boxes* do tipo *Mid-Range* geralmente são os escolhidos por operadoras de TV por assinatura, pois suportam todas as funcionalidades necessárias para uso das aplicações interativas básicas e possuem canal de retorno. O *settop box* do tipo *High-End* contempla todas as funcionalidades dos tipos de STB anteriores e traz como novidades o avanço na tecnologia e a personalização de usuário, interface para uma *home-network*.

	LOW-END	MID-RANGE	HIGH-END
Entradas	Sintonizador para o meio de transmissão e padrão adotado.	Sintonizador para o meio de transmissão e padrão adotado. Opcionalmente múltiplos sintonizadores para o PVR.	Sintonizador para o meio de transmissão e padrão adotado, múltiplos sintonizadores, ADSL.
Saídas de vídeo	NTSC/PAL MPEG2 PAL-M no caso do Brasil.	NTSC/PAL MPEG2 PAL-M no caso do Brasil.	NTSC/PAL MPEG2 PAL-M no caso do Brasil. HDTV e suporte para Wide-screen.
Saídas de Áudio	*Dolby* digital (phono ou digital).	Dolby digital (phono ou digital)	Dolby digital (phono ou digital).
Interface de dados	RS232	IEEE-1394, USB, RS232.	IEEE-1394, USB, RS232. Opcional: Bluetooth.
Canal de Retorno	Modem telefonia fixa (opcional)	Modem telefonia fixa, chipset para voz sobre IP	Modem telefonia fixa, chipset para voz sobre IP, ADSL.

Performance Gráfica	256 cores. Quatro planos: vídeo (tela cheia ou escalonado), imagem fixa, imagem de fundo.	256 cores. Quatro planos: vídeo (tela cheia ou escalonado), imagem fixa, imagem de fundo.	256 cores. Quatro planos: vídeo (tela cheia ou escalonado), imagem fixa, imagem de fundo.
CPU		> 100 MIPS	> 300 MIPS
Memória	Flash: 1 módulo de 4 MB. SDRAM : 4-8 MB.	Flash: 8 MB. SDRAM : 16 MB.	Flash: 16-64 MB. SDRAM : 32-128 MB.
Opcionais		Controle remoto. Teclado sem fio. Hard-disk para funcionalidades de PVR	Controle remoto. Teclado sem fio. Hard-disk para funcionalidades de PVR

Tabela 2.4 - Tipos de STB
FONTE: Adaptado de PICCOLO, 2002. [41]

As características dos tipos de receptores digitais podem ser observadas com a evolução dos *middlewares*, que ocorreu de acordo com a capacidade de interatividade. Segundo Morris e Smith-Chaigneau [36], o *middleware* MHP somente teve sua versão disponibilizada para a versão de interatividade plena na sua versão 1.1., e tem seus perfis diferenciados conforme a Tabela 2.5.

Os receptores digitais ou *settop box* são divididos em perfis desde a especificação do JavaTV e das versões do MHP. Estes perfis foram divididos com base no grau de interatividade. Os três perfis são: *Enhanced Broadcast Profile*, que é de baixo custo e apenas atua como receptor de sinal digital via operadora de TV; o *Interative Broadcast Profile*, que considera um receptor com canal de retorno onde as aplicações podem ser baixadas via protocolo IP; e, finalmente, o perfil *Internet Access Profile*, que possibilita utilizar aplicações de Internet como *E-mail*, *WebBrowser* e aplicações HTML.

Internet Access Profile	Não disponível	+ Java Internet client APIs + WEB-Browse e E-mail + DVB-HTML (opcional)
Interactive Broadcast Profile	+ DVB Java API extensão para canal de retorno + Protocolos de canal de retorno incluindo IP	+ DVB-HTML (opcional) + Xlet download via HTTP + Aplicações Inner
Enhanced Broadcast Profile	Java VM DVB Java APIs Formatos: MPEG, GIF, JPEG, PNG, etc Protocolo de transporte Broadcast	+ Aplicações armazenadas + Smart Card APIs
	MHP 1.0	**MHP 1.1**

Tabela 2.5 - MHP – *Multimedia Home Plataform*
FONTE: MORRIS; SMITH-CHAIGNEAU, 2005, p. 12. [36]

A interatividade tornou-se um recurso fundamental para a TV Digital. O telespectador deixa de ser meramente passivo para se tornar ativo. O telespectador pode interagir com a televisão e escolher seu próprio conteúdo. Já existem tipos de aplicação como o EPG, que possibilitam ao usuário escolher sua programação. O grau de interatividade, segundo Fernandes [37], pode ser classificado em três categorias: local, intermitente e plena.

Na TV Digital, a interatividade local ocorre quando o usuário troca informações com a aplicação sem a emissora de sinal digital receber informações de retorno. O *Eletronic Program Guide* - EPG - é um tipo de aplicação para interatividade local.

Na interatividade intermitente, o receptor digital apresenta um dispositivo de *hardware* chamado canal de retorno. É chamado intermitente devido ao fluxo de dados ser somente unidirecional, ou seja, do telespectador para a emissora via provedor de interatividade, porém a emissora não consegue enviar respostas ao telespectador. O canal de retorno é considerado não-dedicado. O provedor de interatividade pode ser acessado via linha telefônica ou via ADSL. Este tipo de receptor digital ou *settop box* é muito utilizado para receber aplicações do tipo para votações, pesquisa de opinião e *quiz*. Estas aplicações não necessitam que o difusor envie respostas ao telespectador.

Finalmente, nas aplicações para interatividade plena, os *receptores digitais* necessitam ter canal de retorno dedicado. O canal de retorno para emprego de canais de comunicação de dados pode utilizar as tecnologias de telefonia celular, redes *ad-hoc*, radiofrequência, telefonia fixa com modem telefônico, serviços ADSL ou *Power Line*

Communications - PLC. Este receptor digital apresenta um dispositivo de *hardware* que possibilita o envio e a recepção de dados para a operadora de *broadcast*, a interatividade é bidirecional. Este recurso de interatividade possibilita que a televisão funcione como um computador. Os recursos deste tipo de receptor possibilitam o acesso à Internet e o dispor de algumas aplicações como *e-mail*, *chat*, compras, *homebanking* e educação à distância através da navegação. Outra novidade é a possibilidade da comunicação entre telespectadores.

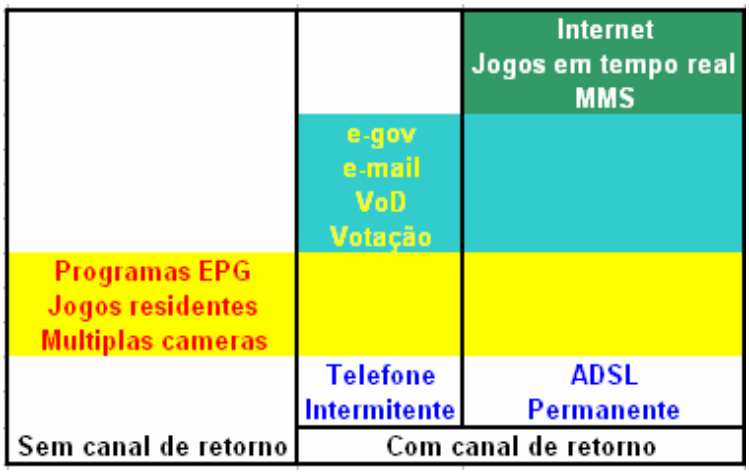


Figura 2.12 - Tipos de interatividades
FONTE: Adaptado de TAVARES et al., 2005. [71]

Com a TV Digital interativa, abre-se uma oportunidade muito grande de desenvolvimento de um mercado produtivo onde as demandas serão cada vez maiores. A pesquisa nos tipos de aplicações e padrões de usabilidade já experimentadas nos outros paises passa a ser um trunfo para o rápido desenvolvimento de aplicações interativas para a TV Digital brasileira.

2.9 CENÁRIOS DE APLICAÇÕES INTERATIVAS PARA TV DIGITAL

Para desenvolvimento de aplicações ou de ferramentas para interatividade, é muito importante conhecer as classes de cenários das aplicações como referência para o desenvolvimento desses tipos de aplicações. Esta seção tem como objetivo principal esclarecer os padrões para implementações dos diversos tipos de aplicações.

Alguns autores como Schwalb [35] definem três tipos de aplicações conforme a Figura 2.13 a seguir. Além desses tipos de aplicações existem outras, como programas interativos de música, programas educacionais, cursos *on-line*, etc.

Figura 2.13 - Tipos de aplicações
FONTE: MORRIS; SMITH-CHAIGNEAU, 2005, p. 10. [36]

No grupo de entretenimento, os EPGs, aparecem como a mais básica aplicação interativa para TV Digital. Schwalb [35] descreve as principais funcionalidades de uma aplicação EPG. Os recursos de pesquisar, *preview*, selecionar e até na opção de assistir aos programas, os *VODs* podem estar presentes. A Figura 2.14 apresenta alguns modelos.

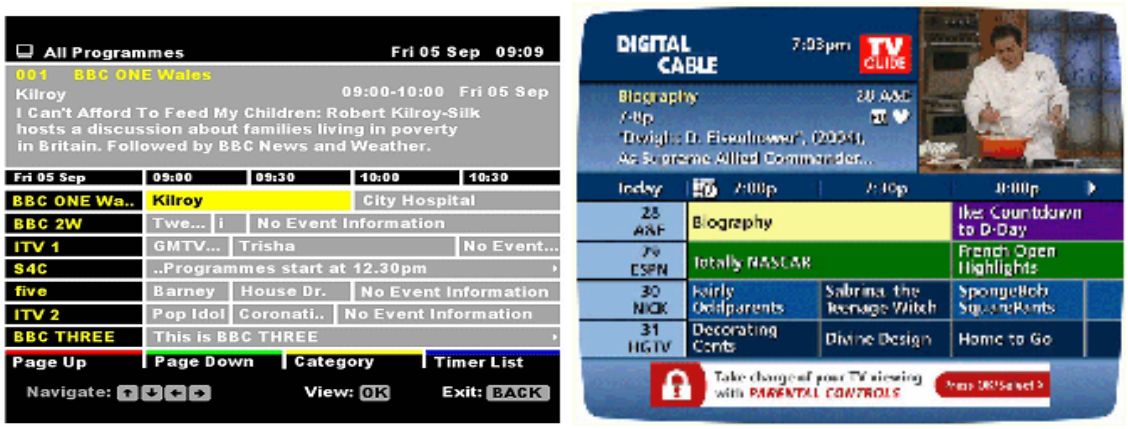
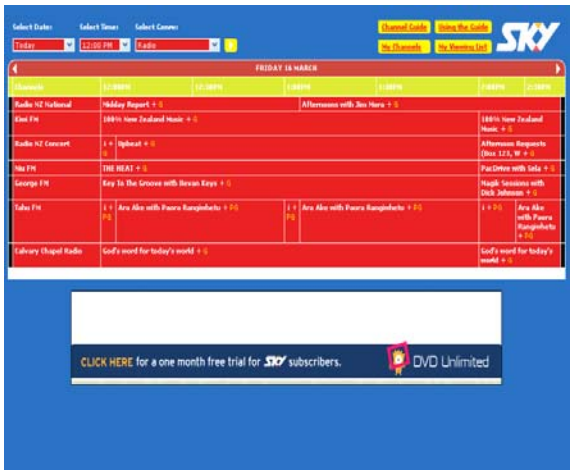


Figura 2.14 - EPG
FONTE: COMPUTERWORLD, 2008. [59]

Nas aplicações de VOD, a disponibilização de *trailler* é fundamental, e dependendo da quantidade de *traillers* a requisição é feita pelo canal de interatividade. Schwalb [35] recomenda que se observe na Figura 2.15 que aplicações para VOD disponibilizam um vídeo virtual Os *games show* e *multiplayer gaming* podem apresentar diferenças significativas. Os *multiplayers* necessitam do canal de retorno para a interação, mas nem todos os *games shows* necessitam do canal de interatividade. Os jogos de apostas também necessitam de canal de retorno ou interatividade.

Figura 2.15 - VOD
FONTE: COMPUTERWORLD, 2008. [59]

As aplicações do grupo de TV-*Commerce* são feitas para um perfil de usuário que necessita de segurança e brevidade na resposta da aplicação. As do grupo de comunicação também necessitam de segurança e se aproximam um pouco mais das aplicações da Internet. Todas estas aplicações podem ser observadas nas figuras a seguir.


Channel Guide
Using the Guide
My Channels
My Viewing List
FRIDAY 16 MARCH
CLICK HERE for a one month free trial for SKY subscribers.
DVD Unlimited


ROCK
WE WILL ROCK YOU
24 HOURS A DAY
ROCK
NEO ROCK
90's ROCK
80's ROCK
70's ROCK
HARD & HEAVY
PURE ROCK


TV
SAT.1
ASTRA Mosaic
VOX
RTL
super RTL
PHOENIX
QVC
arte
KiKA
CANAL+
FRANCE
Radio Mosaic
Home Shopping Europe D


For the latest details on World events, press BLUE then select News.
BBC 15:27 TV PLUS HELP LATEST MENU

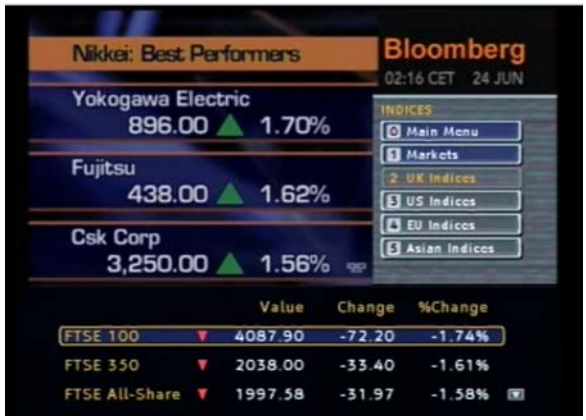

Taquilla
02:25
EN QUE PIENSAN LAS MUJERES
Próximo pase 02:30 Dial 102
FUTBOL
ADULTO
AYUDA
OK Seleccionar
Salir


TV Guide
itv DIGITAL
03 ITV1 Carlton/LWT
14:57
14:55 What A Carry On
15:25 Tweenies CBBC
15:45 Arthur CBBC
16:10 Mona The Vampire CBBC
16:20 Home Farm Twins CBBC
A compilation of some of the funniest moments from the riotous Carry On films, starring Kenneth Williams, Hattie Jacques and Sid James. (S/T)
press ◄ ► to change channel
press green for regional listings
Check TV7 for this week's full listings
back help menu


THIS ROOM MONITORED T ALL TIMES
106 Sky One
7.07pm Mon 21
NOW The Simpsons
7:30pm The Simpsons
Search Time
Search Channel
Search Favourite


Theme Search
NOVA
Type
MOVIE
Subtype
Action
Day
All Week
Time of Day
All Day
Select
Confirm
Return


Kids Awards 2003
Vote for your favourites in a different category every week!
Vote Now!
Push
VOTE
COMP
TOON TIPS
MISS US
PUSH A BUTTON TO JOIN THE FUN!


Nikkei: Best Performers
Bloomberg
02:16 CET 24 JUN
Yokogawa Electric
896.00 1.70%
Fujitsu
438.00 1.62%
Csk Corp
3,250.00 1.56%
INDICES
0 Main Menu
1 Markets
2 UK Indices
3 US Indices
4 EU Indices
5 Asian Indices
Value Change %Change
FTSE 100 4087.90 -72.20 -1.74%
FTSE 350 2038.00 -33.40 -1.61%
FTSE All-Share 1997.58 -31.97 -1.58%


NEXT
SCORE
0
LEVEL
0
LINES
0
TETRIS
GO PLAY TV
OPTIONS


mmm... Pizza!
Free Delivery
Large £9.99
*Pepperoni Passion
*Mighty Meaty
*Vegetarian Supreme
*Deluxe
*Full House
*Mixed Grill
*Ham & Pineapple
PEPPERONI PASSION
Press SELECT for details
Open.....
Go to
Add item

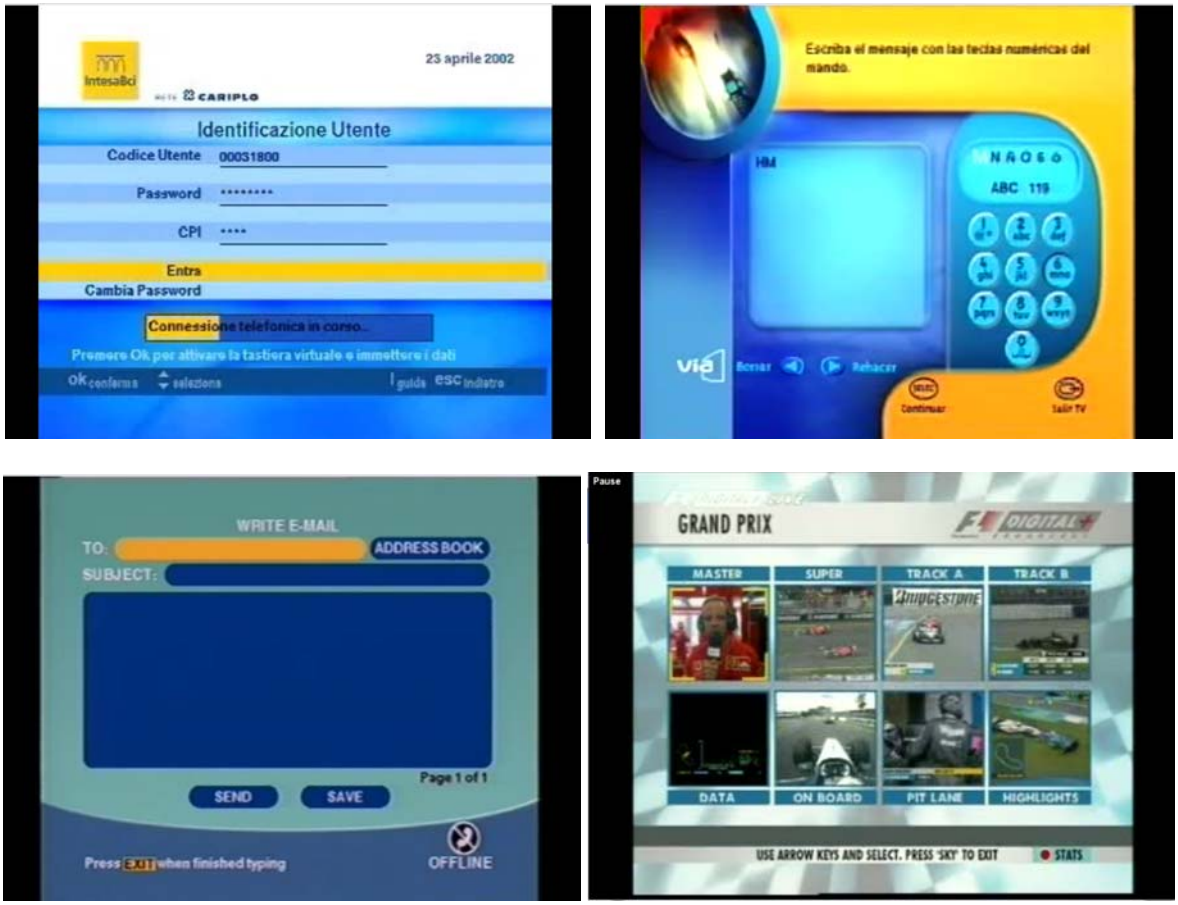


Figura 2.16 – Aplicações interativas
FONTE: BROADANDBANANAS, 2009. [58]

Morris e Smith-Chaigneau [36] também definem as aplicações de quatro tipos: *Service-bound, Unbound, Stored* e *Native.*

Aplicações do tipo *Service-bound* são baixadas via operadoras de *broadcast*. Estas aplicações são baixadas toda vez que o usuário deseja e podem ser iniciadas pelo usuário automaticamente e disponibilizadas por canais de TV específicos. Estas aplicações são automaticamente desativadas quando o usuário muda de canal. Um exemplo de aplicação *Service-bound* é um programa de *Quiz.*

Aplicações do tipo *Unbound* são também baixadas via operadas de *broadcast*, mas são independentes do canal a que o usuário está sintonizado. São aplicações que não são associadas a nenhum programa de TV. Um exemplo são os EPGs, *home shopping*, *T-bank*, *games* e outras.

Aplicações do tipo *Stored* são também baixadas como as do tipo *Unbound*, mas são armazenadas localmente nos receptores e de rápida carga para uso. Esses tipos de aplicações

também funcionam como EPG e gerenciadores de aplicações. Estas aplicações têm prioridade baixa e podem ser rejeitadas pelo telespectador.

Aplicações do tipo *Native* são gravadas pelo fabricante de receptor digital e também utilizada para explorar os recursos do *settop box*.

Após se conhecer as principais características das aplicações, fica mais difícil não aceitar as limitações da interface das aplicações para TV Digital. A necessidade de superação das limitações, normalmente leva o artista ou autor a soluções com alto grau de qualidade gráfica.

2.10 USABILIDADE NA TV DIGITAL INTERATIVA

Para Becker [56], o conceito de usabilidade, apesar de ter surgido nos computadores, tem na norma ISO 9241, uma boa definição que é a eficácia, eficiência e a satisfação do usuário em interagir com um sistema. Esses três elementos que definem o conceito de usabilidade são muito difíceis de serem atingidos pela aplicação. Aplicações para TV Digital interativas são feitas para usuários com conhecimento pouco especializado ou específico para a aplicação e essas características ressaltam a importância da usabilidade e o quanto (como) esse conceito é fundamental.

A heterogeneidade dos futuros usuários dos sistemas interativos com diferentes níveis de formação, com alguns não necessitando de treinamento, por estarem já alfabetizados digitalmente, e outros sem nenhuma alfabetização digital, é o público-alvo dos aplicativos interativos. Becker [56] diz que mesmo os analfabetos totais, que não conseguem sequer relacionar um número ao significante alfanumérico, entendem a televisão. O público da TV interativa é muito mais variado do que os usuários de computadores e mesmo da Internet.

A TV interativa, vista como uma combinação de TV e mais o canal de retorno, também dito como canal de interatividade, possibilita ao usuário receber e enviar informações, e esta comunicação TV-telespectador deve ser facilitada por um bom padrão de usabilidade. A televisão conversa com as pessoas, se faz entender por ela mesma. Essa é uma característica que não pode ser ignorada com a televisão digital interativa, sob pena de elitizar uma das poucas tecnologias democráticas existentes no país hoje.

Apesar de todas as aplicações interativas serem recheadas de controles gráficos e de informações multimídia, a existência de recursos enriquecidos de atributos e possibilidades

como cores, estilos, formatos, sons etc., aumenta a complexidade da aplicação e pode dificultar o entendimento pelo usuário. Este é um desafio constante em aplicações interativas, mas para a TV Digital, é muito mais difícil projetar uma aplicação para colocá-las em acordo com as altamente variadas características físicas, cognitivas e sociais dos usuários. Um padrão de referência em usabilidade como o proposto pela BBC de Londres pode facilitar o trabalho dos projetistas justamente nesta questão. Quando se fala em aplicação para TV, é inevitável a comparação com os recursos disponíveis nas aplicações da WEB e dos computadores pessoais. A limitação dos recursos de uma TV Digital interativa e a demanda por mais recursos para a TV vêm despertando bons debates em fóruns para compreensão com relação aos recursos que estão disponíveis para o telespectador.

Muitos dos padrões de usabilidade para aplicações interativas com mídias, textos e caixas de entradas e saídas de informações sofrem grande influência do uso destes elementos no computador. Segundo Becker [56], no computador a interface e a imagem podem coabitar de maneira mutuamente exclusiva. Na tela do computador, a TV e a interface podem repartir a tela e esses dois recursos podem ser manipulados ao mesmo tempo. Já na TV, o usuário trabalha no modo de operação pausada.

A TV apresenta diversas diferenças em relação a um PC, que não podem ser ignoradas. A TV tem uma tela de menor resolução, área periférica sujeita a distorção, disponibiliza dispositivos bastante limitados com relação a um PC, não oferece rolagem horizontal, apresenta maior lentidão nas respostas. Todas estas características limitam bastante os recursos de usabilidade em um projeto para aplicações de TV Digital interativa. Por exemplo, esta menor resolução obriga à utilização de fontes com formato bem maior e com uso muito maior de área na tela. A consequência disto é um espaço muito menor para disponibilização de informações interativas.

A BBC de Londres, como uma das precursoras de TV Digital na Europa e no mundo, depois de muitos testes e estudos, recomenda alguns formatos de fontes para textos nessa modalidade e cores para utilização dos elementos gráficos, imagens e botões.

A SKY do Brasil vem adotando padrões próprios conforme a Figura 2.17 que fogem do padrão da BBC, o que permite até observar um padrão de usabilidade que não respeita as limitações do usuário. Os tamanhos de fontes e as cores não seguem um bom padrão.

Figura 2.17 - Aplicações veiculadas na SKY do Brasil
FONTE: BECKER et al., 2006, p. 7. [51]

Observe que as fontes têm tamanho menor que 18 e as cores de fundo claras não são excluídas. A BBC [57] apresenta sete considerações como base para implementação das aplicações interativas para TV Digital:

a) O tamanho das fontes no texto não deve usar tipos menores que 24 pontos;

b) nenhum texto dever ter fontes menores que 18 pontos em qualquer circunstância;

c) as entrelinhas nos textos devem ser maior que o normal;

d) os espaços entre os caracteres deve ser aumentado em 30%;

e) um texto deve ter no máximo 90 palavras aproximadamente;

f) textos separados em pequenos blocos para serem lidos instantaneamente;

g) textos claros com fundos escuros.

A BBC [57], no seu guia de estilos, reforça o perfil visual das aplicações interativas introduzindo o conceito de que todos os serviços devem ser fáceis de se usar, tanto para jovens como para pessoas idosas. A BBC afirma que uma em 30 aplicações tem um prejuízo visual. Ao criar gráficos, fontes claras devem ser usadas, e as cores devem apresentar um bom contraste entre o texto e o fundo. Observe na Figura 2.18 a seguir:

Figura 2.18 - Exemplos da BBC
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 2. [57]

A BBC [57] também define alguns termos técnicos para TV Digital como *Picture Safe*, *Text Safe*, *Pixel Size*, *Widescreen*, *Tall and thin*, *Short and fat* (*or wide*), *Black bits - top & bottom*, *Full Height Anamorphic*. Todos estes termos são situações que acontecem na TV Digital interativa e ajudam ao autor de aplicações a entender suas limitações e adaptar suas aplicações. Observe nas figuras a seguir o significado destes termos e, principalmente na Figura 2.21, que os padrões gráficos não são alterados mesmo que SDTV 4:3 para 16:9. Observe na Figura 2.19, a seguir, que a formatação dos elementos gráficos não é modificada quanto a sua posição, ou seja, a adaptação de conteúdo praticamente não existe. Existem outros motivos como robustez da aplicação *versus* tamanho que justificam a não adaptabilidade.

Figura 2.19 - Adaptação de conteúdo
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 9. [57]

Tall and thin

Black bits – top & bottom

Figura 2.20 - Aplicação 16:9 em tela 4:3
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 10. [57]

Short and fat

Black bits – left & right

Figura 2.21 - Aplicação 4:3 em tela 16:9
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 10. [57]

A BBC também recomenda o corte em pixel para aplicações SDTV em tela HDTV conforme a Figura 2.22 abaixo:

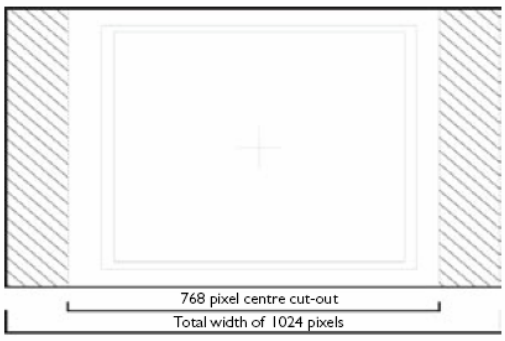


Figura 2.22 - Produção para widescreen
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 12. [57]

Existem outras técnicas aplicadas pela BBC, como a *Centre Cut Out* e a *Letterbox* conforme a Figura 2.23. Nos *displays* de 16:9 a adaptação se faz na relação 14:9.

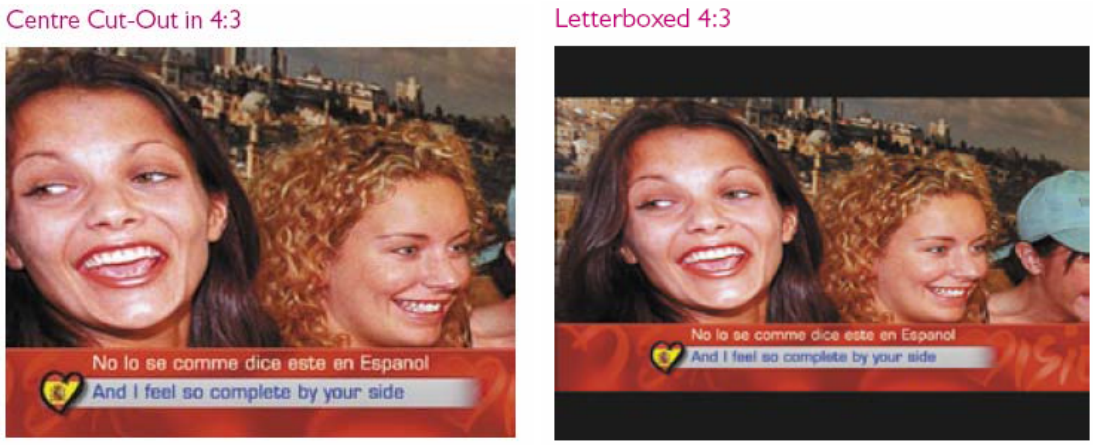


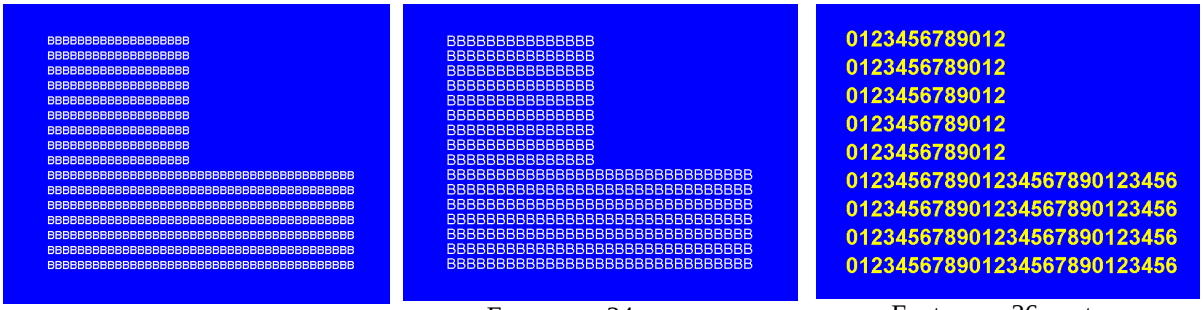
Figura 2.23 - Adaptação para display 4:3
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 13. [57]

Observe, na Figura 2.24, que a BBC adota como padrão a fonte Tireseas, Gil Sans com formatos em 18, 24 e 36 pontos. As fontes devem ser sem serifa.

Figura 2.24 - Fontes usadas pela BBC
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 15-16. [57]

Estes padrões de tamanhos de fontes são muito importantes para o planejamento de textos nas aplicações para TV Digital interativa. Nas figuras abaixo, o laboratório de TV Digital da UnB simulou a utilização máxima de caracteres por tamanho de fontes em um padrão SDTV. Conforme as recomendações da BBC, os caracteres devem ter um espaçamento um pouco maior que o normal, 30%, devido ao brilho excessivo que uma TV

tem. Neste contexto, textos para fontes de tamanho 18 pontos devem ter no máximo 19 caracteres por linha e 9 linhas para ¼ de tela. No caso de 24 pontos deve ter 15 caracteres por linha e 9 linhas para ¼ de tela, e para 36 pontos deve ter 12 caracteres por linha e 5 linhas para ¼ de tela. Os padrões de resolução gráfica dos programas podem ser LDTV: 320 X 240, SDTV: 640 X 480, EDTV: 720 X 480 e HDTV: 1920 X 1080.



Fonte com 18 pontos | Fonte com 24 pontos | Fonte com 36 pontos

Figura 2.25 - Distribuição de fontes em padrão SDTV

Com relação à navegação, a BBC [57] também apresenta recomendações de usabilidade. A BBC tem também seis considerações:

a) Dizer ao usuário onde está, como começar, e aonde pode ir em seguida;

b) fornecer a qualquer hora o *feedback* todas as vezes que alguém executar um comando;

c) ensinar o usuário como usar os serviços em segundos;

d) relacionar modelos mentais maiores;

e) apresentar mecanismos navegacionais previsíveis e consistentes;

f) incentivar a livre navegação com poucos trajetos predeterminados e limitados, e fornecer a saída rápida sob a forma de uma rota da saída, ou o acesso próximo-imediato ao vídeo da tela cheia.

A BBC afirma que uma boa navegação deve providenciar uma navegação consistente, ajuda o usuário a tomar decisões, prover destaques, tratar as latências de forma explícita e amigável para o usuário, comunicar para esperar que a aplicação esteja toda carregada, educar o usuário a utilizar atalhos, e habilitar o usuário a sair da aplicação de qualquer ponto.

Com relação às teclas básicas de navegação do controle remoto, a BBC sugere usar, de preferência, a fonte Gil Sans 22 para seus textos, manter posição horizontal e sempre na sequência vermelho, verde, amarelo e azul, conforme a Figura 2.26 a seguir. Manter a posição das cores, mesmo que todas as cores não sejam disponibilizadas, e manter ao máximo as funções das teclas em toda a aplicação.

Figura 2.26 - Teclas de navegação ou funções
FONTE: BBCi & INTERACTIVE PROGRAMMES, 2005, p. 30-31. [57]

A BBC também recomenda evitar a navegação e uso de números do controle remoto para navegação. Caso seja necessária a utilização dos números, manter suas funções por toda a aplicação. Os textos instrucionais devem ser utilizados na região dos botões coloridos, ou seja, no rodapé da aplicação. Com relação aos logotipos, a BBC recomenda utilizar em tamanho máximo de 10 milímetros.

Os estudos de usabilidade são fundamentais para a construção de componentes gráficos e para a compreensão da necessidade de não utilização dos recursos de outros ambientes como a Internet e o computador. A padronização desses detalhes de usabilidade nas aplicações para TV Digital no Brasil é importante para uma melhor aceitação dessas limitações pelos usuários. A iniciativa da SKY do Brasil em definir seu padrão pode gerar uma expectativa distorcida dos recursos gráficos em uma aplicação para TV Digital interativa.

Utilizar o padrão de usabilidade recomendado pela BBC pode soar como apenas uma decisão, mas não se deve desconsiderar a experiência da BBC em TV Digital e lembrar do pioneirismo dela em aplicações interativas desde o *Multimedia and Hypermedia Expert Group* - MHEG.

2.11 CONCLUSÃO

Muitos dos experientes desenvolvedores de aplicações em geral podem fazer uma analogia com a sua plataforma de trabalho, e concluir que as aplicações da TV Digital chegarão ao estágio das aplicações *desktop*, porém, ao desconhecer estas características destacadas nesse capítulo, poderão também colaborar bastante para a inviabilidade de uso de suas aplicações.

A TV Digital, com suas limitações de banda, força o desenvolvedor a economizar recursos de programação. Este esforço, ao invés de depreciar a aplicação, pode significar uma diminuição de custos, devido a menor utilização de banda, e consequentemente valorizando o trabalho do desenvolvedor.

Em contrapartida, para esta limitação de tamanho das aplicações, o uso de imagens com boa qualidade e boa compactação, junto com componentes gráficos para facilitar a interatividade, justificam o uso das aplicações desenvolvidas com linguagens de *script*, valorizando a importância do LuaComp neste contexto.

Neste capitulo, os tipos de linguagens como declarativas e procedurais também têm uma influência muito grande na produção das aplicações interativas. Importante conhecer as vantagens e desvantagens na escolha do tipo de linguagem, mas a tendência é que as aplicações sejam híbridas.

Finalmente, um autor de aplicações que utilize uma ferramenta de autoria sem conhecer os padrões de usabilidade, os diversos cenários e tipos de aplicação, mesmo sendo um experiente desenvolvedor de aplicações para computadores (experiente), não concluirá um produto de acordo com o contexto da TV Digital interativa.

3 FERRAMENTAS DE AUTORIA

3.1 INTRODUÇÃO

Para Pressman [3], desde a crise do *software*, a engenharia de *software* vem desenvolvendo ferramentas *Computer Aided System Engineering* - CASE - que visam a facilitar o processo de produção. Em Farias [4] se define ferramenta CASE como uma tecnologia que consiste em fazer uso de ferramentas computacionais para o desenvolvimento de programas. Pressman [3] diz que a engenharia de *software* propõe o uso de programas que auxiliem as fábricas de *softwares* na produção de sistemas com as seguintes características: uma coleção de ferramentas úteis que facilitem a construção de um produto, um leiaute organizado que também facilite encontrar rapidamente as ferramentas e usá-las de maneira eficiente e um artesão habilitado que entenda como usar essas ferramentas efetivamente.

Para Guimarães [1], uma ferramenta de autoria é um tipo de ferramenta CASE que oferece um ambiente de programação de alto nível e também permite ao programador abstrair fases ao especificar tarefas. Ferramentas de autoria podem ser empregadas a fim de abstrair do autor de toda ou pelo menos de parte da complexidade representada pela utilização de uma linguagem de programação na criação de aplicações.

Uma ferramenta CASE oferece um Ambiente Integrado de Desenvolvimento – IDE - e se baseia na metodologia conhecida como Rápido Desenvolvimento de Aplicações - RAD. O conceito de RAD surgiu na década de 1970, mas somente com a publicação do livro de James Martin, *Rapid Application Development*, em 1991, as ferramentas CASE tomaram força. Piske e Seidel [2] afirmam que RAD é uma técnica que engloba desenvolvimento interativo, construção de protótipos e utilização de ferramentas CASE (engenharia de *software* assistida por computador). Para os autores, as ferramentas RAD, devido a sua facilidade de implementação de aplicativos, encorajam os usuários a também participar do processo de análise e *design*.

Segundo Farias [4], as primeiras ferramentas CASE comerciais surgiram por volta de 1982, ganhando força a partir de meados de 1990, com o surgimento do Delphi e Visual BASIC. Estas ferramentas RAD são produtos compostos de um ambiente gráfico de produção, também chamado de IDE, e um compilador para uma linguagem de programação específica. O Delphi da Borland e o Visual Basic da Microsoft foram as ferramentas de

autorias proprietárias mais utilizadas nos últimos tempos e ainda são muito usadas hoje. Estas ferramentas tiveram uma influência muito grande na programação, difundindo padrões de interface e usabilidade nos aplicativos.

A ferramenta CASE é responsável pela padronização de produtos, automação do processo de *software*, reengenharia e engenharia reversa, interoperabilidade de ferramentas, geração automática de código e coleção de dados. Farias [4] também afirma que uma ferramenta CASE nunca minimizará ou simplificará o rigor intelectual em especificar, projetar, implementar e manter a qualidade de *software*, não corrigirá problemas organizacionais, não medirá qualidade e não resolverá problemas específicos da área de conhecimento dos engenheiros.

Existem vários modelos para processos de *software* e, independente da escolha, a ferramenta CASE não está fora deste contexto. Nenhum desses modelos está fora das atividades básicas de especificação, desenvolvimento, validação e evolução. Os principais modelos de processo de desenvolvimento de *software* são: modelo cascata, modelo de prototipagem, modelo de processos evolucionários e modelos formais. Sendo permeada por estes modelos de processos de *software*, a ferramenta CASE tem sua adoção cada vez maior na automatização dos processos de desenvolvimento de *software*.

Em Farias [4], a tecnologia CASE pode ser divida em três categorias: tecnologia de suporte ao processo de produção, tecnologia de gerência de processo e tecnologia Meta-CASE. A tecnologia de suporte ao processo de produção apresenta ferramentas que suportam as atividades de especificação, *design*, implementação e teste, e tem como exemplo as ferramentas para UML. As ferramentas CASE também dão suporte à tecnologia de gerência, direcionam e ajudam a gerenciar as tarefas numa atividade específica, como por exemplo o *Project*. Finalmente, as ferramentas de Meta-CASE são ferramentas que criam ferramentas. Uma ferramenta CASE podem também ser dividida por tipos de funcionalidades, como: gerenciamento, edição, gerenciamento de configuração, prototipagem, suporte a programação, análise de programas, testes, depuração, documentação, e reengenharia. Uma ferramenta de autoria pode ser considerada como do tipo prototipagem.

Independente do tipo de produto que se deseja gerar com o uso de uma ferramenta CASE, esta ferramenta veio para suprir a necessidade de acelerar o tempo de produção de aplicativos perante uma crescente demanda. A TV Digital, com a interatividade, traz uma nova demanda por aplicativos e torna-se inevitável a utilização de ferramentas CASE. As

ferramentas de autoria, ferramentas CASE para prototipagem, têm tido bastante destaque no meio de produção de aplicações para TV Digital.

A ferramenta CASE do tipo para prototipagem ajuda na criação rápida de protótipos, suporte à definição de leiaute de telas, integração de telas e geração de código fonte. Guimarães [1] também defende que a ferramenta de autoria pode ser empregada com objetivo de abstrair o autor de toda ou parte da complexidade de se utilizar uma linguagem de programação. O usuário de uma ferramenta de autoria ganha muito tempo na produção de um aplicativo que envolve interatividade, pois pode se concentrar mais no processo artístico da criação. Muitas das ferramentas de autoria não são feitas para usuários leigos, mas são projetadas para atender a demandas por níveis de *expertise.*

Segundo Guimarães [1], as ferramentas de autoria têm seus recursos atrelados aos recursos da própria linguagem. Estas ferramentas oferecem ambientes gráficos para facilitar a abstração do usuário e para isso oferecem estruturas auxiliares. Estas estruturas auxiliares como *Project Manager* e *Properties*, servem como apoio para o desenvolvimento de qualquer tipo de aplicação, seja *Web*, *Desktop* ou para TV Digital.

O desenvolvimento de aplicações interativas para TV Digital exige talento artístico dos profissionais ligados à TV, devido ao uso de mídias como imagens, vídeos e sons, conhecimento básico em programação e conhecimentos das características do funcionamento do sistema de transmissão. As ferramentas de autoria disponíveis atualmente para TV Digital exigem conhecimento técnico aprofundado. Apesar do SBTV ainda não ter seu *middleware* disponível para desenvolvimento e testes de aplicação, a ferramenta de autoria como o *Composer* e emuladores de receptores virtuais já estão sendo disponibilizados no mercado. As aplicações de TV Digital são classificadas como modelos hipermídia e interativos.

Para Waker [6], aplicações hipermídia podem ser definidas como aplicações que misturam os conceitos de multimídia e hipertexto. Segundo Salgado [7], o conceito de hipermídia é definido como um sistema que manipula um conjunto de informações pertencentes a vários tipos de mídia (textos, imagens, sons e outros), que podem ser lidas de forma não linear através dos diversos caminhos de acesso disponíveis. Primo [8] afirma que o hipertexto é uma ferramenta que permite a navegação e, consequentemente, a interatividade não linear. Um documento hipermídia possui “nós” como em todo documento hipertexto para acesso a mídias.

Segundo Rodrigues [21], hoje, no mundo dos tipos de linguagens para o desenvolvimento de aplicativos para TV Digital, existem muitas aplicações que foram implementadas em linguagens declarativas e/ou procedurais ou imperativas. Conforme já explicado nos capítulos anteriores, a principal diferença é que na linguagem declarativa a curva de aprendizado é menor que nas linguagens procedurais. A linguagem declarativa não exige conhecimento das estruturas de controle que existem as linguagens procedurais. Para Rodrigues e Soares [15], algumas ferramentas de autoria que trabalham melhor com sincronismo temporal e espacial são baseadas em linguagens declarativas e, em contrapartida, estas aplicações são limitadas quanto aos recursos de programação, por isso a valorização das soluções híbridas. As linguagens declarativas são baseadas no XML. A maioria das aplicações com interatividade plena, com canal de retorno, utiliza linguagem procedural ou imperativa. Ainda de acordo com os autores [15], também é possível criar aplicações com interatividade através da linguagem declarativa.

Segundo Piske e Seidel [2], para o melhor entendimento sobre os recursos das ferramentas de autoria, este trabalho abordou os diversos recursos utilizados pelas ferramentas CASE, as funcionalidades oferecidas para o usuário e as linguagens utilizadas pelo gerador de código. As linguagens de programação são fatores fundamentais para a limitação dos recursos da aplicação.

Existem diversas ferramentas de autoria para TV Digital, e inúmeras outras para *Web* e para aplicações *Desktop*. Este capítulo faz uma análise de algumas dessas ferramentas, levantando suas principais características. As próximas seções começarão com ferramentas de autoria para aplicações *Desktop*, passando por ferramentas de autoria para aplicações da Internet, até chegarmos às ferramentas de autoria para aplicações de TV Digital. Esta sequência segue a evolução das ferramentas de autoria e suas influências entre si.

3.1.1 *Delphi*

Pugh e Doherty [9] comentam que em 1995, a Borland lançou uma ferramenta CASE proprietária para ambiente Windows que utilizava a linguagem Pascal como base para geração do código fonte. Nesta época o *Delphi* foi considerado a mais avançada ferramenta RAD para o sistema operacional Windows. O *Delphi* foi considerado o mais importante paradigma de programação desde a introdução da programação visual.

O *Delphi* era uma evolução do Turbo Pascal e disponibilizava o Pascal como *Object* Pascal. O *Object* Pascal combinava a linguagem Pascal com a Biblioteca de Componentes Visuais - VCL. O VCL disponibilizava aproximadamente 75 objetos ou componentes visuais e outros não visuais para acesso a banco de dados em sistemas cliente-servidor. A facilidade de construir aplicações gráficas com fácil acesso a banco de dados, aliada à fácil curva de aprendizado, foram fundamentais para a consolidação da importância das ferramentas CASE para aplicativos *Desktop*. O *Delphi* teve seus recursos evoluindo conforme suas versões lançadas, sendo que este trabalho utiliza o *Delphi* 7.0 como base.

O *Delphi* disponibiliza um rico ambiente gráfico também conhecido com IDE. Esta rica ferramenta gráfica de implementação de aplicativos para computadores pessoais inovou com um ambiente composto de várias janelas independentes, conforme a Figura 3.1 a seguir, diferentemente dos aplicativos MDI, que tinham uma única janela principal capaz de conter várias janelas secundárias, porém não desconexas, como por exemplo, as ferramentas *Office* da *Microsoft*. As diversas janelas do *Delphi* foram muito importantes para a padronização dos recursos oferecidos pelas ferramentas CASE.

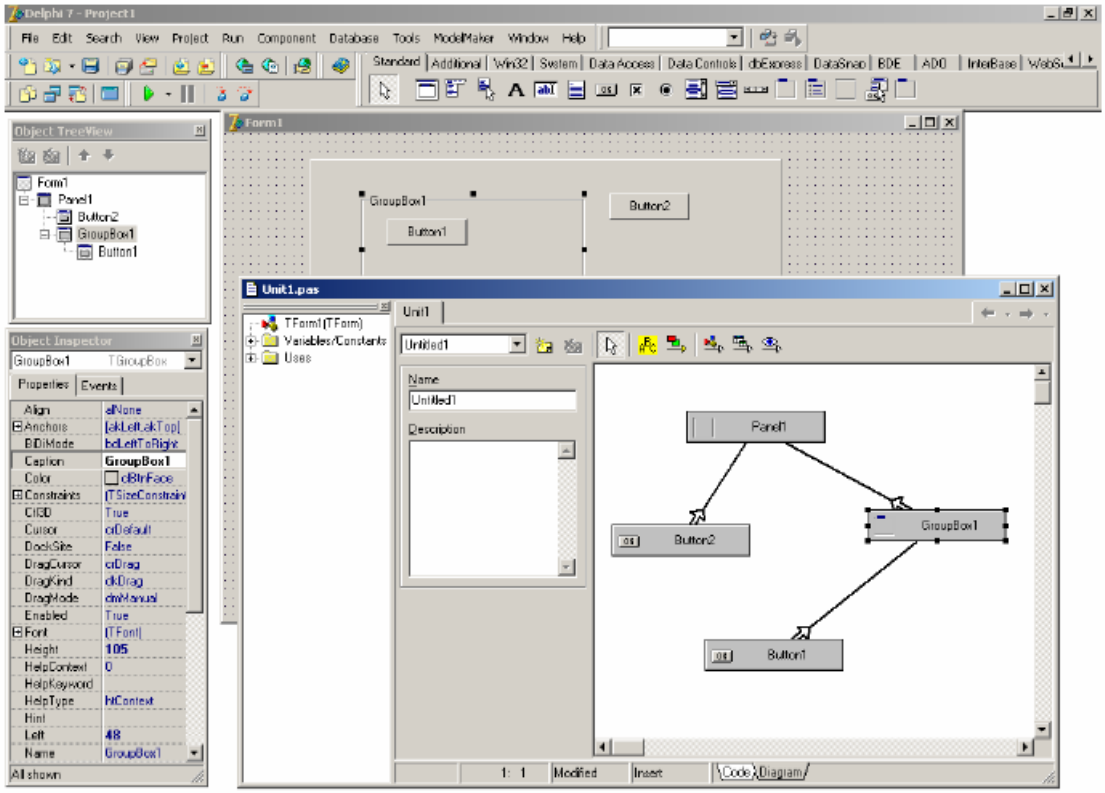


Figura 3.1 – Ambiente de janelas interativas
FONTE: Capturado do *software Delphi* 6.0

O *Delphi* apresenta sua janela principal na parte superior da tela como na maioria dos aplicativos e disponibiliza quase todos os recursos na forma de ícones. Estes recursos são a Barra de Menus, a Paleta de Componentes e a Barra de Ferramentas. Todas as janelas possibilitam ajuste de tamanho e podem ser minimizadas ou maximizadas. A janela de componentes é a parte mais utilizada pelo *Delphi*. Esta janela é também chamada de paleta e possui diversas abas com páginas que agrupam categorias de componentes. Estas páginas armazenam componentes gráficos que se tornaram padrão em aplicativos gráficos. A Barra de Ferramentas disponibiliza acesso rápido aos comandos mais comuns do *Delphi*, como criar formulários, abrir e salvar arquivos, ou compilar e executar aplicativos. Ela também pode ser configurada de várias formas, como, por exemplo, adicionar botões de comandos utilizados frequentemente, reordená-los ou retirá-los. A Barra de Menus é utilizada apenas para comandos pouco comuns como salvar ou compilar um projeto.

A Ferramenta *Delphi* foi a mais famosa ferramenta gráfica para geração de aplicações *desktop*, mas foi também importantíssima ao apresentar novos recursos para a interface e interatividade. Podemos citar o *Form*, formulário, como a plataforma base para construção de um aplicativo no *Delphi*.

Os formulários são as partes visíveis de um aplicativo e neles são inseridos componentes como botões, listas etc. Formulários podem ser usados de muitas maneiras diferentes em um aplicativo. Um formulário pode ser desde a janela principal, até uma caixa de mensagem.

O *Delphi* também possui um recurso de visualização dos componentes não gráficos utilizando um visualizador do tipo árvores, o *TreeView*. O *Delphi* 6 mudou para o *Object TreeView*. O *Object TreeView* apresenta todos os componentes e objetos do formulário na forma de árvore. Esta estrutura do tipo árvore mostra a relação pai/filho entre componentes e formulários. Esta técnica é bastante utilizada para a visualização de hipertextos.

O *Delphi* também trouxe a possibilidade de toda aplicação ser gerada a partir de um projeto. Também possibilita que o projeto seja visualizado de diversas formas. Na verdade, os modos de visualização são também os recursos apresentados pela IDE ou ambiente gráfico. O modo de visualização padrão é chamado de *Diagram*.

Como toda a ferramenta CASE de geração de aplicativos baseados em uma linguagem de programação, o Delphi disponibiliza também um editor de código para acesso a

autores mais experientes ou com bom conhecimento da linguagem de programação. O *Delphi* 4 também adicionou um novo recurso de navegação rápida para as unidades de códigos.

Como os componentes gráficos são os principais elementos de uma aplicação e devido ao número elevado de parâmetros para suas configurações, o *Delphi* disponibilizou desde sua primeira versão uma ferramenta importantíssima para conhecimento da capacidade dos componentes, que é o *Object Inspector*. O *Object Inspector* é utilizado para alterar as propriedades e os eventos de cada componente associado. A programação orientada a eventos teve seu impulso bastante influenciado pelo uso do *Delphi*.

Além das ferramentas para manipulação dos componentes, o *Delphi* também oferece recursos para manipulação de sua própria ferramenta CASE, como o recurso *Desktop*. Este recurso permite organizar as janelas da maneira ideal para o trabalho. Outro recurso importante para implementação dos projetos em *Delphi* é o TO-DO List, que cria uma lista de tarefas agendadas. Para apoio a programadores experientes, o *Delphi* também disponibilizou ainda uma ferramenta para listar a relação hierárquica das classes contidas em seu projeto. As teclas de atalho também foram bastante utilizadas pelos programadores.

3.1.2 *Visual Basic*

A *Microsoft*, apesar de ter seu grande foco comercial voltado para os usuários, teve o *Visual Basic* como grande concorrente do *Delphi*. O *Visual Basic* tem a grande vantagem de poder explorar os principais recursos do Windows, pois é também um produto *Microsoft*. Esta ferramenta também foi muito importante para solidificar a importância das ferramentas CASE no mercado. Muitos dos recursos apresentados pelo *Delphi* foram também utilizados. Na verdade, o *Delphi* e o *Visual Basic* se apoiavam nos conceitos da programação RAD e são muitíssimos parecidos com pequenas diferenças nos nomes dos componentes e uso de linguagem de programação.

O *Visual Basic* tinha como linguagem base o *Basic*. Umas das poucas diferenças em relação ao *Delphi* era a forma como as ferramentas geravam o aplicativo. O *Delphi* gerava um único arquivo executável e o *Visual Basic* gerava um programa principal e suas DLLs. O *Visual Basic* adicionou ainda uma importante ferramenta, os *templates*. O *template* é uma ferramenta que traz o conceito de reusabilidade de código e principalmente de reusabilidade

de leiaute. Esta ferramenta pode ajudar muito um usuário leigo que não tenha noção de interface. Hoje, a maioria das ferramentas de autoria traz esse recurso.

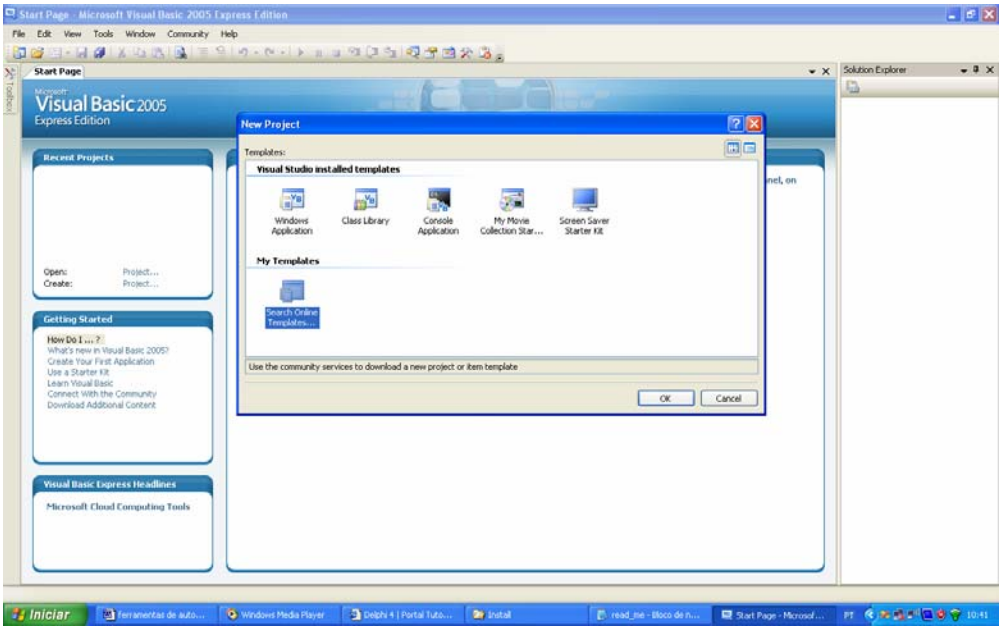


Figura 3.2 – Tela do *Visual Basic*
FONTE: Tela capturada do *software Visual Basic*, 2005

3.1.3 *NetBeans*

As ferramentas de apoio à fase de modelagem e desenvolvimento são geralmente muito dispendiosas e com custos consideráveis. Este problema tem sido equacionado através do uso de ferramentas *open-source* ou *software* livre. O *NetBeans* é uma ferramenta desenvolvida pela *Sun Microsystems* para desenvolvimento de aplicações Java. A linguagem Java é uma linguagem de *software* livre e hoje uma das mais utilizadas para desenvolvimento de aplicações, tanto para *Desktop* como para *Web* e PDAs, conforme janela de criação do projeto na Figura 3.3.

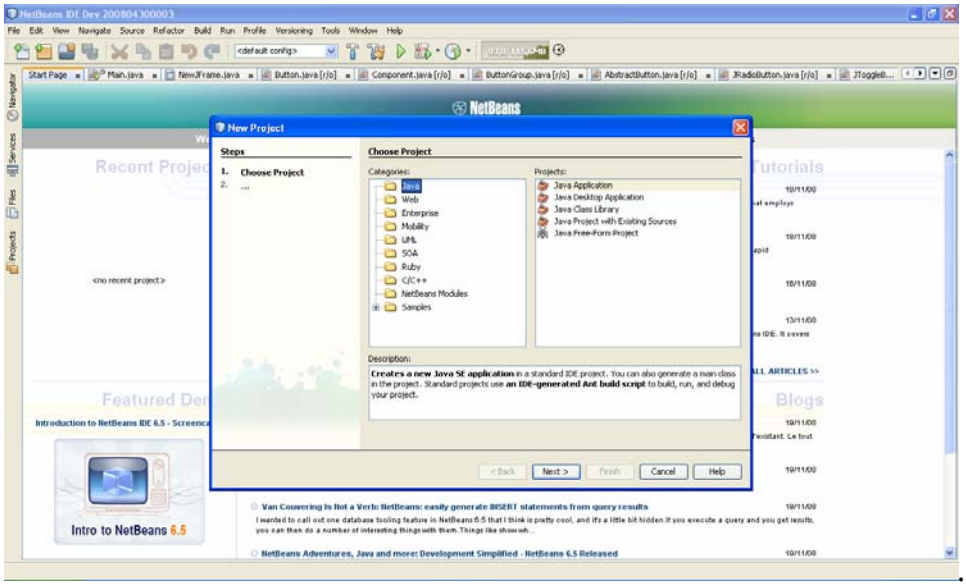


Figura 3.3 – Tela do *NetBeans*
FONTE: Tela capturada do *software NetBeans*.

A plataforma *NetBeans* é um amplo *framework* baseado na API *Swing*. A plataforma contém APIs que simplificam a manipulação de janelas, ações, arquivos e muitos outros itens típicos para aplicativos. Esta ferramenta CASE possibilita a adição de recursos distintos para o aplicativo ser gerado através de módulos conhecidos como *plugins*. Um módulo *plugin* é um grupo de classes Java que fornece um recurso específico a um aplicativo. Um *plugin* é a menor parte de uma plataforma e os *plugins* podem ser desenvolvidos e distribuídos separadamente. Normalmente um conjunto de recursos e funcionalidades são desenvolvidos como um *plugin*. O *NetBeans* também possibilita criar módulos que disponibilizem suas tecnologias de última geração [30].

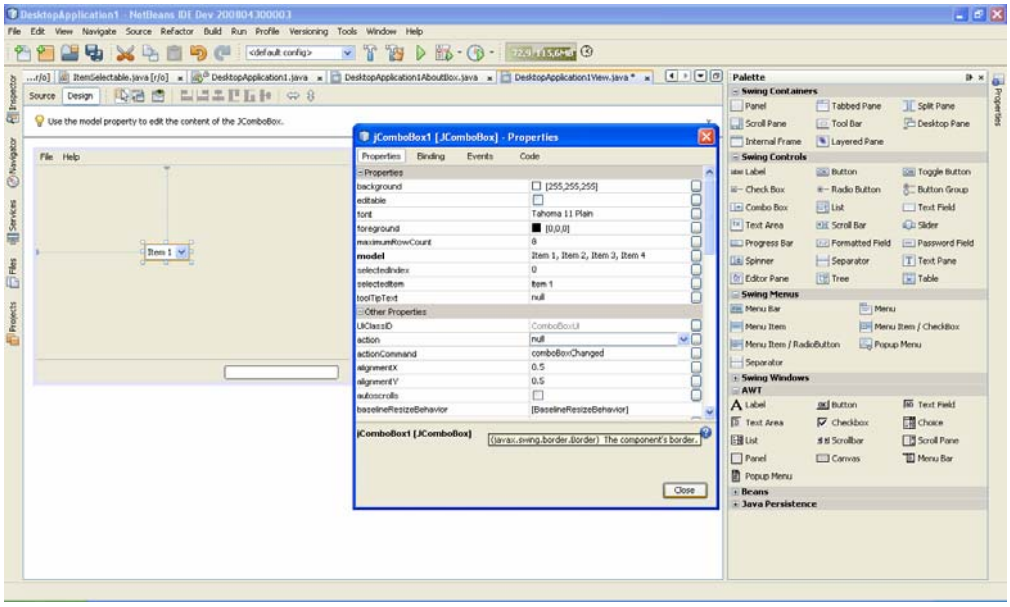


Figura 3.4 – Tela do *NetBeans*
FONTE: Tela capturada do *software NetBeans*.

O *NetBeans* apresenta muitas das funcionalidades utilizadas tanto no *Delphi* como no *Visual Basic*. Na Figura 3.4, observa-se a disponibilidade dos componentes gráficos e suas propriedades associadas e visualizadas em janela auxiliar. O *NetBeans*, diferentemente do Eclipse, prioriza a disponibilização das ferramentas gráficas, apesar de ser mais pesado que o Eclipse para o computador, ou seja, exigindo mais memória.

3.1.4. Eclipse

O Eclipse é uma ferramenta de autoria desenvolvida pela IBM para geração de aplicações Java inicialmente para aplicações *Desktop* e *Web*. Esta ferramenta tem característica de ser mais leve que o *NetBeans* e possibilita, também, através de instalação de *plugin* a instalação de módulos. Com exceção do código principal, todas as funcionalidades são implementadas por *plugins*.

O Eclipse é uma IDE inicialmente projetada para desenvolvimento em Java e conta hoje com poderosas e inúmeras modificações e *plugins* para integrar ferramentas de desenvolvimento em inúmeras linguagens, como PHP, *Rails*, C/C++, SQL, *iPhone*, todas as linguagens *Web* e até algumas mais remotas, como COBOL e, ultimamente, a linguagem Lua e o NCL. Por isto, tem se tornado indispensável para programadores, alguns *webdesigners* e ultimamente para programadores para aplicações de TV Digital.

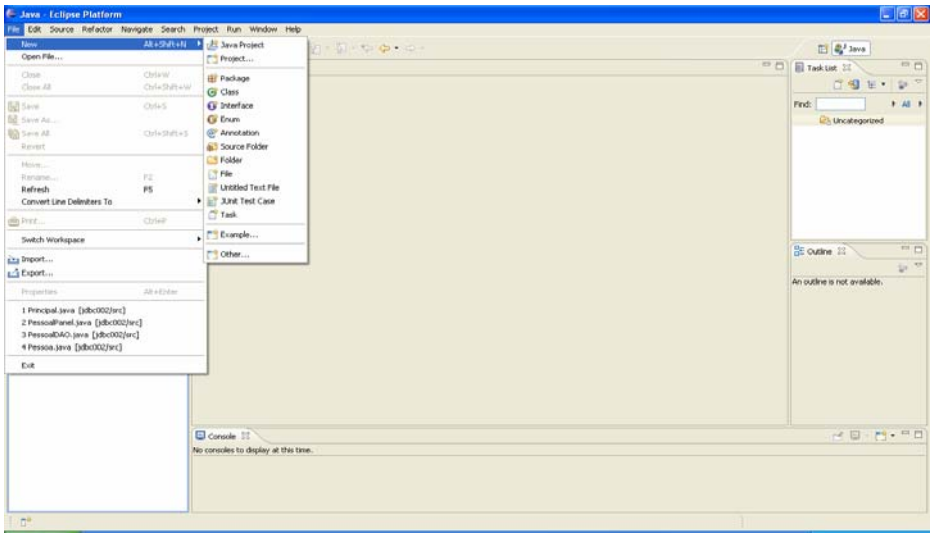


Figura 3.5 – Tela do Eclipse
FONTE: Tela capturada do *software* Eclipse.

Para se ter uma idéia da importância das ferramentas de autoria para aplicações *desktop*, a Universidade Federal do Maranhão [31] lançou o NCL Eclipse. O NCL Eclipse aproveita os princípios da própria ferramenta de autoria Eclipse para gerar produtividade, facilidade e integração. O NCL Eclipse tem algumas características como: coloração sintática das *tags*, atributos e comentários XML, *Outline View* possibilitando navegar no documento NCL, sugestão de código ou *autocomplete* de forma dinâmica e contextual, segundo a Norma ABNT NBR 15606-2:200. Valida automaticamente documentos NCL e marca o erro no documento, facilitando a sua localização, mecanismo para esconder/revelar nós XML (*folding*), formatação automática de código XML e execução do documento NCL usando o GINGA NCL *Emulator*.

Outra novidade anterior ao NCL Eclipse é o LuaEclipse [32]. O LuaEclipse é uma coleção de *plugins* desenvolvidos para o Eclipse, que juntos formam um IDE para o desenvolvimento de aplicações na linguagem de programação Lua. Nesse ambiente, é possível editar Lua *scripts* com sintaxes destacadas, complementação automática de código, compilação de erros, execução de *script* pré-configurado, além das ferramentas de que dispõe a plataforma Eclipse. O principal objetivo do LuaEclipse é que as novas ferramentas possam ser desenvolvidas a partir da extensão da arquitetura que fornece plataforma Eclipse e LuaEclipse, possibilitando a ampliação de suas capacidades.

Aplicações para TV Digital têm sido produzidas para o padrão Europeu com a utilização do Eclipse e através da API proprietária *Multimídia Home Plataform* - MHP. Esta

aplicações podem ser compiladas e emuladas pelo *XletView* [33] na Figura 3.6 a seguir, que é uma ferramenta de *software* livre com licença GNU.

Figura 3.6 – Tela inicial do *XletView*
FONTE: CARVALHO; ARAÚJO, 2007, p. 5. [68]

3.1.5 *Dreamweaver* e *FrontPage*

A Internet e seus aplicativos surgiram como um paradigma do uso de hipertextos. O HTML, Linguagem de Marcação de Hipertextos, apareceu como uma linguagem declarativa e uma nova alternativa para autores sem a necessidade do mesmo conhecimento básico que os programadores de linguagens procedurais necessitam para produzir aplicativos. As ferramentas de autoria para aplicativos para Internet também sofreram influência dos recursos visuais e de programação explorados pelo *Delphi* e *Visual Basic* e acrescentaram novos recursos para *link* das mídias, trazendo novas alternativas para arquivos multimídia.

Pode-se observar que, na Figura 3.7, os recursos de ferramentas de componentes gráficos, barras de menus e outras ferramentas de apoio ao desenvolvimento, são muito parecidos com os disponibilizados no *Delphi*.

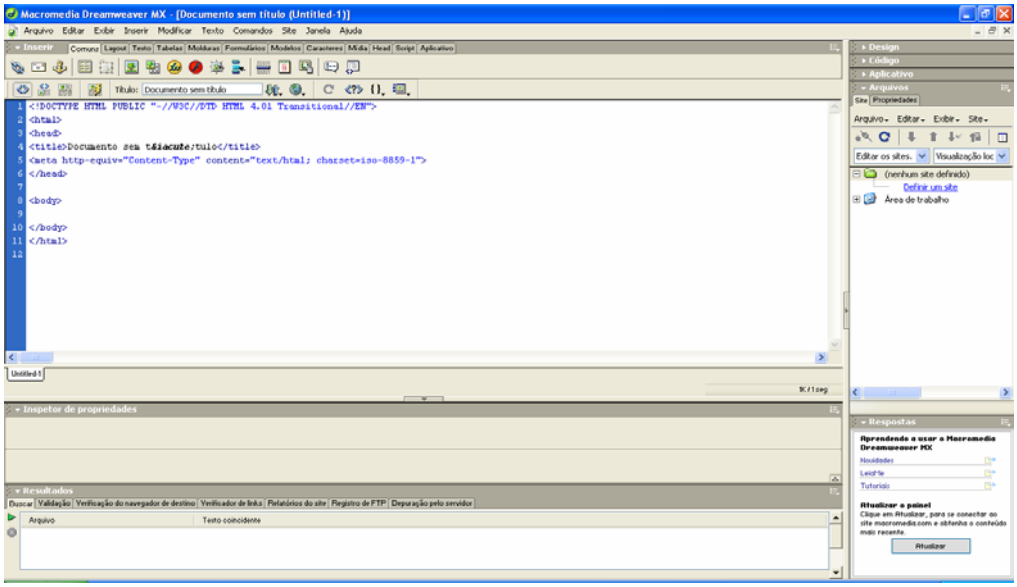


Figura 3.7 – Ferramentas de componentes
FONTE: Tela capturada do *software* Dreameweaver.

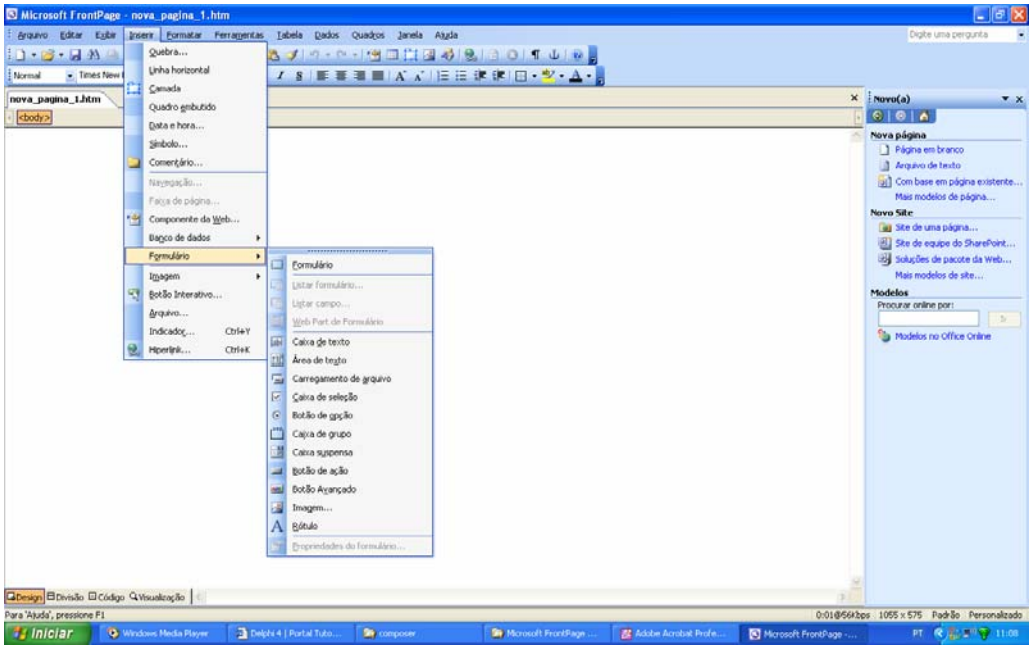


Figura 3.8 – Ferramentas de componentes
FONTE: Tela capturada do *software* FrontPage.

A utilização de *templates* foi uma das grandes novidades apresentadas pelas ferramentas de autoria para aplicações *Web*.

3.1.6 *Macromedia Flash*

A influência da Internet é tão grande que a produção de objetos de mídia para animação passou a ser bastante utilizada nas aplicações com interatividade. Objetos animados são produtos bastante utilizados em televisão e cinema. A empresa Adobe resolveu desenvolver uma ferramenta de autoria para implementação de aplicativos com animação chamada *Flash*. Esta ferramenta atinge hoje mais de 98% dos usuários na Internet [23]. O *Flash* utiliza a linguagem *ActionScript*, que é uma variação do *ECMAScript* [24]

A ferramenta de autoria *Macromedia Flash* é composta por uma IDE e por um emulador ou *player*. Esse emulador é usado para depurar e apresentar as aplicações criadas. O *Flash* possui a máquina virtual *ActionScript Virtual Machine* - AVM -, que processa os *scripts* gerados.

O *Flash* procura ser o mais intuitivo possível, apesar da animação exigir o conhecimento de técnicas muito específicas e pouco conhecidas por um leigo. Ao entrar nesta ferramenta, uma janela é mostrada sugerindo um tutorial que ajuda a personalizar seu aprendizado. O *Flash* distingue o usuário *designer* do desenvolvedor que conhece a linguagem *ActionScript*.

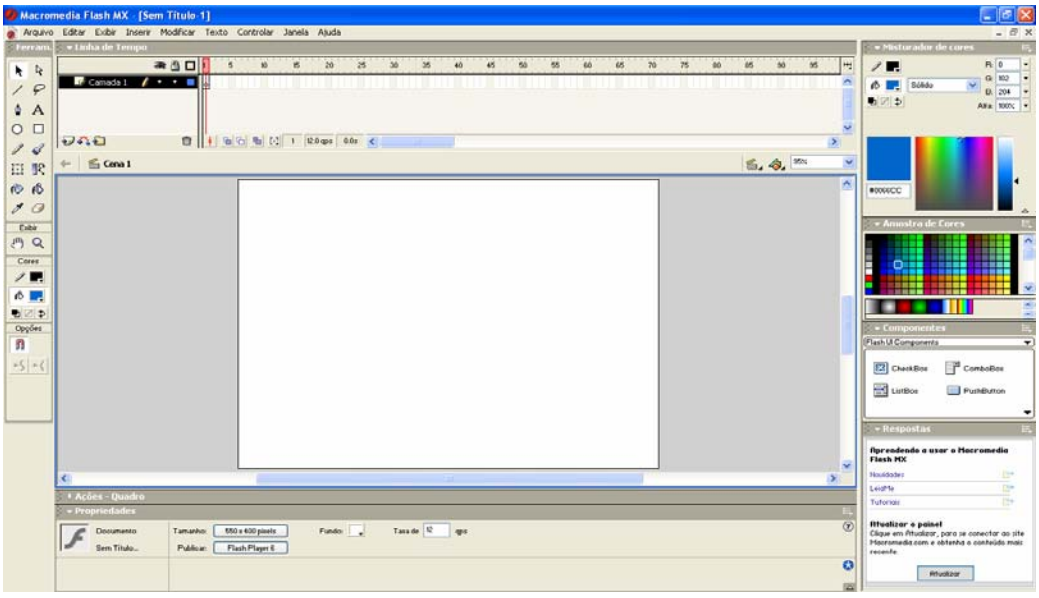


Figura 3.9 – Tela inicial do *Flash*
FONTE: Tela capturada do *software* Flash.

Na Figura 3.9 é possível observar as diversas regiões ou janelas com os principais recursos para implementação. O *Flash* é dividido em oito regiões: Barra de Ferramentas,

Linha do tempo, Misturadores de Cores, Amostra de cores, Componentes, Respostas, Ações – Quadro e Propriedades. As principais regiões que servem de apoio no desenvolvimento das aplicações *flash* são a Barra de Ferramentas, a Linha do Tempo e a região da Cena ou *Stage*.

Na região Barra de Ferramentas estão os recursos básicos de desenho e animação. Na janela ou região Cena ou *Stage* está o espaço onde são realizadas as tarefas de edição de gráficos de forma muito parecida com um papel onde são feito os desenhos.

A Linha do Tempo é a região onde os elementos desenhados na Cena são organizados. Ela é composta de duas partes, uma para organizar as camadas e outra para organizar os fotogramas ou quadros. As camadas são dispostas em linhas e os fotogramas ou quadros são dispostos em colunas. Nos quadros ou fotogramas podem existir três tipos de imagens: imagens-chave, que o próprio autor desenha na região da cena, imagens-fixas, copiadas nos quadros seguintes ao da primeira imagem-chave, e imagens de interpolação, que são criadas pelo programa e permitem a transição gradual entre duas imagens-chave. A região Linha do Tempo, junto com a região Cena, é responsável pela visão espaço/temporal. A especificação dos relacionamentos espaço/temporais somente é permitida através da edição dos *scripts*, dificultando o uso por usuário leigo em animação, sem conhecimento prévio desta ferramenta de autoria.

A região Componente é onde os objetos gráficos como botões e caixas de texto são utilizados para prover interatividade. A região Ações – Quadro é a região que mais provê funcionalidades, pois oferece recursos para edição do código *ActionScript*.

3.1.7 GRiNS

O sincronismo temporal para aplicações hipermídia foi inicialmente disponibilizado com o desenvolvimento da linguagem declarativa SMIL pela W3C. A *GRaphical INterface for creating and playing SMIL documents* - GRiNS - foi uma das primeiras ferramentas de autoria para facilitar a implementação de aplicações *Web* que necessitassem utilizar a SMIL. A SMIL, Linguagem de Integração de Multimídia Sincronizada possibilita o sincronismo temporal de mídias [18]. O GRiNS tem a implementação de aplicações com sincronismo como seu principal recurso. Bulterman et al. [20] explica que o GRiNS é um ambiente de autoria que trabalha com o conceito de visões temporais, espaciais e textuais integradas.

Na visão temporal o autor pode montar uma aplicação multimídia com base em um *timeline*, linha de tempo. A ABNT-NBR 15606-2:2008 [19] coloca que as aplicações multimídia são aquelas que manipulam objetos do tipo áudio, vídeo, imagem, texto e aplicações. Na visão espacial o GRiNS disponibiliza uma espécie de *container* com as regiões onde os objetos de mídia serão apresentados, ou seja, o leiaute dos objetos de mídia. A visão espacial talvez seja a mais importante, pois apresenta a árvore hierárquica das regiões definidas onde os objetos de mídias serão exibidos. Na visão textual esta ferramenta oferece recursos básicos para edição do código SMIL através de um pequeno editor. As alterações feitas no código através de editor são refletidas na aplicação através do botão *apply*. Este editor de texto possui poucos recursos de edição. A existência de várias visões permite ao autor optar pela melhor forma de desenvolvimento da aplicação e da maneira que mais lhe convém.

O GRiNS privilegia o sincronismo de mídias. Este recurso é muito fácil de ser visualizado pelo seu editor ou visão textual, pois os recursos de sincronismo são claramente descritos na linguagem SMIL. O GRiNS não permite a edição ao vivo de documentos devido à limitação da linguagem SMIL. O GRiNS também oferece um emulador ou *player* que servirá de visualizador da aplicação e testes.

3.1.7 *LimSee2*

O *LimSee2* [21] é uma ferramenta de autoria utilizando a linguagem declarativa SMIL desenvolvida pelo grupo *Web Adaptation and Multimedia* – WAM - do Instituto *National de Recherche en Informatique* - INRIA. O *LimSee2*, da mesma forma que o GRINS, é um ambiente de autoria com múltiplas visões integradas. Estas visões são um pouco mais fáceis de serem visualizadas do que no GRINS, pois são prontamente disponibilizadas em cinco janelas para visão estrutural, espacial, temporal, de atributos e textual.

Na visão estrutural o LimSee2 reflete a estrutura de árvore que o SMIL apresenta como um código de padrão XML. A visão espacial disponibiliza as regiões onde os objetos de mídias serão alocados. Na visão temporal os objetos estão representados na forma de retângulos onde seus comprimentos estão relacionados com o tempo de duração. Os objetos de mídias podem ser redimensionados na linha do tempo. Na visão de atributos, os objetos selecionados na visão temporal têm suas propriedades disponibilizadas para alteração. Por

fim, na visão textual o *LimSee2* disponibiliza também um editor de texto para visualização do código SMIL.

O *LimSee2* é uma ferramenta de autoria que não tem como principal foco a interatividade e seus recursos são limitados pela própria linguagem SMIL.

3.1.8 JAME AUTHOR

O JAME AUTHOR [12] é uma ferramenta de autoria voltada para aplicativos para TV Digital tendo como base a linguagem Java para o *middleware* MHP/OCAP. Esta ferramenta faz parte de um grupo de desenvolvimento de soluções para TV Digital chamado *Interactive* TV, do *Fraunhofer Institute for Media Communication* - IMK. Esta ferramenta de autoria possibilita ao usuário não codificar uma linha da aplicação e trabalha com editor WYSIWYG e com ambiente emulador para testar a aplicação. Pode ser observado que a criação de documentos hipermídia nessa ferramenta de autoria é parecida com a criação de documentos HTML, o que vem a facilitar bastante sua utilização por usuários não programadores em Java ou *Webdesigners*.

Esta ferramenta CASE, além de ter absorvido inúmeros recursos de interface e gerenciamento do projeto das diversas ferramentas de autoria para *desktop* e Internet, induz o usuário a conhecer os padrões de usabilidade da TV Digital. Observa-se na Figura 3.10 que o projeto começa com a definição da resolução da aplicação.

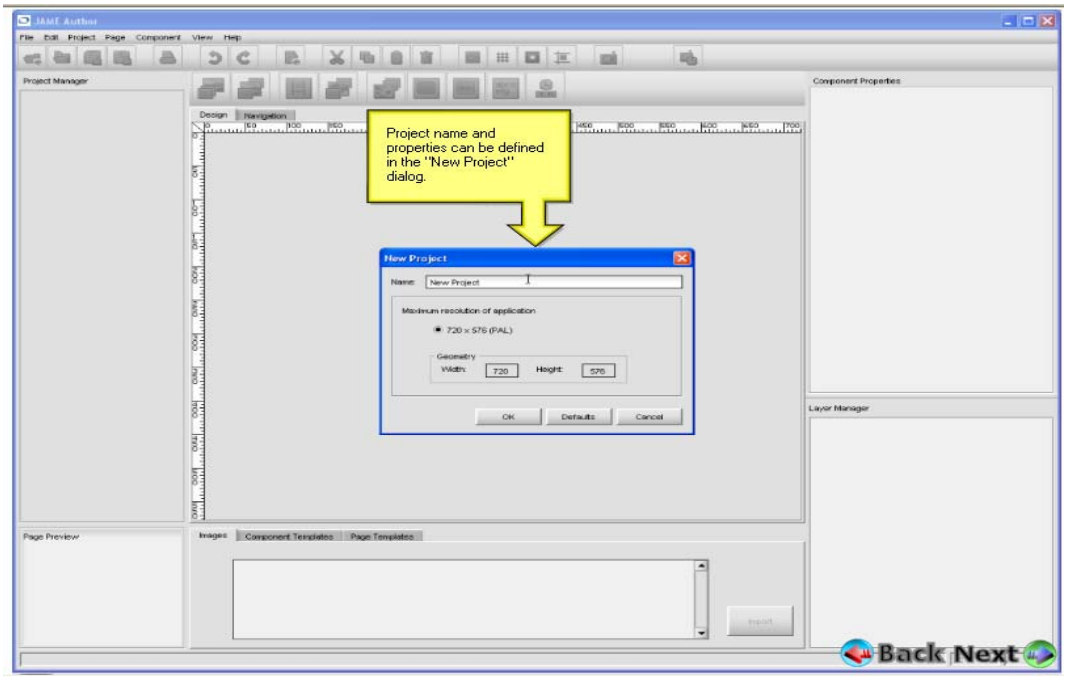


Figura 3.10 – Definição da resolução da aplicação
FONTE: JAME, 2008. [12]

As principais funcionalidades do JAME AUTHOR são: criar projetos e adicionar páginas, adicionar e modificar componentes, definir navegação, criar foco de transferência entre os componentes e testar a aplicação em emulador.

As opções de criar projetos e adicionar páginas trabalham basicamente com a construção de páginas como base para inclusão de componentes utilizando técnicas de hipertexto semelhante ao que é utilizado nas páginas *Web*. Este sistema de páginas possui três tipos de eventos: *Start*, *Error* e *Loading*. Estes eventos são, na verdade, outras páginas que poderão ser adicionadas posteriormente. Um projeto no JAME AUTHOR possui páginas de sistema pré-definidas para lidar com tratamento de erros.

A opção de adicionar e modificar componentes possibilita que o autor da futura aplicação insira componentes gráficos através da técnica *Drag and Drop*. Esta técnica é uma das principais novidades do *Delphi* e *Visual Basic*. Os componentes têm suas propriedades visualizadas através de uma janela de apoio ao projeto. Cada componente pode ter diferentes estados de visualização, que correspondem a estar ou não em foco, estar habilitado ou não, visualizado ou não e pressionado ou não. A propriedade *backgroundColor* serve como atributo que representa quando o componente está em foco. Os componentes têm suas propriedades divididas em modelo *default*, *focused*, *pressed* e *disabled*. Estas classificações resumem os possíveis estados dos componentes gráficos de interação.

Para o JAME AUTHOR a configuração *default* de um componente é aquela definida pela própria ferramenta como padrão e que pode ser utilizada sem maiores conhecimentos do usuário. A configuração *focused* é utilizada com a mudança de cores de todos os sub-componentes como textos, cores das bordas, etc., quando o componente estiver em foco. A opção *pressed* oferece também mudança de cores para os sub-componentes no caso de ter o telespectador escolhido o componente com o "*enter*" do controle remoto. Por último, o *disabled*, que é um estado possível para um componente gráfico significando a condição de desabilitado.

O JAME AUTHOR também oferece uma opção de definir navegação, conceito de hipertextos, que obriga o programador a escolher sua página inicial. Para possibilitar a navegação entre as páginas esta ferramenta oferece uma caixa de diálogo onde se escolhe a página destino, o componente que iniciará o processo de mudança de página e o primeiro componente a apresentar o foco na página destino.

A opção criar foco de transferência entre componentes é uma das mais importantes ferramentas para a interatividade nas aplicações de TV Digital. O JAME AUTHOR também utiliza, conforme a Figura 3.11, uma caixa de diálogo solicitando os componentes origem e destino do foco e associando a tecla de seta do controle remoto. O controle de foco é a parte mais complicada de se implementar nos *frameworks*, pois a classe que controla o foco precisa mandar mensagem para o componente e avisar para que ele receba ou libere o foco para o próximo componente.

Como na maioria das ferramentas CASE, o JAME AUTHOR possui regiões ou janelas que englobam diferentes funcionalidades e que dão suporte ao desenvolvimento de todo o projeto. O *Project Manager* é uma região onde são listadas as páginas de um aplicativo. O *Page Preview* é a região de pré-visualização de uma página selecionada no *Project Manager*. O *Property Editor* mostra as propriedades quando um componente de uma página aberta na *Work Area* é selecionado, possibilitando também que estes atributos sejam modificados. A janela *Layer Manager* lista as principais camadas suportadas (*Graphic*, *Vídeo* e *Background Layer*). Importante lembrar que os recursos disponibilizados por esta ferramenta de autoria devem estar de acordo com o MHP.

O *Project Manager* como ferramenta central disponibiliza dois modos de edição que são o de *design* e o de navegação. O modo *Design* foi projetado para dar os principais recursos de edição dos componentes e até utilizar *templates* de componentes feitos em Java. O

modo de navegação é utilizado para definição da estrutura navegacional entre componentes e entre páginas.

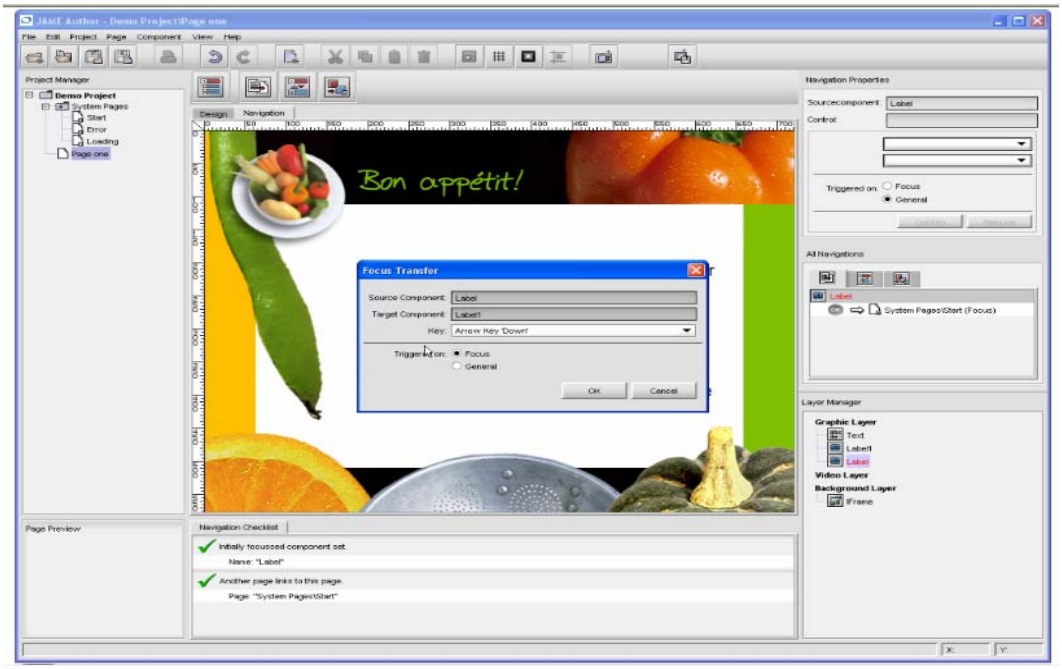


Figura 3.11 – Caixa de diálogo da aplicação
FONTE: JAME, 2008. [12]

Também como apoio ao projeto, o JAME AUTHOR disponibiliza um emulador ou simulador que se integra ao MHP/Java e permite que o autor execute e valide seu trabalho. O JAME AUTHOR como uma ferramenta de autoria para aplicações de TV Digital, não oferece suporte a sincronismo espacial/temporal algum e serve principalmente como um gerador de código para linguagem procedural. Esta ferramenta disponibiliza recursos para implementação de aplicativos com recursos de interatividade com relacionamentos de referência entre documentos e a mudança de foco entre componentes.

3.1.9 *Cardinal Studio*

O *Cardinal Studio* [14] é uma ferramenta de autoria para criação de aplicativos de TV Digital interativa com uso da linguagem Java e *Middleware* MHP. Esta ferramenta é voltada para atender programadores e profissionais não programadores que trabalham nas emissoras de televisão. O *Cardinal* propõe atender a pessoas com pouca experiência em

programação e inclui muitos recursos para profissionais de *design* gráfico. Ele ainda possibilita que o usuário inclua também parte de seu código à aplicação.

O *Cardinal* traz um recurso que tem sido bastante utilizado nas ferramentas de autoria e principalmente para aplicações WEB, como os *templates*. Este *software* disponibiliza modelos de *templates* e também possibilita o uso de *templates* implementados pelo próprio usuário. O *Cardinal* também possui um emulador para testes da aplicação criada. O *Cardinal Professional* 4.0 traz a novidade de *design* em camadas, com o objetivo de diminuir o tamanho da aplicação e consequentemente menos consumo de memória. Esta versão também disponibiliza um guia SDK para possibilitar ao programador a criação de novos componentes.

O *Cardinal* tem como modelo de abstração o uso de "*Acts*", atos ou cenas, onde componentes de interação são adicionados. Ele também acrescenta componentes "invisíveis" aos recursos da aplicação. Os componentes "invisíveis" ou não gráficos são aqueles que, além de não serem desenhados na tela, servem fundamentalmente como estruturas auxiliares. Estes componentes são na maioria das vezes utilizados para o canal de retorno, conexão http, botões de controle remoto, *stream event* e outros. Os componentes visíveis ou gráficos são os mais comuns e são utilizados para interatividade com entrada e saída de dados em painéis, áreas de texto e botões.

No modelo de atos, o *Cardinal* tem os componentes gráficos e não gráficos estruturados em camadas. As camadas podem ser compartilhadas entre atos. Os componentes são inseridos em uma espécie de *container*. As camadas podem ser divididas em camadas invisível, gráfica e de vídeo. A camada de vídeo tem como propriedade o *Vídeo Source*. Se esta propriedade estiver nula, a aplicação estará recebendo como vídeo principal o da operadora do canal, caso contrário a aplicação terá um vídeo próprio agregado.

A interatividade no *Cardinal* é definida através de eventos. Estes eventos são disparados quando algum componente tem o foco ou algum botão do controle remoto é acionado. Os componentes têm seus eventos descritos na propriedade *actionPerformed*.

Para apoio ao desenvolvimento do projeto o *Cardinal* disponibiliza regiões com funcionalidades específicas. Estas regiões são: Eventos, Canvas, Propriedades de Componentes e Repositório de Componentes. A região de Eventos tem um "*timeline*" ou uma espécie de linha de tempo que serve para associar os componentes às ações do usuário da aplicação a ser construída. A região Canvas é utilizada para definir o leiaute dos componentes

gráficos na tela. A região de Propriedades serve para mostrar os atributos possíveis e modificáveis, tanto para componentes visíveis como para invisíveis. A região de Repositório serve para disponibilizar os ícones de todos os componentes do *Cardinal*.

Como todo aplicativo de TVDI tem seu tratamento de navegação e foco como processo fundamental, o *Cardinal* configura automaticamente a navegação de acordo com a ordem de adição dos componentes gráficos focáveis. A mudança de ordem de foco entre componentes é possível através do modo de navegação e todas as teclas do controle remoto podem ser associadas à mudança de foco.

O *Cardinal* é uma ferramenta voltada para implementar código Java no padrão MHP voltado apenas para prover recursos de interatividade sem sincronismo temporal/espacial, mas possibilita este tratamento através de codificação adicional. Ele também não oferece recursos diretos para edição ao vivo de documentos, mas possibilita a especificação de *stream events*. Por fim, é também importante destacar que esta ferramenta de autoria permite testar a aplicação através de um emulador no padrão MHP.

3.1.10 *AltiComposer*

Em 2001 a Alticast lançou sua ferramenta de autoria para TV Digital. O *AltiComposer* suporta linguagem Java para *middleware* no padrão DVB-MHP. Esta ferramenta também foi desenvolvida para dar suporte para autores sem especialização em programação. Os usuários avançados têm nesta ferramenta o recurso de poder criar seus próprios componentes estendendo uma API chamada Kit de Desenvolvimento de Componentes - CDK [22].

O *AltiComposer* resolveu definir um modelo de produção próprio e baseado no perfil de produção de conteúdo da indústria de TV e cinema. Este modelo se baseia em cenas. Estas cenas são divididas em planos e estes planos são compostos de *shots*, cada um com vários atores, conforme a Figura 3.12 a seguir.

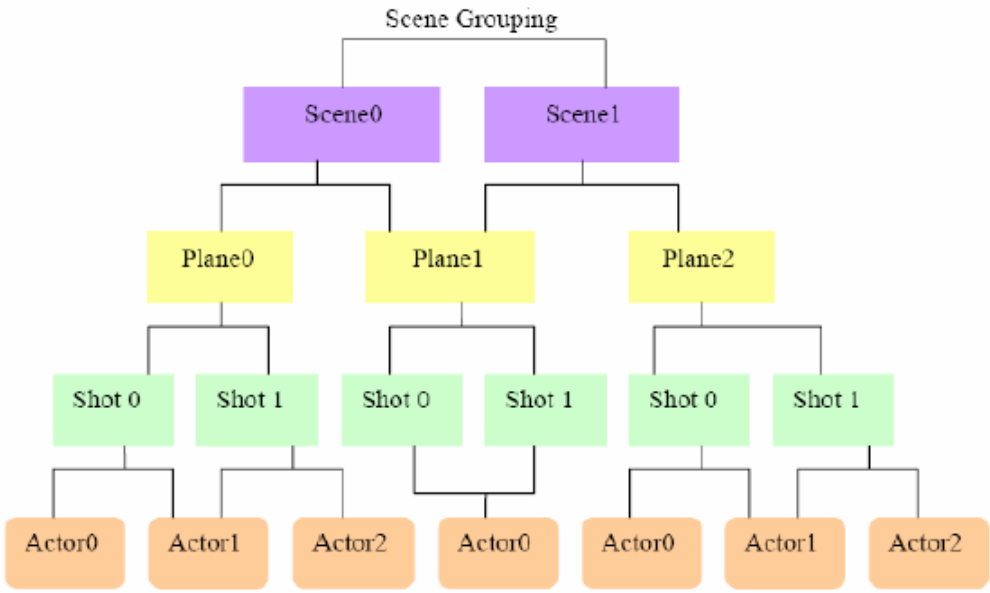


Figura 3.12 – Planos *Alticomposer*
FONTE: GUIMARÃES; COSTA, 2006, p. 47. [69]

No *AltiComposer*, um ator é um objeto Java com comportamento próprio. Este objeto Java serve para responder a eventos da interação do usuário com o controle remoto. Estes atores também são divididos em visuais e não visuais, da mesma forma que nas ferramentas de autorias anteriores. O *Shot* é a unidade ou *container* responsável pela configuração da navegação. O Plano é a unidade básica para salvar e carregar o conteúdo para a apresentação. Os atores, *shots* e planos compõem a cena do aplicativo.

Como estrutura de apoio à implementação das cenas, o *AltiComposer* apresenta cinco regiões ou janelas como: *Project Window*, *Library Window*, *Stage*, *Property* e *Short Transition*. A *Project Window* ou Janela do Projeto é dividida em três partes: a mais à esquerda apresenta a relação hierárquica entre cenas e planos. Na parte inferior à direita são listados os *Shots* que compõem cada plano da divisão anterior e na parte superior, também à direita, estão os atores que fazem parte dos *Shots*. A *Library Window* ou Janela da Biblioteca é o repositório de todos os recursos que podem ser utilizados pelo *AltiComposer*. A região ou janela *Stage* é a área de trabalho onde os componentes escolhidos na *Library Window* são alocados, ou seja, o leiaute dos componentes na tela. A *Property* é a janela de propriedade dos componentes selecionados na *Stage*. A *Short Transition* representa a janela de transição ou navegação entre os *Shots*.

O *AltiComposer* não foi projetado para implementação de sincronismo temporal/espacial. Esta ferramenta também não possibilita edição ao vivo de documentos e não possui emulador para testes da aplicação gerada.

3.1.11 *Composer*

O *Composer* é uma nova versão melhorada do *HyperProp* [25]. Esta ferramenta de autoria para documentos hipermídia também adota o conceito de visão para implementação dos seus aplicativos utilizando a linguagem Java para gerar aplicativos em linguagem declarativa NCL, que é baseada no modelo conceitual NCM [28]. O *Composer* possibilita aos atores que editem hiperdocumentos especificando componentes com suas propriedades, links internos e links externos, além de também especificar as características de cada componente.

O *Composer* [1] pode ser considerado a primeira ferramenta de autoria para o *middleware* do sistema brasileiro de TV Digital, o SBTVD. Esta linguagem foi desenvolvida pela PUC-RJ para possibilitar a elaboração de documentos hipermídia. Outro importante fator de valorização do *Composer* é ser ele uma ferramenta de *software* livre. Esta característica tem ajudado muito, hoje em dia, a evolução das ferramentas de *software*.

O *Composer* tem como forte característica a sincronização das visões e para isto utiliza a API Java NCM ao invés de utilizar diretamente a API DOM. O modelo *Document Object Model* - DOM - é utilizado para armazenar e manipular árvores XML. Os arquivos XML são compostos de *tags* organizadas hierarquicamente. Para cada visão, existe um par de compiladores responsáveis por traduzir a representação de um documento NCL no modelo Java NCM. Quando o modelo Java NCM é modificado, objetos observadores são notificados e acionam os compiladores específicos de cada visão para, a partir do modelo Java NCM, gerar um novo modelo para cada uma das visões.

O *Composer* teve sua interface gráfica avaliada por uma equipe de Interface-Homem Computador - IHC. Esta ferramenta de autoria visa ajudar o autor no desenvolvimento de aplicações hipermídia voltadas para TV Digital abstraindo toda, ou pelo menos parte da complexidade de se programar em NCL. Para isso, além da abstração de visões, o *Composer* diponibiliza outros mecanismos de suporte ao usuário, como por exemplo, mecanismos de ajuda e mensagens de erro.

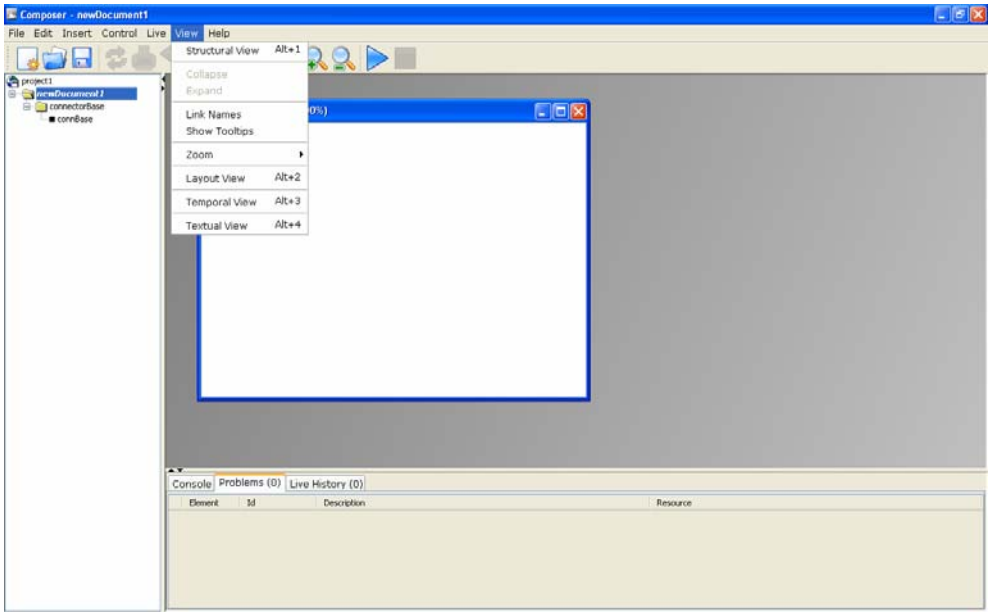


Figura 3.13 – Mecanismos de ajuda e mensagens de erro
FONTE: Tela capturada do *software* Composer.

O NCL, em comparação a outra linguagem declarativa de sincronismo como o SMIL é uma evolução trazendo novas funcionalidades, inclusive sincronismo espacial [29]. Como o foco desta ferramenta de autoria é facilitar a implementação de aplicações hipermídia, a disponibilização de componentes gráficos para entrada e saída de dados é feita através de um código procedural também identificado como mídia, o que é permitido pelo NCL.

Como na maioria das ferramentas de autoria para linguagem declarativa e hipermídia, o *Composer* também utiliza o modelo de visões para orientar a implementação de seus aplicativos. O *Composer* é composto por quatro tipos de visões: gráfica estrutural, gráfica temporal, gráfica de leiaute e textual. Estas quatro visões são integradas.

Aplicações NCL e de hipermídia têm como base para inserção das mídias a visão de leiaute. A visão de leiaute, conforme a Figura 3.14, possibilita a criação e configuração das regiões onde as mídias serão disponibilizadas. Esta visão possibilita ao autor criar, editar e apagar regiões. O programador pode criar sub-regiões ou regiões sobrepostas. As regiões podem ser definidas em tamanho por pixel ou por tamanhos relativos ou percentuais com relação à região-pai.

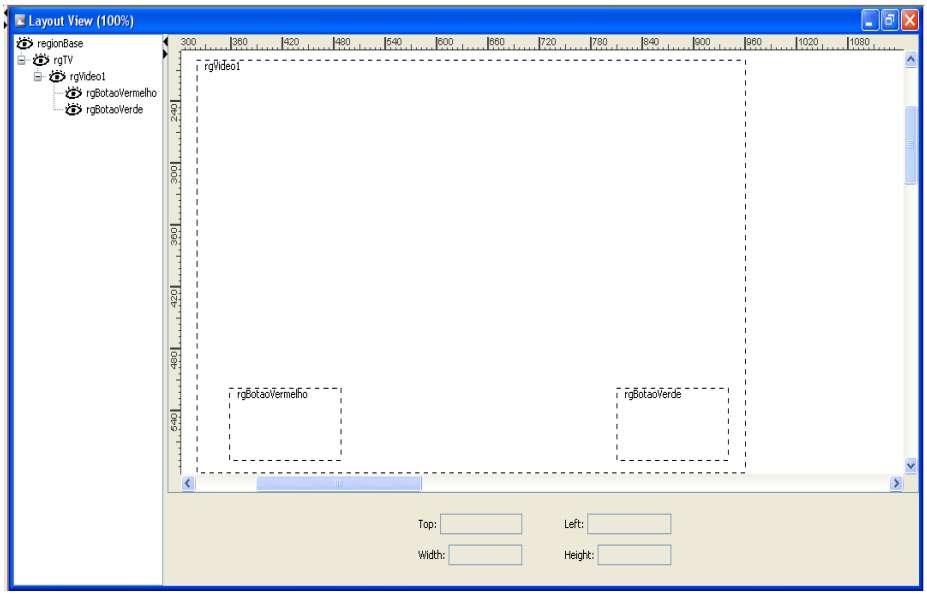


Figura 3.14 – Visão de leiaute
FONTE: Tela capturada do *software* Composer.

A visão estrutural possibilita ao usuário criar a estrutura lógica do aplicativo NCL que tem seus objetos de multimídia interligados por elos respondendo a eventos e sendo associados a estados. O programador pode criar, editar e apagar composições, objetos de mídia e elos. Na Figura 3.15, visão estrutural tem nós de contexto e os nós de mídia representados por vértices de um grafo. Os elos são representados por arestas entre vértices do grafo.

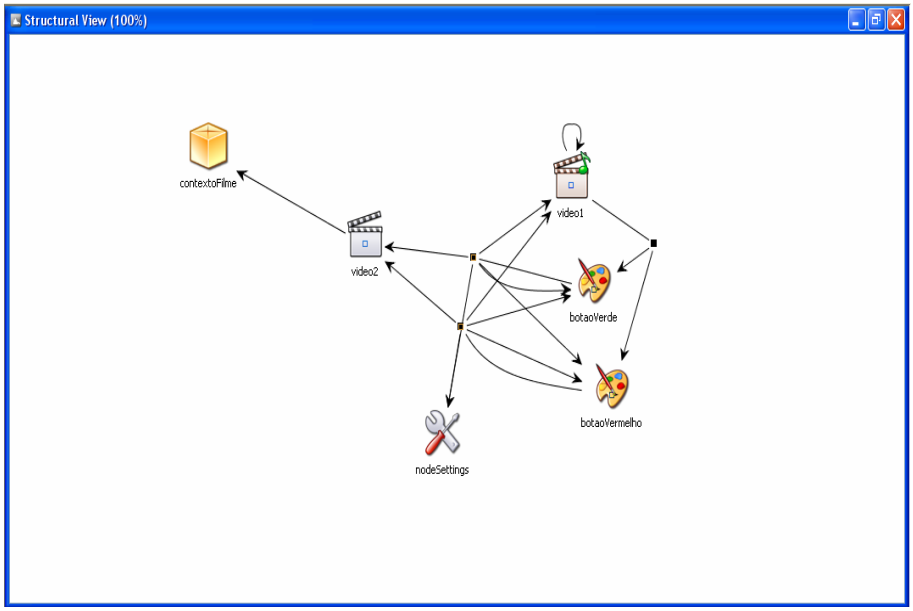


Figura 3.15 – Visão estrutural
FONTE: Tela capturada do *software* Composer.

Outra importante visão é a textual, que permite a edição do código NCL. Na Figura 3.16 a seguir é possível observar a estrutura do código NCL que é baseado no formato XML e também considerado como um documento XHTML. Um documento XHTML é baseado em linguagem declarativa baseada em mídias [19]. O NCL é considerado uma linguagem de cola, pois define a própria cola que prende as mídias em documentos multimídias.

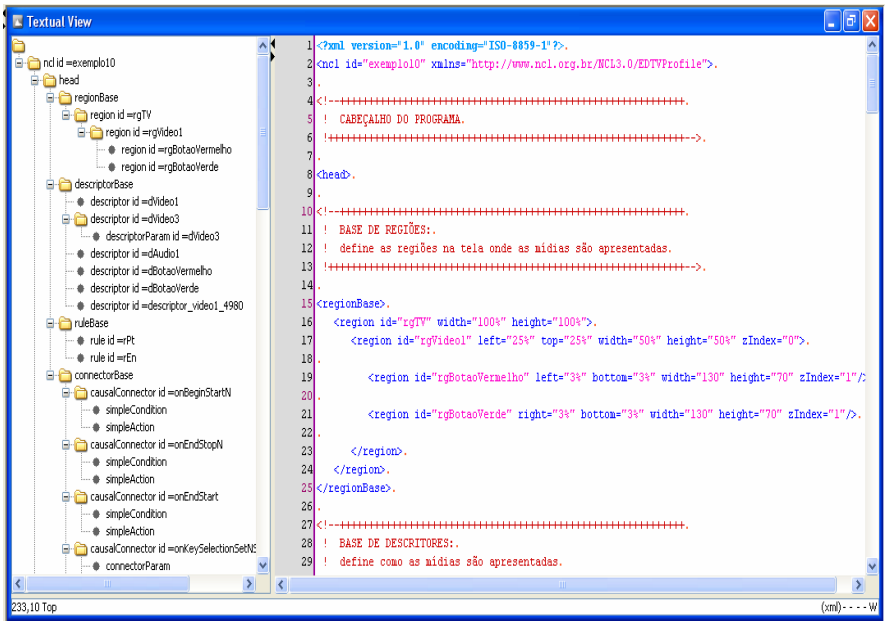


Figura 3.16 – Visão textual
FONTE: Tela capturada do *software* Composer.

Finalmente, a visão temporal permite também a especificação e edição de relacionamentos de sincronização temporal entre entidades de um documento NCL, definindo suas posições relativas no tempo. A visão temporal disponibiliza relacionamentos temporais com as âncoras temporais presentes nos nós de mídia (nó de áudio, vídeo e imagens), que são entidades chaves para a visualização do documento no eixo do tempo.

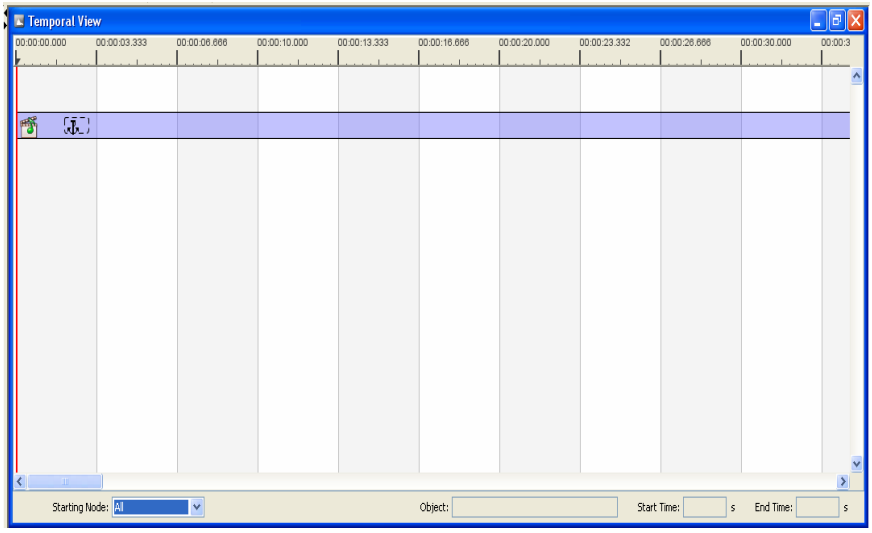


Figura 3.17 – Visão temporal
FONTE: Tela capturada do *software* Composer.

O *Composer* tem como foco principal a criação e manipulação de objetos de mídia dispostos na visão temporal. A criação de objetos de mídia no tempo e seus relacionamentos com os demais objetos são tratados de forma a tornar a especificação da apresentação mais intuitiva. Soluções para a representação e edição de objetos de mídia, relacionamentos entre objetos, relacionamentos interativos e edição ao vivo também são tratados. Finalmente, o *Composer* integra técnicas, a fim de facilitar o processo de edição de documentos NCL para TV Digital interativa. O *Composer* tem um *link* com o emulador GingaNCL Player [34], conforme a Figura 3.18 a seguir.

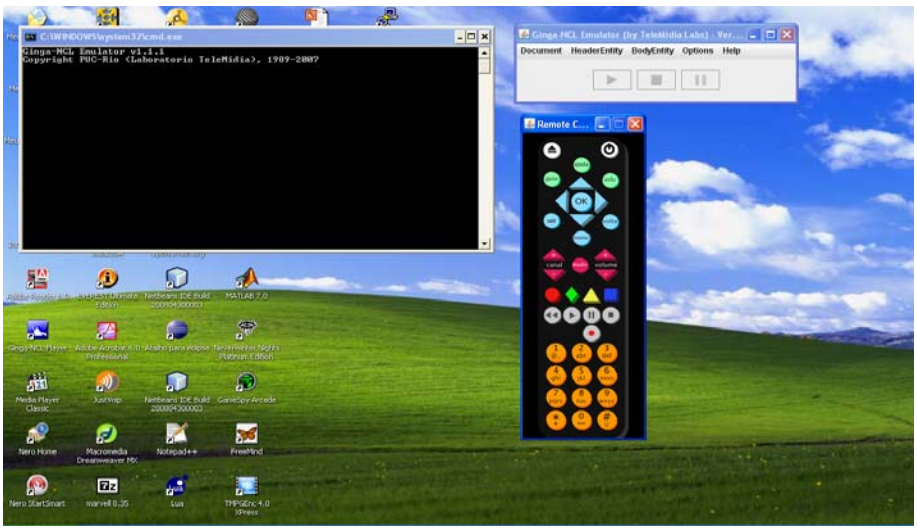


Figura 3.18 – Link do *Composer* com o emulador Ginga NCL *Player*
FONTE: Tela capturada do *software* Ginga Emulator.

O GingaNCL *Player* é outra ferramenta disponibilizada pelo laboratório Telemídia da PUC-RJ para testes de aplicações NCL.

3.2 CONCLUSÃO E ANÁLISE COMPARATIVA

Observados os inúmeros recursos oferecidos pelas ferramentas de autoria, pode-se chegar à conclusão que a maioria das ferramentas está convergindo para a disponibilização dos mesmos recursos e com apenas diferenciação quanto ao ambiente de aplicação. A seguir, a Tabela 3.1 apresenta as principais e originais características que contribuíram para o desenvolvimento das ferramentas de autoria descritas neste capítulo.

Ferramenta	Contexto Principal	Principais Características	Linguagem	Foco	Perfil
Delphi	- Desktop.	- Componentes gráficos; - Componentes não visíveis; - Drag and Drop; - Multijanelas; - Controladores de propriedades; - Formulários; - Gerenciador de projetos; - Programação orientada a eventos.	- Procedural; - Object Pascal; - Linkeditável;	- Interatividade e Banco de Dados.	- Software proprietário;
Visual Basic	- Desktop.	- Uso de DLL; - Uso de Templates;	- Procedural; - Basic; - Pré-Linkeditável.	- Interatividade e Banco de Dados.	- Software proprietário;
NetBeans	- Desktop.	- Programação Orientada a Objetos; - API Swing;	- Procedural; - Java; - Interpretável - JVM.	- Interatividade com entrada de saída de dados; - Programação Orientada a Objetos;	- Software livre;
Eclipse	- Desktop.	- Plug-in; - LuaEclipse; - NCLEclipse;	- Procedural; - Java; - Interpretável - JVM.	- Interatividade com entrada e saída de dados; - Programação Orientada a Objetos;	- Software livre;
DreamWeaver e FrontPage	- WEB.	- HTML;	- Declarativa; - HTML e ASP;	- Hipertexto e Hipermídia;	- Software proprietário;

Ferramenta	Contexto Principal	Principais Características	Linguagem	Foco	Perfil
			- Interpretável por Browser.		
Flash	- WEB; - Desktop.	- Animação e SctionScrpt	- Declarativa; - ECMAScript; - Interpretável; - AVM;	- Animação.	- Software proprietário;
JAME AUTHOR	- TV Digital;	- Estados dos componentes gráficos ou visíveis: Foco, Pressionado, Habilitado e Visível; - Emulador; - Editor do código.	- Procedural; - Java; - Middleware MHP/OCAP.	- Interatividade com entrada de saída de dados; - Visão Design; - Visão Navegação.	- Software proprietário;
Cardinal Studio	- TV Digital;	- Templates; - Emulador; - Container; - Tratamento de eventos; - Componentes invisíveis; - API para canal de retorno;	- Procedural; - Java; - Middleware MHP/DVB	- Interatividade com entrada de saída de dados; - Visão em camadas: Invisível, Gráfica e de Vídeo.	- Software proprietário;
AltiComposer	- TV Digital.	- Voltado para profissionais de Televisão;	- Procedural; - Java; - Middleware MHP/DVB	- Interatividade com entrada de saída de dados; - Baseado em Cenas, Planos, Shots e Atores;	- Software proprietário;
GriNS	- WEB.	- Hipermídia. - Sincronismo de mídias;	- SMIL e XML;	- Sincronismo temporal; - Visões: Temporal, Espacial e Textual.	- Software proprietário;
LimSee2	- WEB	- Hipermídia; - Sincronismo de mídias. - Visão de atributo dos componentes;	- SMIL e XML;	- Sincronismo temporal; - Visões: Temporal, Espacial , Textual. E de Atributos.	- Software proprietário;
Composer	- TV Digital;	- Hipermídia; - Sincronização das visões com base no modelo NCM; - NCM: Elos, Estados e Eventos; - Emulador GingaNCL.	- Declarativa; - NCL;	- Sincronismo temporal/espacial; - Visões: Temporal, Leiaute, Estrutural e Textual.	- software Livre.

Tabela 3.1 - Tabela de comparação entre ferramentas de autoria

Neste capitulo é importante destacar que a principais diferenças entre as ferramentas de autoria para computadores e para TV Digital são o conceito das visões para o autor e o sincronismo das mídias. Não se pode esquecer que o tempo de maturidade de uma ferramenta e sua demanda a torna madura e testada, exemplo: *Delphi* e *Visual Basic*. Com isso também torna-se possível entender que as ferramentas de autoria para TV Digital estão sendo conhecidas e depuradas pelos novos autores de aplicações. Finalmente, concluímos que as ferramentas de autoria sempre trarão novos recursos, mas a necessidade de um autor não leigo é fundamental para o seu bom uso.

4 AUTORIA EM NCLUA PARA TV DIGITAL INTERATIVA

4.1 INTRODUÇÃO

A demora na disponibilização de receptores digitais com o *middleware* brasileiro, a dificuldade de disponibilização do Ginga-J para desenvolvimento de aplicações com interatividade plena, e a confirmação da linguagem Lua como alternativa ao Ginga-J, vem despertando um crescente interesse nos desenvolvedores em utilizar o Ginga-NCLua como solução, pois já vem disponibilizada no emulador desde fevereiro de 2008 [45].

O LuaComp é uma ferramenta de autoria de aplicações para TV Digital interativa projetado para criação de aplicações híbridas (procedurais e declarativas) com códigos em linguagem de *script (Lua)* e NCL. O LuaComp disponibiliza componentes gráficos baseado no *framework* LuaOnTV [44].

O LuaOnTV foi desenvolvido no laboratório de TV digital do ENE/UnB sob a gerencia desse autor e orientação do professor Paulo Gondim. Para a construção do LuaOnTV foram definidos requisitos funcionais utilizando componentes gráficos para facilitar a criação de aplicativos e consequentemente suprir toda a necessidade de implementar códigos procedurais, deixando somente a parte de sincronismo de mídia com o NCL. Este *framework* possibilita também uma diminuição substancial do código e consequentemente do tamanho da aplicação que será enviada via *carrousel* ou pré-armazenada em *settop boxes*.

Este capítulo apresenta a linguagem Lua, o LuaOnTv e a arquitetura do LuaComp.

4.2 LINGUAGEM LUA

Lua é uma linguagem de *script* não muito diferente da Tcl, *Perl*, ou *Python*. Como a Tcl, Lua é embarcada e interpretada, pois depende de máquina interpretadora. É muito fácil conectar Lua com outras línguas como C, C++, ou mesmo Fortran. Lua é uma linguagem com uma sintaxe bastante intuitiva e parecida com o *Python*. Lua é interpretada com possibilidade de tipagem dinâmica, e com diversas facilidades [47].

Lua foi criada em 1993 no Tecgraf, laboratório de pesquisa e desenvolvimento da PUC-Rio e que tem ganho e relevância global [54]. Atualmente a linguagem é utilizada principalmente em jogos, como World of Warcraft, The Sims e Sim City. Lua não é uma

linguagem especifica para jogos, sendo também utilizada no software Adobe Photoshop Lightroom e o middleware Ginga do sistema brasileiro.

Outra importante característica a ressaltar é que Lua tem um núcleo com bibliotecas padrão que ocupam menos que 200k, podendo ser utilizada tanto para escrever *script*, como sistemas complexos.

Luiz Fernando Gomes Soares, considerado pai do Ginga-NCL, diz que

> A autoria de aplicações utilizando linguagens declarativas como NCL é vantajosa quando o seu desenvolvimento depende apenas de recursos previstos no projeto da linguagem. No entanto, quando uma aplicação necessita de funcionalidades não previstas pela linguagem declarativa, a solução pode se tornar complicada ou até mesmo impossível [46].

O LuaComp é uma ferramenta de autoria que supre as dificuldades de disponibilizar interatividade com componentes básicos para entrada e saída de dados. Linguagens declarativas como NCL são acessíveis a autores de conteúdo audiovisual que não possuem base de programação.

Soares também afirma que:

> Em NCL, a realização de muitas tarefas é complicada sem o auxílio imperativo, tal como processamento matemático, manipulação sobre textos, uso do canal de interatividade, controle fino do teclado, animações e colisões para objetos gráficos e, de maneira geral, tarefas que necessitem da especificação de algoritmos e estruturas de dados [46].

O pai do Ginga-NCL considera o NCLua, nova classe de objetos de mídia Lua, a principal via de integração de NCL a um ambiente imperativo (procedural).

O Sistema Brasileiro de Televisão Digital, por ainda estar sendo implantado progressivamente, e pela atual indisponibilidade do *middleware* Ginga-J, justifica a importância de desenvolvimento de soluções alternativas em termos de paradigma de programação procedural.

A linguagem Lua e a linguagem Java, apesar de serem procedurais se diferenciam pela forma de gerar suas aplicações. A linguagem Lua é programada através de *scripts* sem a necessidade de se gerar outro código intermediário e a linguagem Java, através de seu código fonte “.java”, necessita da criação de um outro código intermediário “*.class*” para ser executado pela máquina virtual [48]. A linguagem Lua pode ser considerada mais eficiente que Java tanto quanto ao tempo de uso de CPU como utilização de memória [48] e o

observado na Figura 4.1. A escolha da linguagem Lua é motivada pelo seu pouco consumo de memória, e também devido a atual disponibilidade e facilidade de acesso às plataformas de desenvolvimento e testes para aplicações NCL e Lua.

Uma também importante declaração de Luiz Fernando Gomes Soares foi a maior motivação para o desenvolvimento do LuaComp.

> Tem duas coisas no movimento pelo Java que eu, particularmente, não gosto. Nada contra o Java. O Java é bom. Do Java eu gosto. O que está por trás desse movimento em torno do Java é que preocupa, porque pode ser ruim. Podemos, sim fazer muita coisa só com NCL/Lua. Todo mundo sabe disso. mas parece que não sabe [46].

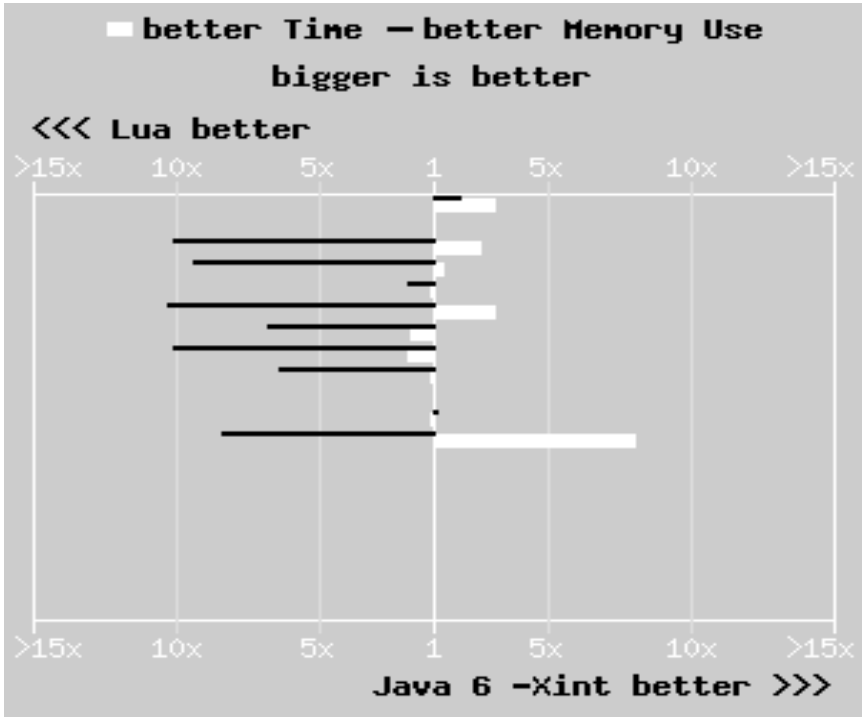


Figura 4.1 - Lua x Java
FONTE: THE COMPUTER LANGUAGE BENCHMARKS GAME, 2008. [49]

4.3 REQUISITOS PARA O LUACOMP

O LuaComp é uma ferramenta de autoria que facilita a criação de um NCLua. "*Um NCLua deve ser escrito em um arquivo separado do documento NCL*" [46] e isto é a principal característica do LuaComp. Um arquivo escrito apenas na linguagem NCL apenas referencia um NCLua ou arquivo Lua. Nas aplicações multimídia, o documento NCL deve ser o componente mestre onde todos os relacionamentos entre objetos de mídia devem ser definidos explicitamente.

Uma aplicação para TV Digital deve ser implementada com um mínimo de intrusão e o LuaComp segue os seguintes requisitos:

a) Os códigos devem ser alterados o mínimo possível;

b) do ponto de vista do desenvolvedor, deve ser mantida uma fronteira bem delineada entre os dois modos de programação, o declarativo, NCL, e o procedural ou imperativa, Lua;

c) em termos de *design*, a relação entre os dois ambientes deve ser ortogonal, em outras palavras, operações em um ambiente não devem produzir efeitos imprevistos no outro, ou seja, harmonia entre o código NCL e o NCLua;

d) manipular arquivos XML como repositório do projeto e das páginas;

e) disponibilizar ambiente de edição WYSIWYG;

f) possibilitar criar e importar *templates* de projetos e páginas;

g) facilitar a manipulação de imagens e seu leiaute na tela [46];

O que mais justifica o uso apenas básico do *Composer* para *link* de mídias audiovisuais e a não exploração de desenvolvimento de aplicações complexas é a dificuldade de editar o código XHTML e dar manutenção a uma aplicação NCL sem ser intrusivo. O padrão *ECMAScript* utilizado na integração *XHTMLECMAScript*, encontrado nos sistemas de TV Digital e na *Web*, é bastante intrusivo e, portanto, vai contra os requisitos apontados anteriormente. O LuaComp segue rigorosamente estes requisitos e tem como objetivo gerar códigos NCL com um mínimo de complexidade. O LuaComp gera códigos NCL oferecendo padrões de usabilidade, interatividade e também disponibilizando um mínimo de sincronismo entre os códigos declarativos e imperativos.

4.4 LUAONTV

O LuaOnTV é um *framework* baseado na linguagem Lua utilizando os recursos do NCLua e baseado principalmente na API Lua Canvas. O LuaOnTv foi desenvolvido no laboratório de TV Digital e disponibilizado de forma gratuita no LuaForge [74]. O LuaOnTv foi uma idéia e projeto deste autor, orientado pelo professor Paulo Gondim. Este framework é uma contribuição deste autor através de sua gerência e dos alunos de graduação [44]. O que

mais motivou o desenvolvimento do LuaOnTV foi a possibilidade de disponibilizar para a comunidade de *software* livre uma ferramenta poderosa de interatividade, ou seja, uma alternativa para o Ginga-J.

O LuaOnTV também teve como desafio a implementação de códigos baseados no paradigma da programação orientada a objetos, pois a linguagem Lua não é uma linguagem para POO, segundo Becker et al. [51]. A idéia é utilizar uma meta-tabela como uma classe onde o campo _index é a própria tabela. Os métodos são implementados internamente na sua meta-tabela, de forma análoga a uma classe. Uma das grandes vantagens da linguagem Lua é que pode receber atributos e funções e guardar em uma única meta-tabela. Assim, é possível definir uma estrutura básica para a classe e um construtor. O *framework* LuaOnTv mostra esta prototipagem. Observe na meta-tabela ou "classe" a seguir que os atributos são declarados sem definição de tipos e com inicialização.

```
DefaultComponent={
 bgImage=nil,
 left=0,
 top=0,
 width=0,
 height=0,
 name='',
 focus=false,
 focusable=true,
 focusColor="red",
 focusVisible=true,
 visible=true,
 group=false,
 focusIndex=0,
 moveUp=0,
 moveLeft=0,
 moveRight=0,
 moveDown=0,
 action=function()end,
 borderColor="black"
}
```

O LuaOnTV também tem uma função com o mesmo nome da meta-tabela ou classe representando o construtor:

```
function DefaultComponent:new(o)
        o = o or {}
        setmetatable (o, self)
        self.__index = self
        if(o.parent~=nil) then
                o.parent:add(o)
        end
        return o
end
```

Da mesma forma que o Java o LuaOnTV tem suas meta-tabelas com os métodos *sets* e *gets* também representados a seguir:

```
function DefaultComponent:getName()
        return self.name
end
function DefaultComponent:setPosition(left,top)
 self.left=left
 self.top=top
end
function DefaultComponent:getPosition()
 return self.left, self.top
end
function DefaultComponent:setbgImage(image)
 self.bgImage=image
end
function DefaultComponent:getbgImage()
 return self.bgImage
end
function DefaultComponent:isFocus()
 return self.focus
end
function DefaultComponent:setFocus(trueOrFalse)
 self.focus = trueOrFalse
end
function DefaultComponent:setFocusable(trueOrFalse)
 self.focus = trueOrFalse
end
function DefaultComponent:getFocusable()
 return self.focus
end
function DefaultComponent:setFocusColor(focusColor)
 self.focusColor = focusColor
end
function DefaultComponent:getFocusable()
 return self.focusColor
end
function DefaultComponent:setWidth(width)
 self.width = width
end
function DefaultComponent:getWidth()
 return self.width
end
function DefaultComponent:setHeight(height)
 self.height = height
end
function DefaultComponent:getHeight()
 return self.height
end
function DefaultComponent:setHeight(height)
 self.height = height
end
function DefaultComponent:getHeight()
 return self.height
end
function DefaultComponent:setFocusVisible(focusVisible)
 self.focusVisible = focusVisible
end
```

```
function DefaultComponent:isFocusVisible()
 return self.focusVisible
end
function DefaultComponent:setVisible(visible)
 self.visible = visible
end
function DefaultComponent:isVisible()
 return self.visible
end
function DefaultComponent:setGroup(group)
 self.group = group
end
function DefaultComponent:isGroup()
 return self.group
end
function DefaultComponent:setFocusParam(focusIndex,moveUp,moveRight,moveDown,moveLeft)
 self.focusIndex = focusIndex
 self.moveUp = moveUp
 self.moveRight = moveRight
 self.moveDown = moveDown
 self.moveLeft = moveLeft
end
function DefaultComponent:addKeyListenner(method)
 self.listener = method
end
function DefaultComponent:setAction(method)
 self.action = method
end
function DefaultComponent:setBgColor(bgColor)
 self.bgColor = bgColor
end
function DefaultComponent:getBgColor()
 return self.bgColor
end
function DefaultComponent:setBorderColor(borderColor)
 self.borderColor = borderColor
end
function DefaultComponent:getBorderColor()
 return self.borderColor
end
```

A implementação de códigos orientados a objetos propicia a reusabilidade de código e consequentemente uma melhor manutenção dos próprios. Ricardo Silva [53] explica que um *Framework* orientado a objetos pode ter seu desenvolvimento baseado em componentes, e sua abordagem promove reuso de projeto e código em um nível de granularidade elevado. Para Ricardo Silva e Tom Price [52], um *framework* orientado a objetos pode ser definido como um arcabouço que tem como principal finalidade aplicar os conceitos de reusabilidade de código e diminuir o tempo e esforço na produção de *software*, através da implementação de estruturas de classes interrrelacionadas.

Segundo Silva [53], um *framework* pode se classificar como dirigido à arquitetura ou dirigido a dados. No primeiro caso a aplicação deve ser gerada a partir da criação de

subclasses das classes do *framework*. No segundo caso, diferentes aplicações são produzidas a partir de diferentes combinações de objetos, instâncias das classes presentes no *framework*. O LuaOnTV é baseado no primeiro caso, dirigido a arquitetura. O LuaOnTV é um *framework* especializado em componentes gráficos e seus atributos e métodos são bastante conhecidos. Apesar de também existir uma combinação entre estes componentes gráficos, por exemplo, um *RadioButtonGroup*, a estrutura das classes e suas heranças resolvem perfeitamente o conceito de combinação de objetos, pois a implementação de suas classes e subclasses é facilitada. O Diagrama de Classe da Figura 4.2 mostra como o *framework* LuaOnTV foi projetado.

O LuaOnTV teve suas funcionalidades baseadas nos componentes gráficos básicos e fundamentais para implementação de aplicações para entrada e saída de dados. A principal característica do *framework* LuaOnTV é disponibilizar classes de componentes gráficos baseados no módulo Canvas com o foco na disponibilização de componentes gráficos que facilitem a interatividade para entrada e saída de dados. O Ginga-NCL também disponibiliza um módulo chamado Canvas através do NCLua. O NCLua possibilita a utilização de componentes gráficos e o desenvolvimento de outros sem a necessidade de instalação de bibliotecas no receptor digital. O código NCL associa uma região e inicia o NCLua instanciando um objeto gráfico como uma variável global canvas. O objeto gráfico canvas passa a ser um nó de mídia onde todas as operações gráficas são feitas. O Lua Canvas é também uma colaboração do laboratório Telemidia [54].

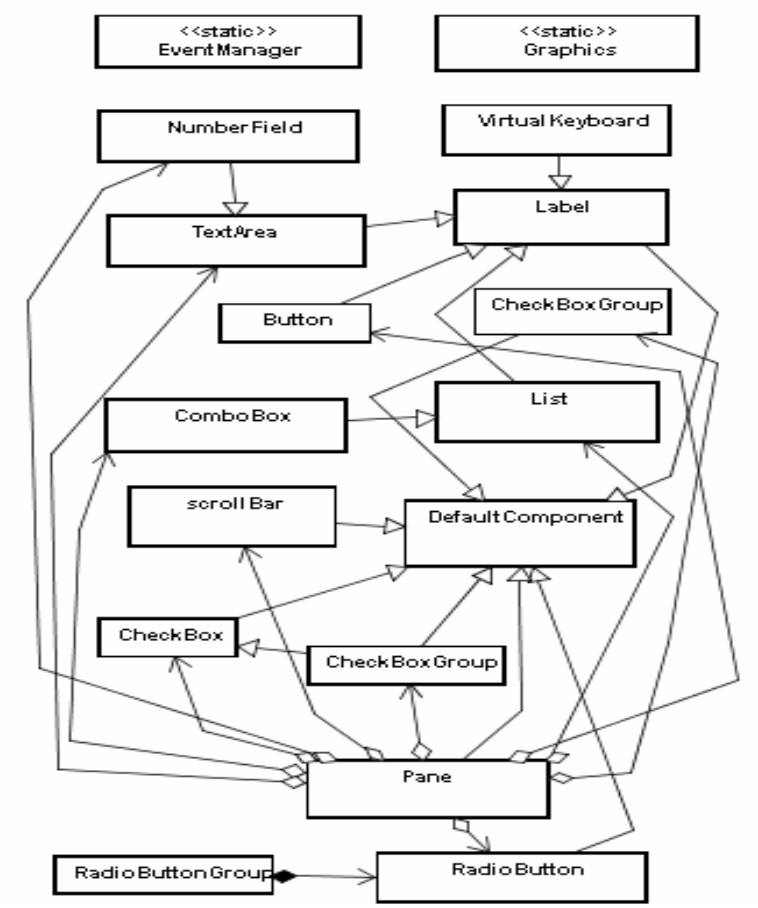


Figura 4.2 – Diagrama de classes – LuaOnTV [44]

Baseando em algumas limitações provenientes da ainda não disponibilização completa do Ginga, o LuaOnTV procurou definir como principais requisitos funcionais: disponibilizar componentes gráficos para facilitar a criação de aplicativos e consequentemente suprir toda a necessidade de implementar códigos procedurais, deixando somente a parte de sincronismo de mídia com o NCL. Este *framework* também teve como principal objetivo criar códigos que possibilitem uma diminuição substancial do código e consequentemente do tamanho da aplicação que será enviada via *carrousel*. O laboratório de TV Digital da UnB comparou a implementação de uma aplicação onde os códigos Lua representaram uma diminuição substancial de tamanho em comparação com o NCL. Uma aplicação com códigos NCL e imagens que simulam estes componentes gráficos chagavam a quase 1,5 Mbytes. A mesma aplicação com estes componentes gráficos chegou a pouco mais de 60 Kbytes.

O LuaOnTV foi planejado em três etapas. A primeira etapa foi para a definição dos componentes gráficos utilizados na maioria dos ambientes de desenvolvimento de interface, como o *NetBeans*, Eclipse e o *Delphi*. A segunda etapa ou a versão 2.0 foi também a definição dos componentes não visuais ou não gráficos, que são utilizados para responder a

eventos e particularmente na TV Digital a navegação de foco. A terceira etapa se resumiu à especificação destes componentes utilizando UML e particularmente o diagrama de classes conforme a Figura 4.2. Importante ressaltar que a não implementação de componentes não gráficos para gerenciamento de canal de retorno foi decidida devido a também não disponibilização de norma para estas funcionalidades. Esta limitação também foi fundamental para a não disponibilização dos componentes não visuais no LuaComp.

Güell et al. [55] afirmam que os componentes gráficos foram desenvolvidos visualmente com base nos padrões de usabilidade utilizados pela BBC de Londres e também através da observação da maioria das aplicações já disponibilizadas na Europa. A norma ISO 9241 define usabilidade como a eficácia, a eficiência e a satisfação que um usuário tem ao interagir com um sistema. A adaptação dos componentes gráficos utilizados em computadores pessoais para o ambiente de TV Digital sofre uma série de diferenças importantes. Uma TV Digital interativa apresenta uma série de características relevantes como: menor resolução com áreas periféricas sujeita a distorções, não rolagem horizontal, dificuldade de leitura de textos pequenos devido à distância do telespectador, além de alguns padrões mundiais de não uso de componentes com relevo.

Alguns ambientes de desenvolvimento como o *Netbeans*, Eclipse e *Delphi* serviram como modelo visual para a implementação dos componentes gráficos. Os componentes visuais escolhidos foram: *Button*, *CheckBox*, *CheckGroup*, *ComboBox*, *Label*, *List*, *NumberField*, *PasswordField*, *RadioButton*, *RadioButtonGroup*, *ScrollBar*, *TextArea*, *TextField*, *KeyBoard* e o *Pane*. Estes componentes visuais foram escolhidos conforme restrição de não uso do *mouse* nas aplicações em TV Digital. Os componentes não visuais ou não gráficos foram definidos para tratar principalmente os eventos de navegação. O *EventManager*, um componente não gráfico e de controle de foco nos componentes, é o principal componente do *framework*.

O projeto do *framework*, conforme a técnica de *framework* baseada em componentes, foi implementado com o forte paradigma da herança. O LuaOnTV pode ser subdividido e dois grupos de componentes, os não gráficos que controlam o foco, o *EventManager*, o repositório base, *Panel*, e uma superclasse dos componentes gráficos, o *DefaultComponent*.

O *EventManager* é uma classe estática que possibilita o controle de eventos, tais como mudança de foco. O controle de foco talvez seja a funcionalidade mais importante de todo o *framework*. O *EventManager* interage com o componente *Panel* que possui os

componentes gráficos, pois o controle de foco incide como o componente atual onde se navega com o foco da aplicação. Como base para todas as outras classes e componentes, o componente *DefaultComponent* foi desenvolvido como a classe principal da qual todos os outros componentes herdam, e possui métodos e atributos comuns a todos os componentes. Como toda aplicação deve ter um painel principal, onde serão adicionados os componentes que compõem a tela, foi desenvolvido um componente principal que atende a essa necessidade, o *Panel*.

O grupo dos componentes gráficos e que herdam do *DefaulComponent* é formado pelo *Button*, *Label*, *List*, *ComboBox*, *CheckBox*, *RadioButton*, *TextArea* e *NumberField*.

Pane	**add(*component*)** – adiciona o componente passado como parâmetro ao Pane; **paint( )** – pinta na tela o Pane e todos os componentes que ele contém; **clear( )** – remove todos os componentes do Pane; **remove(*component*)** – remove o componente passado como parâmetro do Pane. Caso não seja fornecido um parâmetro, o último adicionado é removido.
DefaultComponent	**getName()** – retorna o atributo de identificação do componente; **setPosition(*left,top*)** – muda a posição do componente para os valores dos parâmetros *left* e *top*; **getPosition()** – retorna os valores left e top do componente; **setbgImage(*image*)** – passa o endereço de uma imagem a ser utilizada como background do componente; **getbgImage()** – retorna a imagem de background atual; **isFocus()** – retorna true se o componente estiver em foco e false caso contrário; **setFocus(*trueOrFalse*)** – muda o atributo focus do componente para o valor do parâmetro *trueOrFalse*; **setFocusable(*trueOrFalse*)** – muda o atributo focusable para o valor do parâmetro *trueOrFalse*; **getFocusable()** – retorna o valor do atributo focusable; **setFocusColor(*focusColor*)** – muda a cor do componente quando o mesmo estiver em foco para *focusColor*; **getFocusColor(*focusColor*)** – retorna o valor do atributo focusColor; **setWidth(*width*)** – muda o valor do atributo width para o parâmetro *width*. **getWidth()** - retorna o valor do atributo width; **setHeight(*height*)** - muda o valor do atributo height para o parâmetro *height*. **getHeight()** - retorna o valor do atributo height; **setFocusVisible(*focusVisible*)** – caso seja passado true como parâmetro, o componente será desenhado de forma diferente quando estiver em foco. Caso o parâmetro seja false, não haverá diferença no desenho do componente quando o mesmo estiver em foco. **isFocusVisible()** - retorna o valor do atributo focusVisible; **setVisible(*visible*)** – caso o parâmetro seja false, o componente não será pintado na tela, ficará invisível. Caso seja true, ele será pintado. **isVisible()** - retorna o valor do atributo visible; **setGroup(*group*)** – Define o componente como tipo group caso o parâmetro seja true. Um componente tipo group pode adicionar outros componentes dentro de si. Caso o parâmetro seja false, o componente não será tipo group. **isGroup()** - retorna o valor do atributo group; **setFocusParam(*focusIndex,moveUp,moveRight,moveDown,moveLeft*)** – Define os parâmetros de foco do component. focusIndex é o índice único para cada componente que o identifica na mudança de foco. Os parâmetros moveUp, moveRight, moveDown e moveLeft recebem o índice dos componentes acima, à direita, abaixo e à esquerda do componente, respectivamente;

	addKeyListenner(*method*) – define um método que será chamado sempre que uma tecla for pressionada. Esse método receberá como parâmetro a tecla sendo pressionada; **setAction(*method*)** – define um método que será chamado quando o componente estiver em foco e a tecla "ENTER" for pressionada; **setBgColor(*bgColor*)** – muda a cor de background do componente para *bgColor*; **getBgColor()** - retorna o valor do atributo bgColor; **setBorderColor(*borderColor*)** – muda a cor de borda do componente para *borderColor*; **getBorderColor()** - retorna o valor do atributo borderColor.
EventManager	**changeFocus(key)** – muda o componente em foco atualmente, de acordo com a tecla pressionada, definida pelo parâmetro key. Este método geralmente é chamado automaticamente. **getFocus(context)** – procura por um componente em foco dentro do parâmetro context. Caso seja encontrado um componente em foco, ele é retornado. O parâmetro context não é obrigatório,e caso seja omitido o método procurará no contexto atual. **requestFocus(newFocus)** – retira o foco do componente atualmente em foco e coloca o componente passado como parâmetro em foco. O parâmetro não é obrigatório, e caso seja omitido, o componente focável mais perto do canto superior direito será colocado em foco. **getContext()** – retorna o contexto atual da aplicação. **eraseFocus(component)** – apaga o foco do componente passado como parâmetro e de qualquer componente que eventualmente esteja adicionado dentro de component. **setMainPane(pane)** – define como painel principal o painel passado como parâmetro. A chamada a este método é obrigatória em qualquer aplicação que deva ter mudança de foco. **getMainPane()** – retorna o painel principal atual.
Button	**setBevel(bevel)** – define que o Button deve ter borda, caso o parâmetro seja true e que não deve ter borda, caso contrário; **getBevel()** – retorna o valor do atributo bevel; **setRoundness(roundness)** – define se a borda do Button deve ser arredondada. Deve ser passado como parâmetro um número, usualmente de 1 a 5, sendo que quanto maior o valor, mais arredondada será a borda; **getRoundness()** – retorna o valor do atributo roundness; **getLabel()** – retorna a label do Button.
Label	**setText(text)** – muda o texto atual da Label para text; **getText()** – retorna uma String com o texto atual da Label; **setfontType(fontType)** – define a fonte do texto. O parâmetro deve ser uma String; **getfontType()** – retorna o valor do atributo fontType; **setfontStyle(fontStyle)** – defina o fontStyle do texto. Os valores possíveis são 'bold', 'italic' ou 'bold-italic'; **getfontStyle()** – retorna o valor do atributo fontStyle; **setfontSize(fontSize)** – define o tamanho da fonte do texto. O parâmetro deve ser um inteiro; **getfontSize()** – retorna o valor do atributo fontSize; **setfontColor(fontColor)** – define a cor da fonte do texto; **getfontColor()** – retorna o valor do atributo fontColor;
List	**getIntemsSelected()** – retorna uma tabela com os itens da List atualmente selecionados; **addItems(...)** – adiciona itens à List. Os itens devem estar no formato String, e pode-se passar quantos itens forem precisos de uma vez como parâmetro, separando cada um por vírgula; **removeItems(item)** – remove da List o item da posição passada como parâmetro. O parâmetro deve ser um inteiro; **setWindowSize(windowSize)** – define a quantidade máxima de itens que podem ser mostrados simultaneamente na tela. Caso o número de itens na List seja maior que o tamanha do janela, será adicionado automaticamente um ScrollBar. O parâmetro do método deve ser um inteiro; **getWindowSize()** – retorna o valor do atributo windowSize; **setScrollbarVisibility(scrollbarVisible)** – caso o parâmetro seja true, o ScrollBar da List será visível. O ScrollBar não será visível caso o parâmetro seja false;

	getScrollBarVisibility() – retorna o valor do atributo scrollbarVisible.
ComboBox	**getSelected()** – retorna o item atualmente selecionado. Sempre haverá um item selecionado no ComboBox. Este método retorna usualmente uma Label correspondendo ao item selecionado.
CheckBox	**setSelected(trueOrFalse)** – edita o atributo que diz se o CheckBox está selecionado ou não; **isSelected()** – retorna true caso o CheckBox esteja selecionado e false caso contrário; **setInnerColor(color)** – edita a cor da parte interior do CheckBox; **getInnerColor()** – retorna a cor da parte interior do CheckBox; **setOutterColor(color)** – edita a cor da parte exterior do CheckBox; **getOutterColor()** – retorna a cor da parte exterior do CheckBox; **setSelectionColor(color)** – edita a cor da marca de seleção do CheckBox; **getSelectionColor()** – retorna a cor da marca de seleção do CheckBox; **setLabel(label)** – edita o rótulo do CheckBox. O parâmetro label deve ser um componente da classe Label; **getLabel()** – retorna o componente Label que é o rótulo do CheckBox; **setSelectedImage(imagePath)** – pode-se configurar uma imagem para representar um CheckBox selecionado usando este método e passando o caminho da imagem como parâmetro. Obs.: para atribuir uma imagem para representar um CheckBox em estado unselected usa-se o método setbgImage(imagePath) da classe pai DefaultComponent; **getSelectedImage()** – retorna a imagem que representa um CheckBox selecionado.
RadioButton	**setSelected(trueOrFalse)** – edita o atributo que diz se o RadioButton está selecionado ou não; **isSelected()** – retorna true caso o RadioButton esteja selecionado e false caso contrário; **setInnerColor(color)** – edita a cor da parte interior do RadioButton; **getInnerColor()** – retorna a cor da parte interior do RadioButton; **setOutterColor(color)** – edita a cor da parte exterior do RadioButton; **getOutterColor()** – retorna a cor da parte exterior do RadioButton; **setSelectionColor(color)** – edita a cor da marca de seleção do RadioButton; **getSelectionColor()** – retorna a cor da marca de seleção do RadioButton; **setLabel(label)** – edita o rótulo do RadioButton. O parâmetro label deve ser um componente da classe Label; **getLabel()** – retorna o componente Label que é o rótulo do RadioButton; **setSelectedImage(imagePath)** – pode-se configurar uma imagem para representar um RadioButton selecionado usando este método e passando o caminho da imagem como parâmetro. Obs.: para atribuir uma imagem para representar um RadioButton em estado unselected usa-se o método setbgImage(imagePath) da classe pai DefaultComponent; **getSelectedImage()** – retorna a imagem que representa um RadioButton selecionado.
TextArea	**setKeyOut(key)** – Método responsável por definir o evento de saída do componente, ou seja, o botão que retirará o foco da barra de rolagem ou do teclado virtual. **getKeyOut()** – Retorna o botão responsável pelo evento de saída do foco. **setEraseKey(key)** – Para a situação de caixa de texto de entrada, tem-se a possibilidade de escolher o botão que irá apagar os caracteres da caixa de texto. **getEraseKey()** – Retorna o botão responsável por apagar os caracteres da caixa de texto. **setEditable(trueOrFalse)** – Define se o TextArea é editável ou não, recebendo “true” ou “false”. Lembrando das condições que foram colocadas para o TextArea. **isEditable()** – Retorna se o TextArea é editável ou não. **setCols(cols)** – Define a quantidade de caracteres ou colunas que uma linha deve suportar do TextArea. **getCols()** – Retorna a quantidade de caracteres que cada linha suporta. **setRows(rows)** – Define a quantidade de linhas que o TextArea terá, lembrando novamente das condições colocadas no início. **getRows()** – Retorna a quantidade linhas do TextArea.
NumberField	**setPassword(password)** – Define através de “true” ou “false”, se será do tipo password. **isPassword()** – Retorna se é do tipo password ou não. **setValue(newValue)** – Define o valor da caixa de texto. **getValue()** – Retorna o valor atual da caixa de texto. **setEraseKey(key)** - Para a situação de caixa de texto de entrada, tem-se a possibilidade

	de escolher o botão que irá apagar os caracteres da caixa de texto. **getEraseKey()** – Retorna o botão responsável por apagar os caracteres da caixa de texto. **eraseText()** – Limpa todo o conteúdo do NumberField.

Tabela 4.1 – Tabela de métodos dos componentes LuaOnTV

Para melhor explicar a simplicidade de se implementar um código utilizando o LuaOnTV:

```
dofile('Pane.lua') – semelhante ao import do java, adicionando todos os elementos da classe Pane
painel = Pane:new{width=640, height=480,bgColor="black"} – criasse o objeto painel da classe Pane
-- serão criados objetos para cada componente
objeto0=Image:new{left=43,top=161,width=251,height=214,image="./images/basefundo001.JPG"}
objeto1=Label:new{left=123,top=169,width=95,height=23,text="Banco XYZ",fontColor="blue"}
objeto2=Label:new{left=63,top=206,width=78,height=23,text="Agencia:"}
objeto3=NumberField:new{password=false,left=147,top=204,width=90,height=30,cols=8}
objeto4=Label:new{left=66,top=246,width=75,height=23,text="C.Corr.:"}
objeto5=NumberField:new{password=false,left=147,top=244,width=90,height=30,cols=8}
objeto6=Label:new{left=76,top=289,width=64,height=23,text="Senha:"}
objeto7=NumberField:new{password=true,left=146,top=285,width=90,height=23}
objeto8=Button:new{left=115,top=328,width=100,height=30,text="Entrar"}
-- o objeto painel recebe os componentes gráficos também criados
painel:add(objeto0)
painel:add(objeto1)
painel:add(objeto2)
painel:add(objeto3)
painel:add(objeto4)
painel:add(objeto5)
painel:add(objeto6)
painel:add(objeto7)
painel:add(objeto8)
-- é adicionado o método que controla o foco dos componentes
objeto3:setFocusParam(1,4,2,2,4)
objeto5:setFocusParam(2,1,3,3,1)
objeto7:setFocusParam(3,2,4,4,2)
objeto8:setFocusParam(4,3,1,1,3)
objeto8:setAction(function() dofile('lua002.lua')end)
-- a classe de controle de focu recebe o componente painel
EventManager:setMainPane(painel)
-- o componente que desenha a tela é pintado e consequentemente a aplicação é disponibilizada
painel:paint()
canvas:flush()
```

4.5 LUACOMP

O LuaComp surgiu da necessidade de se ter uma solução para utilizar componentes gráficos na entrada e saída de dados, bem como o uso de imagens para o *design* das telas, e impulsionado principalmente pela demora na disponibilização do Ginga-J. O LuaComp é uma ferramenta de autoria para construção de aplicações híbridas para TV Digital. Esta ferramenta não objetiva reproduzir as mesmas funcionalidades do *Composer*, e sim complementar as

necessidades do mercado oferecendo ao autor ferramentas gráficas de interatividade para entrada e saída de dados, e programação visual com imagens.

Como as linguagens imperativas introduzem uma maior complexidade de programação, a simplicidade e facilidade de uso e entendimento do LuaOnTV aliado a uma ferramenta de autoria com a funcionalidade de uso WYSIWYG transformam o LuaComp numa ferramenta bastante produtiva. Soares [46] destaca o NCLua como solução para suprir as limitações das aplicações declarativas que usam somente o NCL como base. O LuaComp é uma ferramenta de autoria que possibilita o uso do NCLua e consequentemente possibilita a criação de aplicação com quase todos os recursos de interatividade.

Com o intuito de ser uma ferramenta de criação de aplicações interativas para TV Digital que minimize os conceitos de intrusão, o LuaComp tem como principais requisitos:

a) Elaborar uma interface gráfica para agilizar na produção de aplicativos com componentes gráficos para entrada e saída de dados;

b) possibilitar que o autor trabalhe em ambiente WYSIWYG;

c) suporte a edição textual ao código base NCL e ao objeto de mídia escrito em Lua;

d) explorar os recursos do *framework* LuaOnTV;

4.5.1 Arquitetura do LuaComp

Uma ferramenta de autoria para aplicações procedurais ou imperativas raramente consegue criar aplicações com 100% de não edição pelo autor, mas o LuaComp permite, na maioria das vezes, que um autor com um perfil de não saber lógica de programação possa implementar uma aplicação completa. Por exemplo, uma aplicação com interatividade para acesso a noticias especializadas utilizadas no sistema SKY conforme a Figura 4.3 a seguir:

Figura 4.3 – Interatividade SKY
FONTE: Disponível em: http://www.sky.com.br/sonasky/interatividade. Acesso em: 2 mar. 2009.

O LuaComp, como uma ferramenta de autoria, disponibiliza diversos recursos para facilitar a produção do autor. O Luacomp, além de disponibilizar os componentes gráficos, também disponibiliza as visões textuais, de leiaute e estrutural. Nas figuras a seguir são apresentadas as visões do LuaComp.

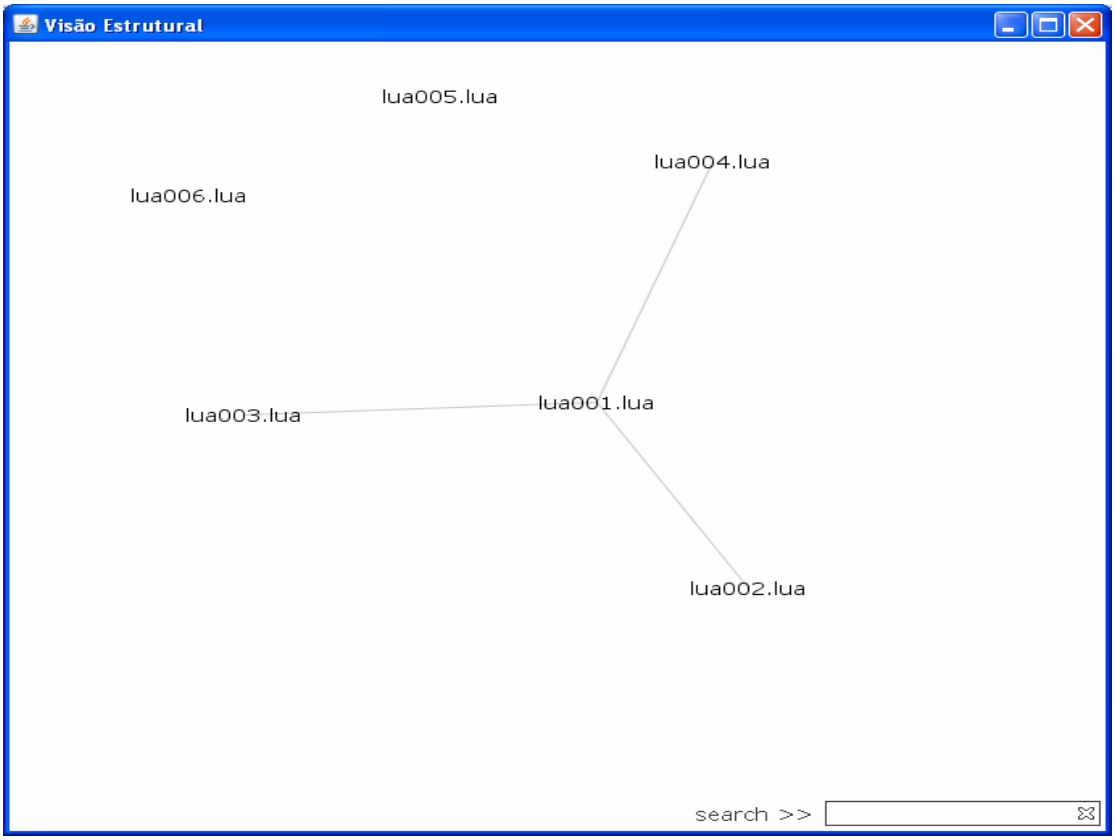


Figura 4.4 – LuaComp – Visão estrutural

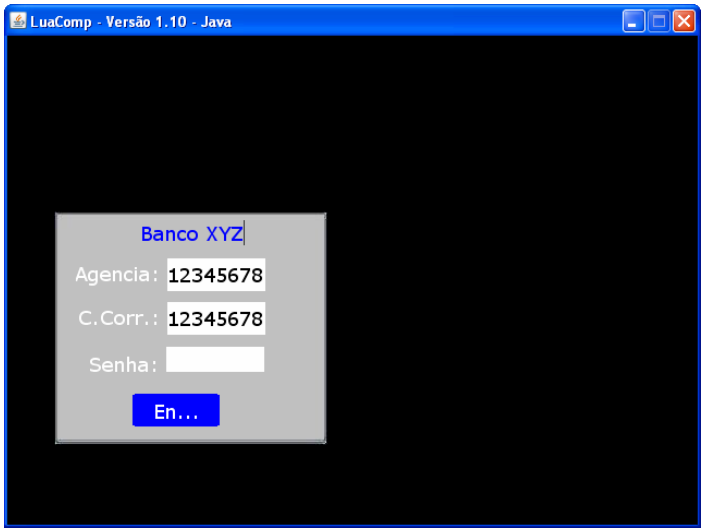


Figura 4.5 – LuaComp – Visão leiaute

Figura 4.6 – LuaComp – Visão textual

A visão estrutural permite que o autor crie, salve e altere seu projeto de páginas, facilitando a manutenção. A visão de leiaute é a principal da ferramenta do LuaComp, pois é a base para a construção do *design* da página e permite a edição WYSIWYG com relação ao emulador do Ginga-NCL. Este recurso pode ser observado na figura 5.9. A visão textual serve com alternativa para modificações e adaptações feita por um autor experiente e conhecedor tanto do NCL como do NCLua.

A visão espacial/temporal não é necessária para o LuaComp, pois seu foco é a criação de aplicativos imperativos que também são objetos de mídia em Lua utilizando a biblioteca NCLua.

Apesar da visão espacial/temporal não existir explicitamente, o LuaComp possibilita a edição do sincronismo da mídia NCLua com o código principal NCL. Um NCLua deve ser escrito em um arquivo separado do documento NCL, que apenas o referencia. Um código NCL acessa um NCLua conforme linha XML a seguir:

<media id="lua" src="lua001.lua" descriptor="dsLua">

A arquitetura do LuaComp tem como diferencial o registro de todo o projeto através de arquivos xml. A capacidade de gravar e recuperar todo o projeto em XML possibilita ao autor criar e utilizar *templates* de tipos de aplicações. Uma produção para TV Digital não é um tarefa simples e precede um projeto visual e até um numero grande de páginas ou hipertextos. Uma ferramenta de autoria que possibilite a continuação do trabalho facilita a produção e o acompanhamento do projeto. Mais que salvar um projeto completo, o LuaComp permite a manutenção de toda a aplicação. A possibilidade de todo o projeto se basear em arquivos XML também possibilita a modificação diretamente no arquivo XML. A Figura 4.7 demonstra a arquitetura do LuaComp.

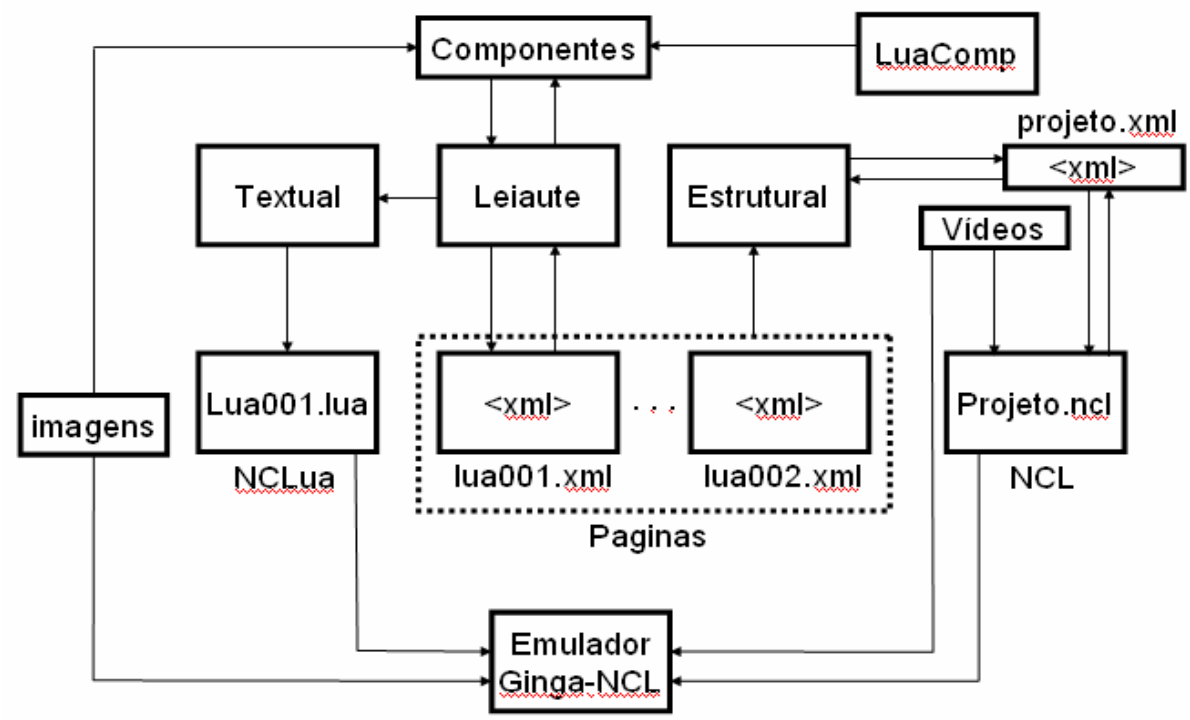


Figura 4.7 – Arquitetura LuaComp

O LuaComp é um *software* desenvolvido pela linguagem Java e, por ser uma linguagem de bastante uso comercial para sistemas de entrada e saída de dados, utiliza os mesmos componentes gráficos implementados no LuaOnTV. Para facilitar a edição destes componentes o LuaComp teve a difícil tarefa de disponibilizar apenas as características e funcionalidades implementadas no LuaOnTV.

Quando o autor utiliza o LuaComp arrastando e colando os componentes gráficos, é disponibilizada uma janela chamada inspetor de objetos que permite ao autor alterar os atributos e os principais eventos do usuário.

4.6 CONCLUSÃO

Independente das vantagens e facilidades de usar uma linguagem não sistêmica ou de *script* como a linguagem Lua, este capítulo procurou destacar os recursos já disponíveis no Ginga e incentivar a pesquisa e o uso do Ginga-NCLua. O amadurecimento e o mercado para o Java são incontestáveis. Mas, uma linguagem leve, com os mesmo recursos de Java, também é um motivo muito importante para se aderir à linguagem Lua.

Outro fator importante na escolha do Ginga-NCLua são as ultimas entrevistas do pai do Ginga-NCL, Luis Fernando Soares, que ressalta a importância das aplicações procedurais, mesmo sendo também um dos autores do *Composer* que é uma ferramenta de autoria para apliacações 100% declarativas. O LuaComp surge como uma ferramenta de autoria para aplicações híbridas por também implementar o NCL.

O LuaComp também é uma oportunidade de se testar o LuaOnTV, um *framework* implementado em Lua e com o paradigma da programação orientada a objetos. O LuaOnTV é um esforço deste autor junto com alunos de graduação, e sob a orientação do professor Paulo R. L. Gondim, em desenvolver uma solução POO para uma linguagem estruturada. Este *framework* foi publicado e disponibilizado como um *software* GPL. O surgimento de uma ferramenta que teste exaustivamente o LuaOnTV, o LuaComp, cria um novo modelo de produto. O LuaComp aponta para novas versões com uso de novos *frameworks* Lua.

5 UTILIZANDO O LUACOMP

O LuaComp é um ambiente de autoria implementado em Java, que visa auxiliar autores com níveis variados de conhecimento na construção de aplicativos procedurais ou imperativos de TV Digital interativa para o Ginga-NCLua. O LuaComp disponibiliza os principais componentes gráficos conforme a Figura 5.1 a seguir.

Figura 5.1 – Componentes do LuaComp 1.0

Neste capítulo são apresentados detalhes de implementação, inovações e os recursos básicos apresentados nas principais ferramentas de autoria. Também é apresentada uma análise comparativa entre as ferramentas de autorias pesquisadas.

5.1 INTERFACE GRÁFICA

A ferramenta de autoria LuaComp, na Figura 5.2, é composta inicialmente por uma barra de menus, palheta de componentes e inspetor de Componentes. A palheta de componentes representa todos os recursos gráficos para implementação de uma página NCLua no LuaComp 1.0. No gerenciamento da produção da aplicação para TV Digital interativa, o LuaComp disponibiliza as visões textual, estrutural e leiaute, conforme a Figura 5.2.

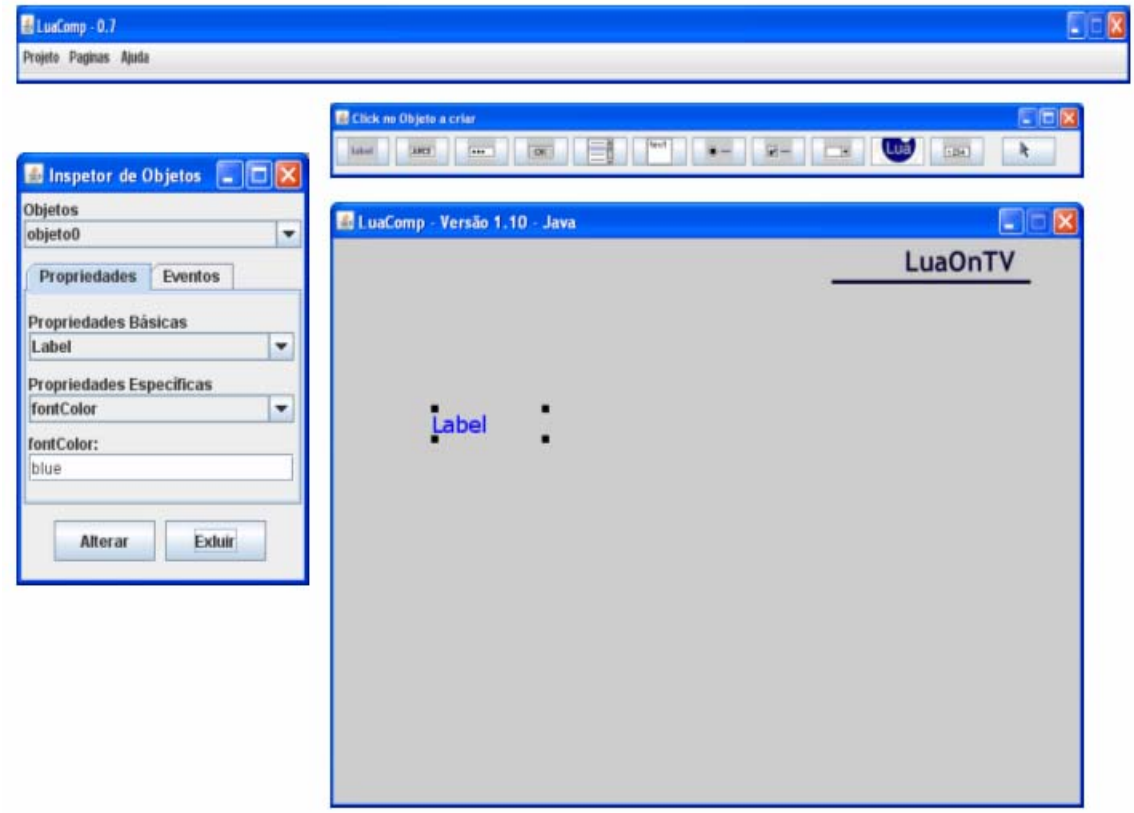


Figura 5.2 – LuaComp

A barra de menus disponibiliza as principais operações para criação das páginas NCLua, do projeto e do código NCL. O autor pode criar uma página com uma imagem base de fundo ou inserir várias imagens junto com os outros componentes de entrada e saída de dados. Observe na Figura 5.3 que as operações disponíveis são básicas. A barra de menus também possui uma opção sobre o LuaComp e um tutorial para facilitar a adaptação do autor a seus recursos.

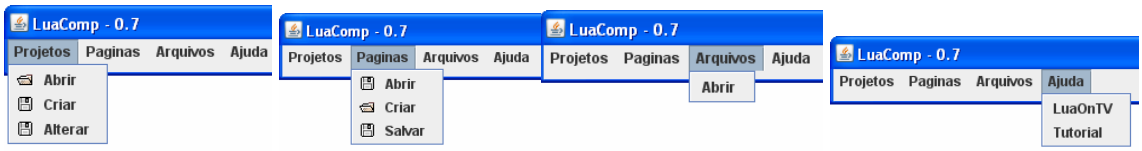


Figura 5.3 – Barra de menus LuaComp

O autor pode optar por iniciar criando páginas ou partindo de um projeto. Se as páginas dos projetos já forem criadas e com *link* entre elas, a visão estrutural apresenta os *links*, se não, as páginas ficam isoladas. Quando se cria uma página e adiciona-se *link* através de um componente, o LuaComp adiciona no XML da página este *link*. Quando se cria ou se

altera o projeto, a visão estrutural é refeita. A visão estrutural ajuda o projetista a encontrar a página que está solta no projeto.

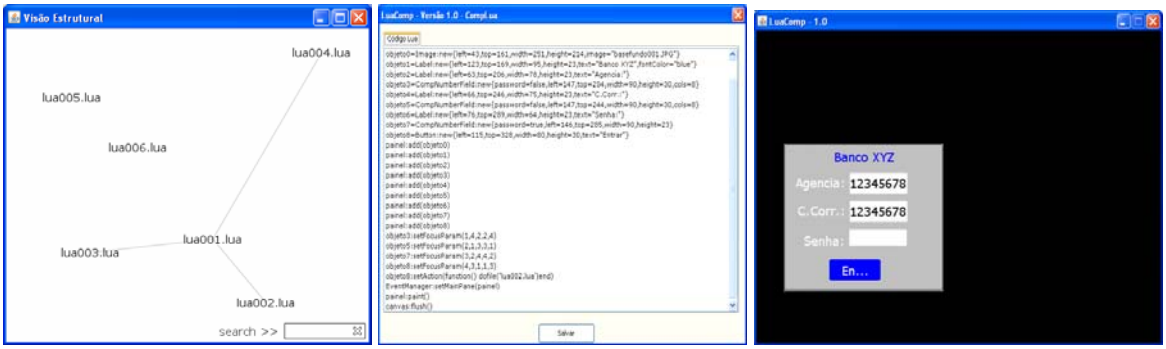


Figura 5.4 –Visões estrutural, textual e leiaute - LuaComp

Quando o autor deseja criar um projeto, ele entra com as informações básicas que servirão para construção do código principal NCL. Neste recurso o autor define: o nome do projeto que será também o nome do arquivo NCL, o vídeo ou programa que será a mídia principal do código NCL, o tempo em segundos para a autodisponibilização da mídia Lua que é a página principal da aplicação NCLua, as páginas que comporão todo o projeto iniciado pela página principal Lua, e finalmente o leiaute da aplicação, escolhendo o posicionamento do programa. O *Layout* X também disponibiliza a inserção de todo os componentes com fundos transparentes, ou seja, o programa tomando toda a tela e a aplicação NCLua por cima e com fundo transparente. Caso o autor deseje adicionar ou retirar alguma página do projeto, a opção Alterar do menu Projetos possibilita este recurso.

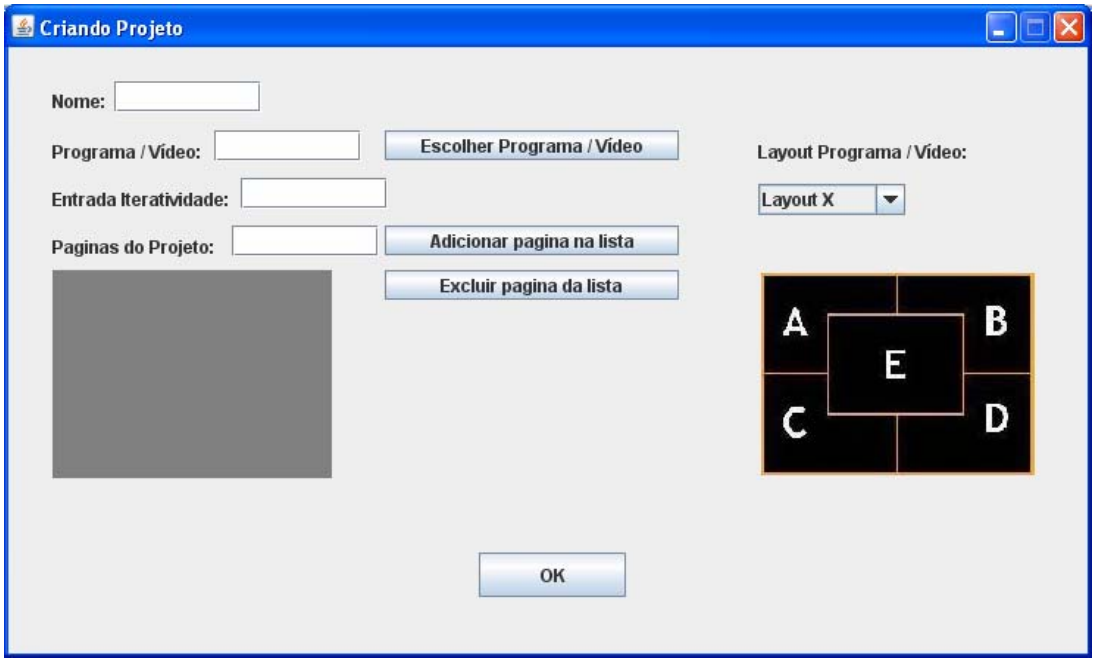


Figura 5.5 – Módulo de criação de projeto

Na implementação das páginas Lua, o autor pode criar uma página com imagem com tamanho máximo no fundo conforme a Figura 5.6 ou sem imagem e inserindo várias imagens.

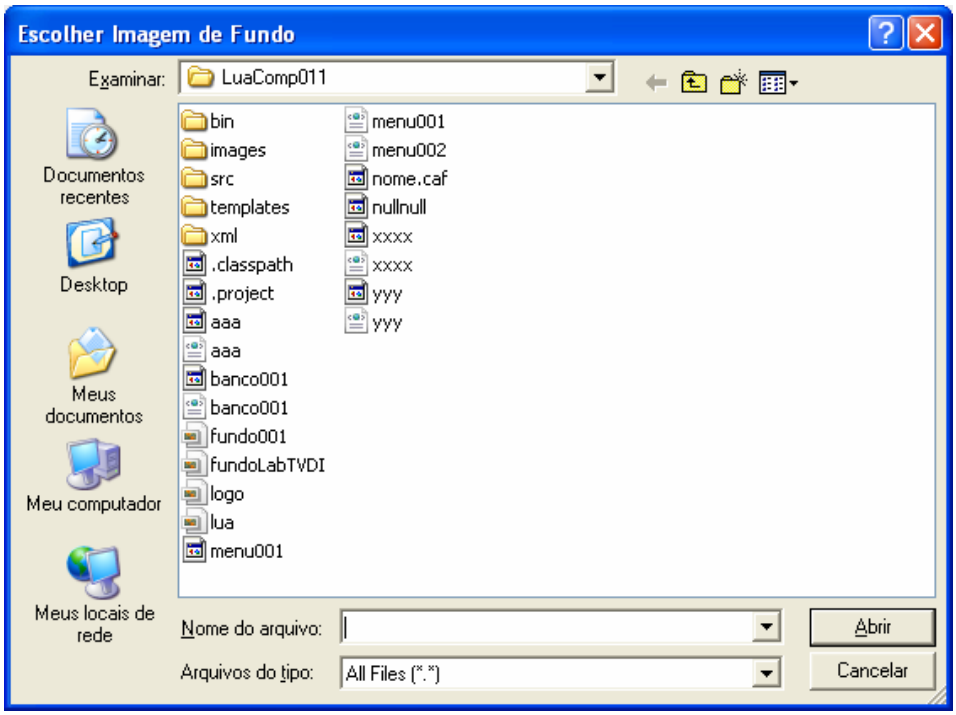


Figura 5.6 – Imagem de fundo - LuaComp

Na inserção dos componentes gráficos, o LuaComp faz o *link* entre os componentes da linguagem Java com o LuaOnTV através do inspetor de componentes, Figura 5.7. O LuaComp também possibilita que algumas alterações como alteração do texto e tamanho de altura e largura sejam feitas com o *mouse* diretamente no formulário de edição. As propriedades básicas de cada componente são os atributos da superclasse *DefaultComponent* do LuaOnTV. A aba de eventos é a responsável pelo tratamento dos principais eventos de interatividade com os componentes gráficos. No caso da versão 1.0 do LuaComp, o único evento tratado é o "*enter*". O evento *enter* possibilita a navegação do usuário e também pode ser utilizado para validar as informações de entrada, como, por exemplo, numa aplicação T-*Bank*.

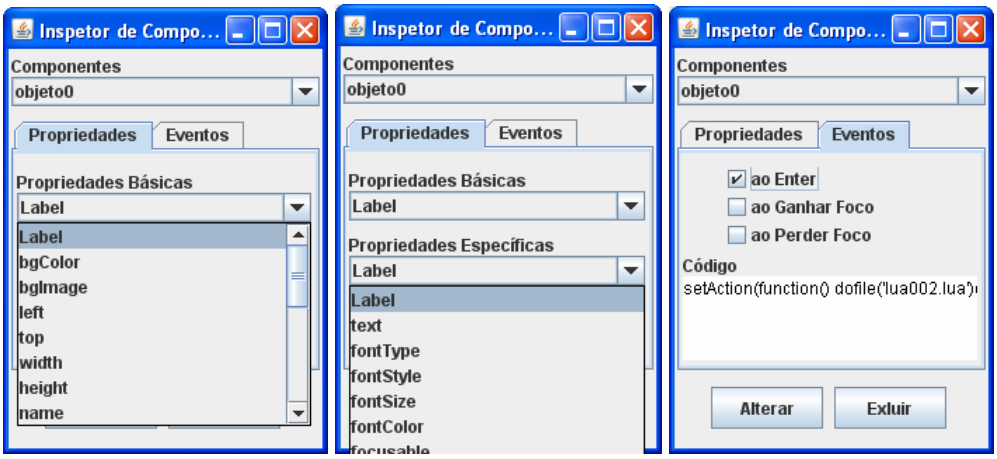


Figura 5.7 – Inspetor de componentes - LuaComp

Após a criação da página, o LuaComp possibilita que o autor salve a página para teste no emulador e guarde a página em arquivo XML para futura continuação ou uso como um *template*.

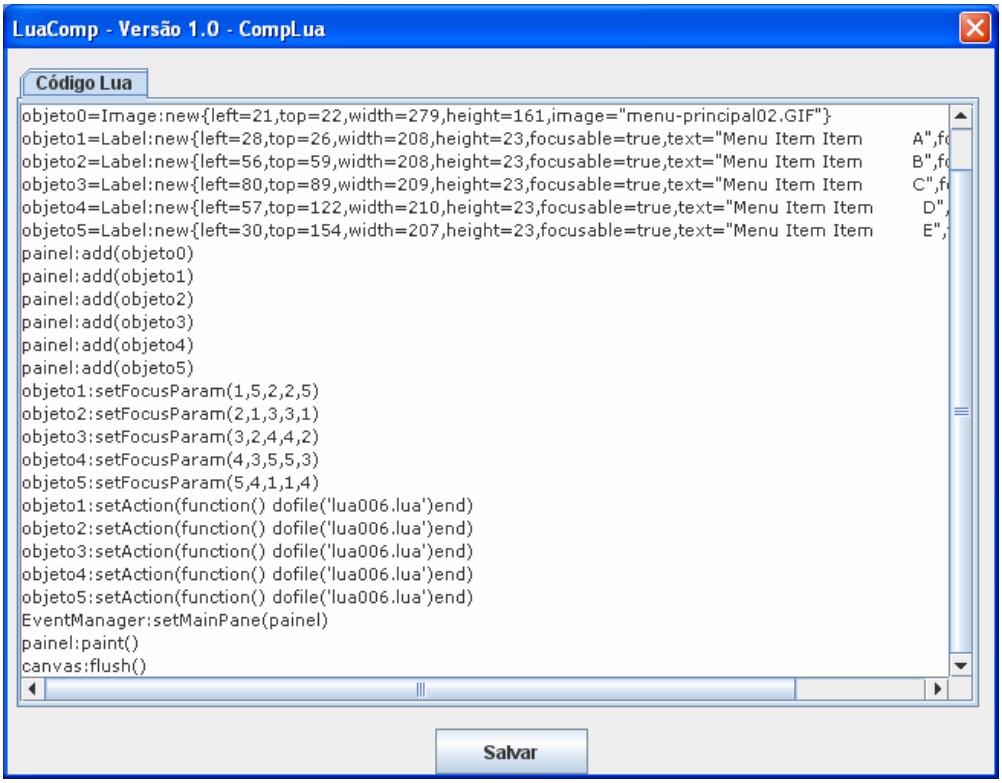


Figura 5.8 – Salvando uma página - LuaComp

5.2 VISÕES

A maioria das ferramentas de autoria para documentos hipermídia trabalha com o conceito de visões. Estes conceitos se encaixam perfeitamente para ambientes hipermídia e não são utilizados nas outras ferramentas de autoria para sistemas como o *Delphi*, *Visual Basic* e outros. O *Composer*, apesar de também ser uma ferramenta de autoria para TV Digital, não utiliza a nomenclatura de visões. O LuaComp resolveu adotar também o conceito de visões como forma de aproximar o linguajar dos autores que provavelmente tem mais experiência em aplicações hipermídia e usarão o LuaComp como um recurso para implementação de aplicações imperativas.

5.2.1 Visão de leiaute

A visão de leiaute permite ao autor movimentar os componentes gráficos escolhidos no menu ou palheta de componentes de forma que a sua posição, altura, largura, cores e outros atributos básicos fiquem representados da mesma forma que a aplicação no emulador

Ginga-NCL. Observe na a Figura 5.9 a seguir que o LuaComp possibilitar a edição do modo WYSIWYG.

Figura 5.9 – Emulador Ginga x LuaComp

5.2.2 Visão estrutural

No LuaComp, a visão estrutural tem como objetivo principal dar ao autor a visão de hipertextos de todo o projeto da aplicação. Para a manutenção e continuidade de um projeto com inúmeras páginas, a visão estrutural facilita a produção. Observe na Figura 5.10 que o LuaComp mostra as páginas ou que ainda não foram implementadas ou que ainda não foram *linkadas*.

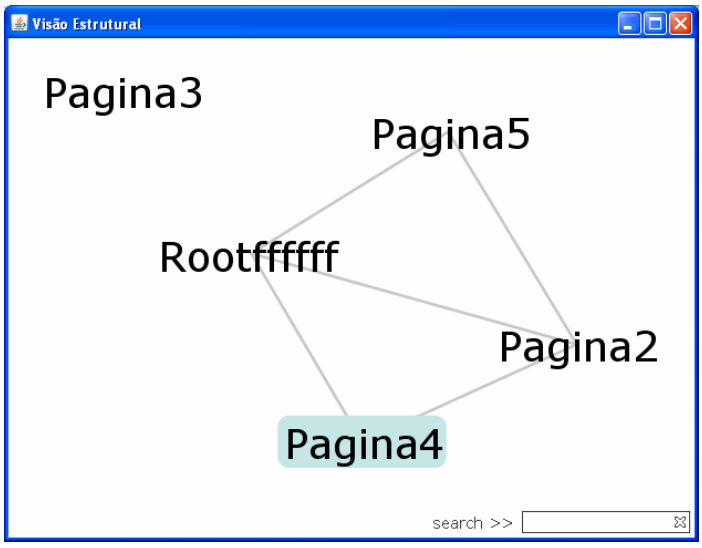


Figura 5.10 – Visão estrutural

5.2.3 Visão textual

A visão textual permite ao autor modificar os códigos tanto do NCL como do NCLua. Uma aplicação pode ser recarregada e posteriormente ter apenas seu código modificado pelo autor. As visões textuais são normalmente utilizadas por profissionais pouco mais familiarizados com as linguagens NCL e NCLua. Importante destacar que a capacidade de modificação do NCLua está associado ao conhecimento do *framework* LuaOnTV. Observe na Figura 5.11 a seguir as duas visões textuais.

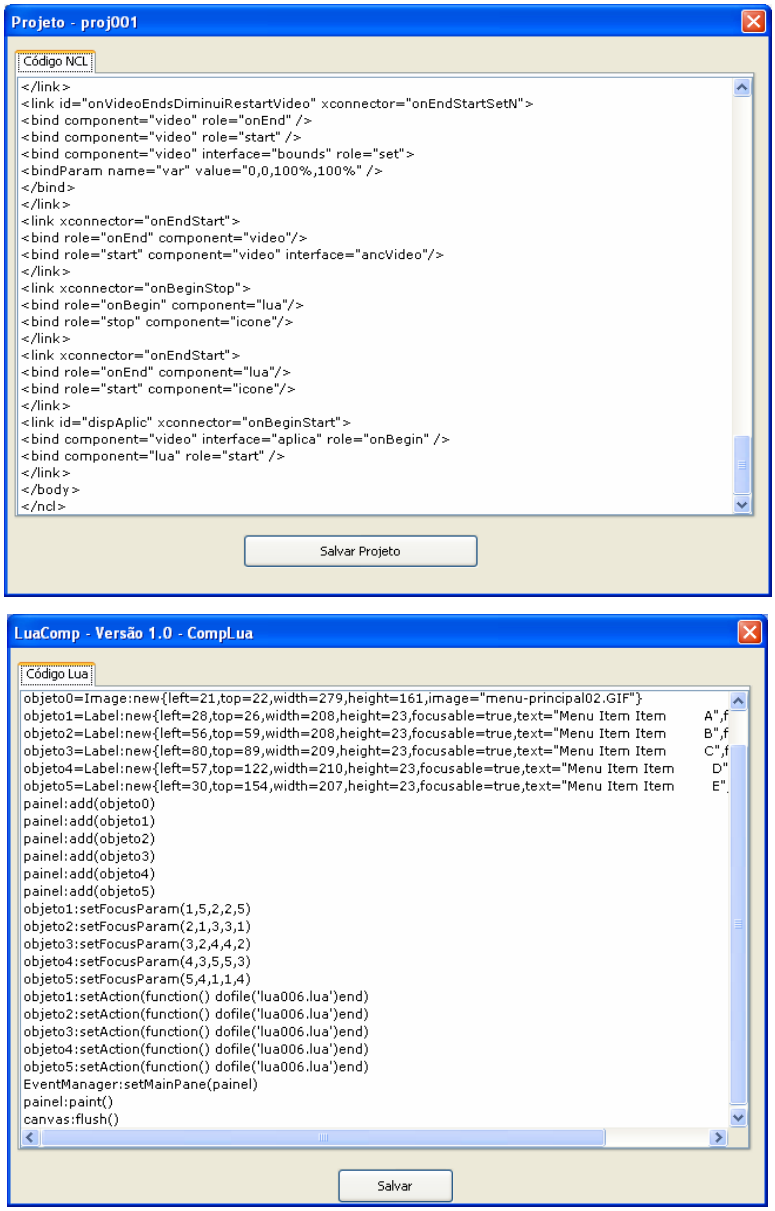


Figura 5.11 – Visão textual

5.2.4 Visão temporal

A visão temporal é bastante necessária quando se usa de um sincronismo de mídias, figura 5.12 do *Composer*. Em comparação com a Composer, o LuaComp necessita apenas de se programar o início e o fim do sincronismo da aplicação com relação à mídia ou programa principal da operadora de broadcast. Na Figura 5.12, O LuaComp é utilizada para se gerar o projeto e o código NCL. È importante observar na Figura 5.12 do *Composer* que esta visão não faz falta para uma ferramenta de autoria como o LuaComp, que não se propõem sincronizar várias mídias.

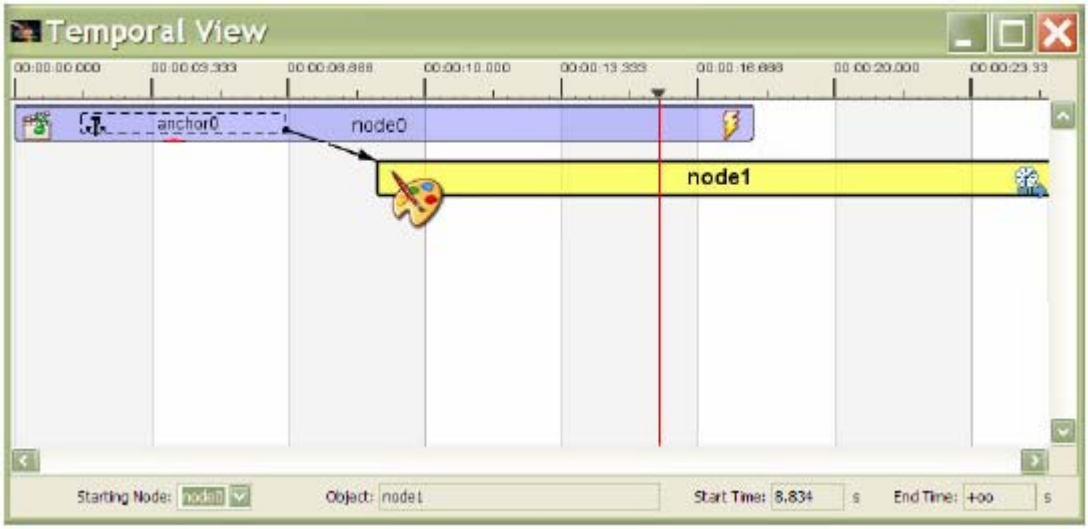


Figura 5.12 – Visão temporal - *Composer*
FONTE: GUIMARÃES, 2007. [69]

5.3 ARQUITETURA DO LUACOMP

O LuaComp é um software desenvolvido na linguagem Java e que gera aplicações híbridas em NCL e Lua com base no framework LuaOnTV. O LuaOnTV, por ser um projeto livre e com uma arquitetura bastante madura, poderá servir, a curto prazo, como modelo para implementação de aplicações para o Ginga-J.

A arquitetura do LuaComp, na Figura 5.13, não apresenta nenhuma relação de herança, mas principalmente associação e dependência de classes, pois o objetivo não é de servir como uma API e sim disponibilizar todos os recursos para uma ferramenta de autoria.

Quando o LuaComp é executado, a classe Principal instancia os objetos das classes PaginaLua, Menus e Palhetas. A classe Principal é associada por composição às classes: Paginalua, formulário principal, Menus, menu principal e, Palhetas, barra de componentes.

A classe PaginaLua é associada à classe FundoAppUI para possibilitar a manipulação do componente do tipo imagem e o ArrayList, que guardará todos os objetos dos componentes gráficos. Quando o autor faz um *drag and drop* de um componente disponível no objeto da classe Palheta, ou seja, clica, arrasta e cola na um componente na PaginaLua, a classe CriaComponente instancia um objeto referente a classe do componente gráfico escolhido na Palheta. A classe PaginaLua também é associada a classe AjustaComponente.

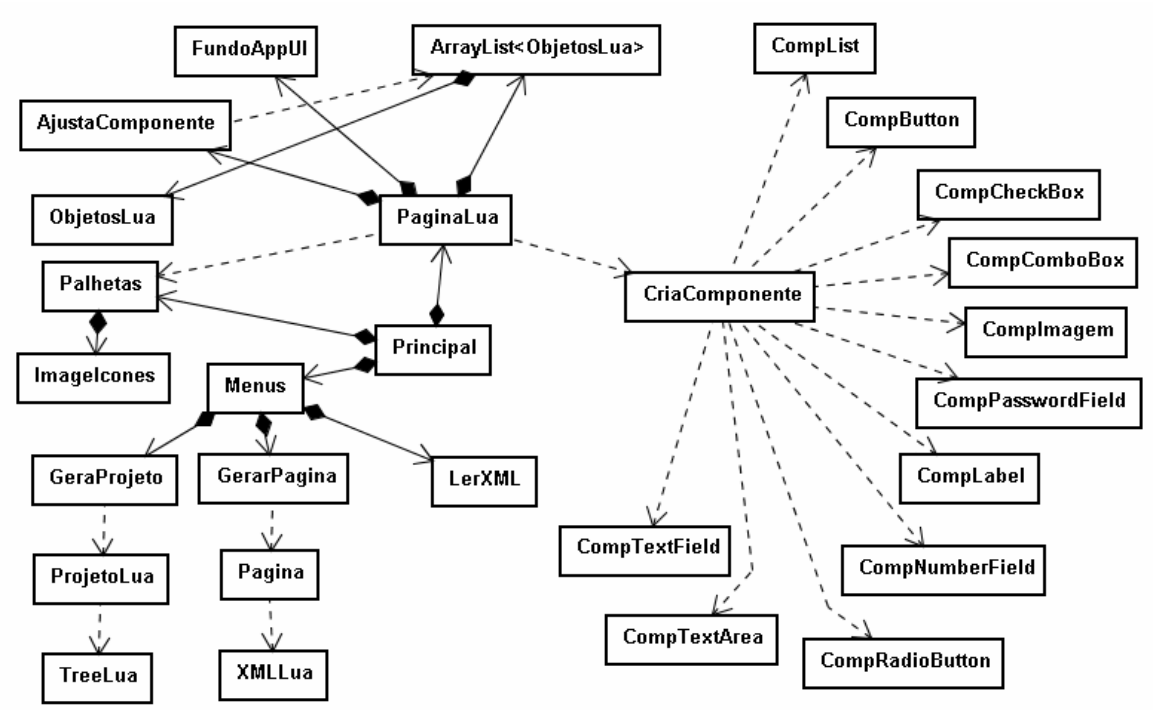


Figura 5.13 – Diagrama de classes LuaComp

Observe que o LuaComp tem todos os componentes gráficos representados pelas classes Comp. A classe CriaComponente é a responsável pela instância destes objetos e também pela inserção em um *ArrayList* que guarda todos os objetos criados. O *ArrayList* é compostos por objetos da classe ObjetosLua que agrega todas as propriedades dos componentes gráficos. A associação dos componentes gráficos com os componentes do *framework* LuaOnTV é feito nas próprias classes Comp. Qualquer modificação nos componentes do *framework* LuaOnTV deve ser refletida nas classe Java dos componentes do LuaComp

Durante a adição dos componentes gráficos na página, o inspetor de componentes fica ajustado para alterar as características visuais e não visuais dos componentes escolhidos. A classe AjustaComponente possibilita a modificação dos componentes gráficos e a alteração

dos atributos relacionado com o LuaOnTV. O AjustaComponente representa o inspetor de componentes. A classe AjustaComponentes é associada à classe Menus.

Quando o autor deseja criar, salvar ou abrir um projeto ou uma página, o LuaComp executa as classes: GeraProjeto, GeraPagina e lerXML através da classe Menus. A classe Menus é associada por agregação a estas classes. A classe GeraProjeto é responsável pela gravação do projeto e do código NCL. A classe GeraPagina também funciona da mesma forma que a classe GeraProjeto, mas gerando os arquivo NCLua. A classe LerXML é a responsável por abrir as páginas e a classe TreeLua é a responsável pelas operações de persistência do projeto.

5.4 ESTUDO DE CASO

Nesta subseção será apresentada uma aplicação disponibilizada no site Broadbandbananas [58] onde o LuaComp utilizará como material para implementação.

Na figura 5.14, a aplicação vai disponibilizar imagens, o programa principal, os botões de menu e as caixas de texto. Cada opção do menu chamará outras telas.

Figura 5.14 - Modelo para estudo de caso [58]

Esta aplicação é composta por 4 imagens que podem ser preferencialmente jpg, gif e png, por terem um tamanho menor que as outras mídias. Os botões de acesso às outras páginas podem ser imagens também, mas o uso do componente gráfico botão ajuda a diminuir o tamanho da aplicação. O programa ou vídeo principal é disponibilizado no canto superior direito.

Para iniciarmos a construção da aplicação, o LuaComp disponibiliza uma janela de menu, Figura 5.15, com a opção do autor iniciar pela construção do projeto com suas páginas e links, ou iniciar pela construção de página à pagina.

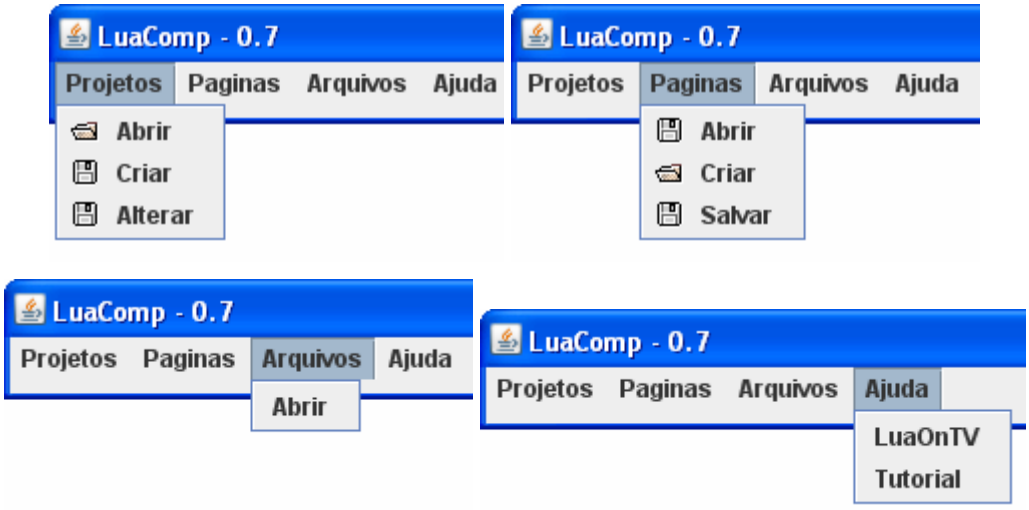


Figura 5.15 - Janelas de Menus

O projeto de toda a aplicação pode ser gerado conforme Figura 5.16 Se, antes de construir as páginas, o autor optar por montar todo o projeto com suas páginas e links, ele terá que alterar o arquivo XML gerado para configurar os links. A Figura 5.16 mostra um projeto com as páginas sem seus links e com seus links. Observe também que se o autor inserir uma página no projeto que não tenha link, o LuaComp apresenta a página sem link. O LuaComp também possibilita o autor alterar o projeto inserindo ou excluindo uma nova página.

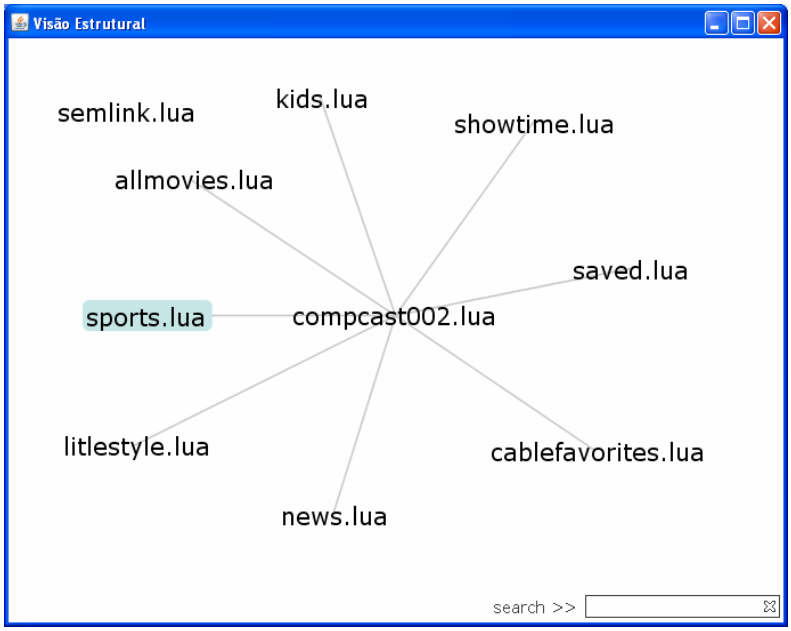


Figura 5.16 - Janela do projeto da aplicação

A vantagem de construir as páginas antes de construir o projetos é que ao criar o projeto, o LuaComp monta os links automáticamente. Para isso, o autor deve alterar o código Lua na opção evento dos botões da aplicação conforme mostra a Figura 5.17.

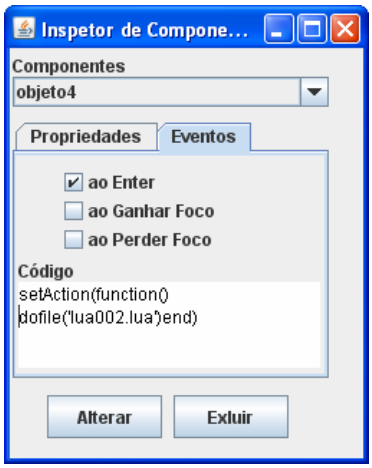


Figura 5.17 - Janela do inspetor de componentes

A Figura 5.18 apresenta a tela para criação do projeto. Observe que esta funcionalidade permite a escolha do programa principal, do sincronismo de entrada da aplicação com relação ao tempo em segundos, as páginas que comporão o projeto e a disposição de leiaute do programa de vídeo principal no código NCL. Quando o projeto é salvo o código NCL é gerado em um arquivo como o nome do projeto. O LuaComp também possibilita a posterior modificação do código através de sua visão textual disponibilizada no menu abrir arquivo e salvar da Figura 5.19.

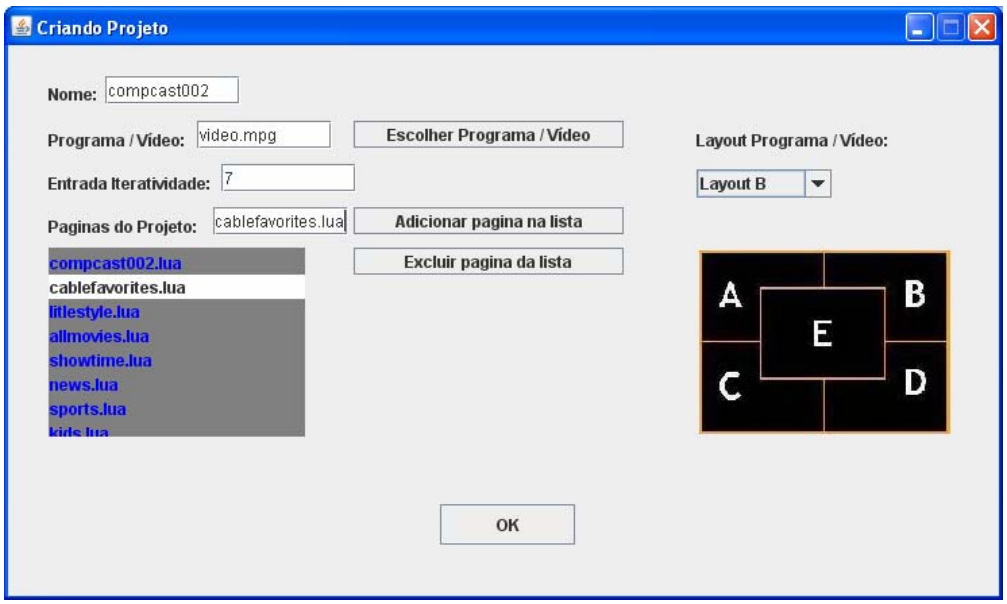


Figura 5.18 - Janela de criação de projeto

O código NCL pode ser modificado pela visão textual conforme Figura 5.19. O LuaComp disponibiliza janelas para todas as opções de escolha de arquivos, Figura 5.20.

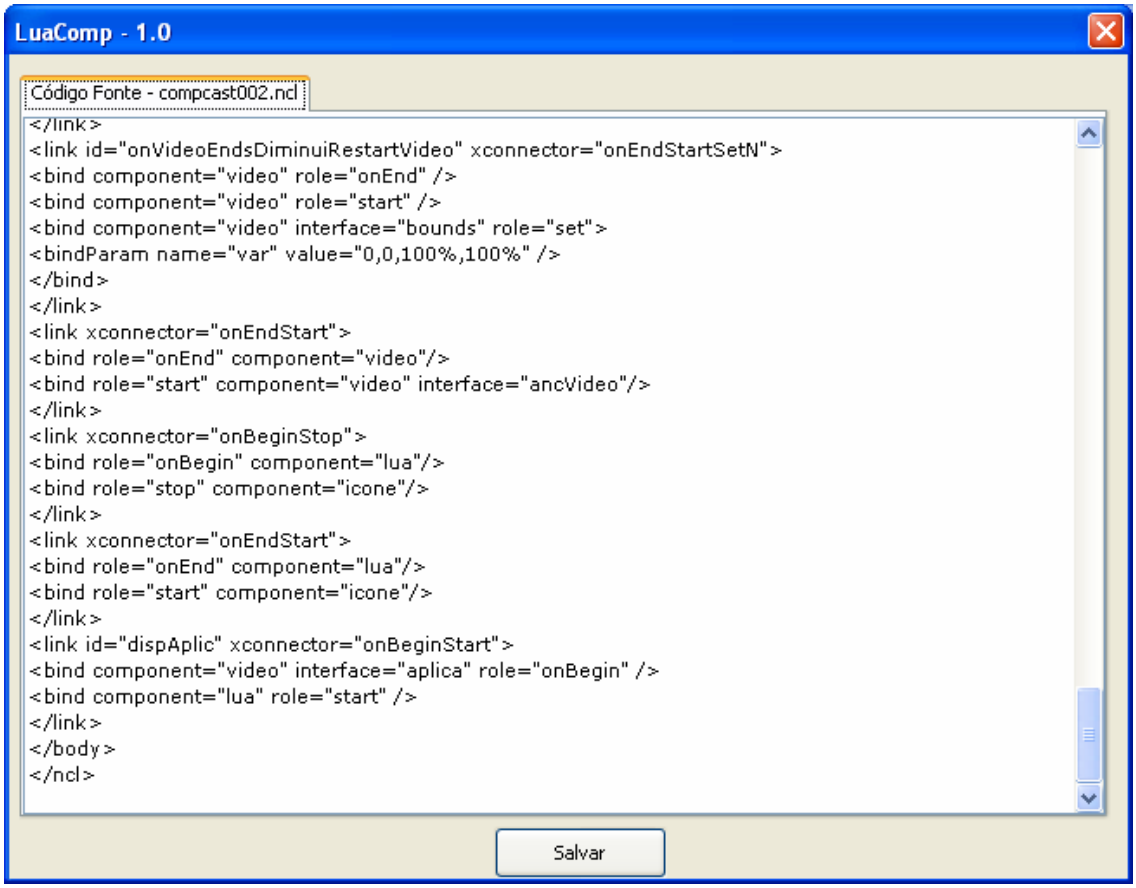


Figura 5.19 - Janela de visão textual

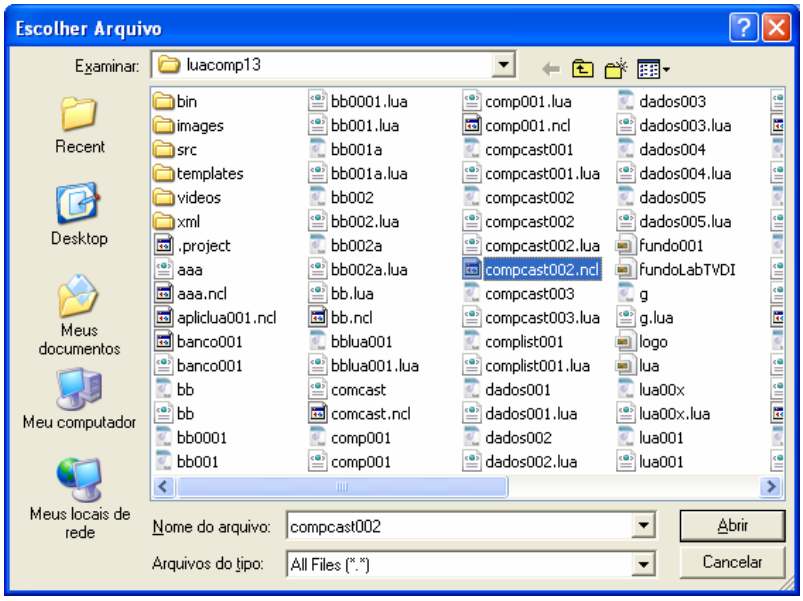


Figura 5.20 - Janela de escolha de arquivos

Voltando a seqüência de construção da aplicação, a Figura 5.20 apresenta as principais janelas para implementação de uma página NCLua.

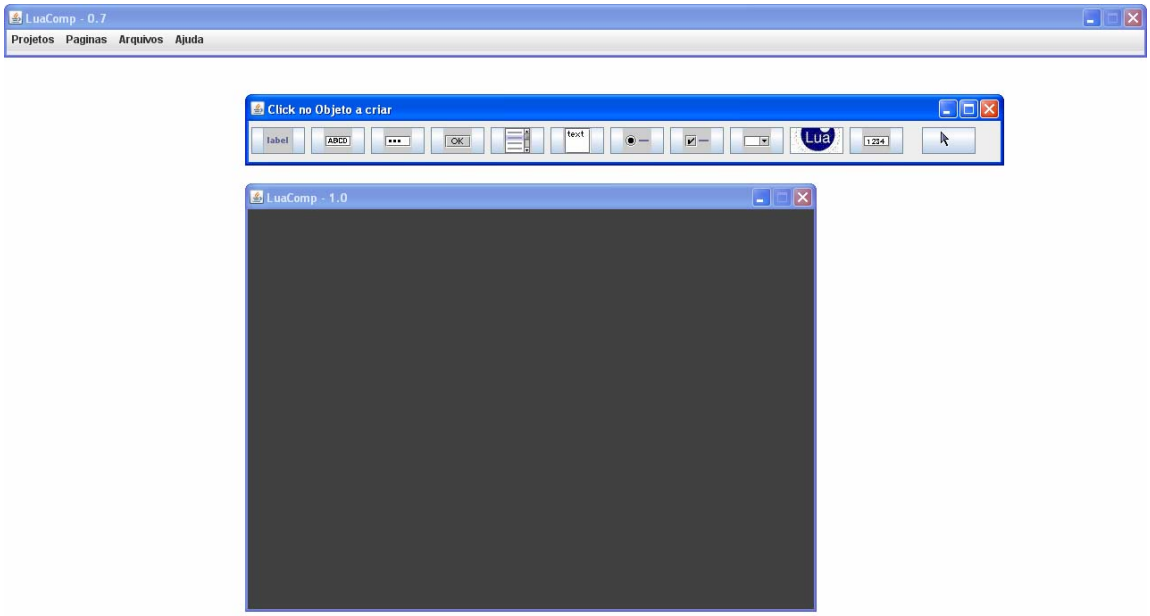


Figura 5.21 - Janelas para implementação

Para o modelo utilizado nesse estudo de caso, Figura 5.14, o autor escolheu inserir primeiro as imagens conforme a Figura 5.21. O autor simplesmente clica no ícone da imagem Lua e posiciona no painel, deixando para depois a configuração do posicionamento, onde ele altera a posição left, top da imagem.

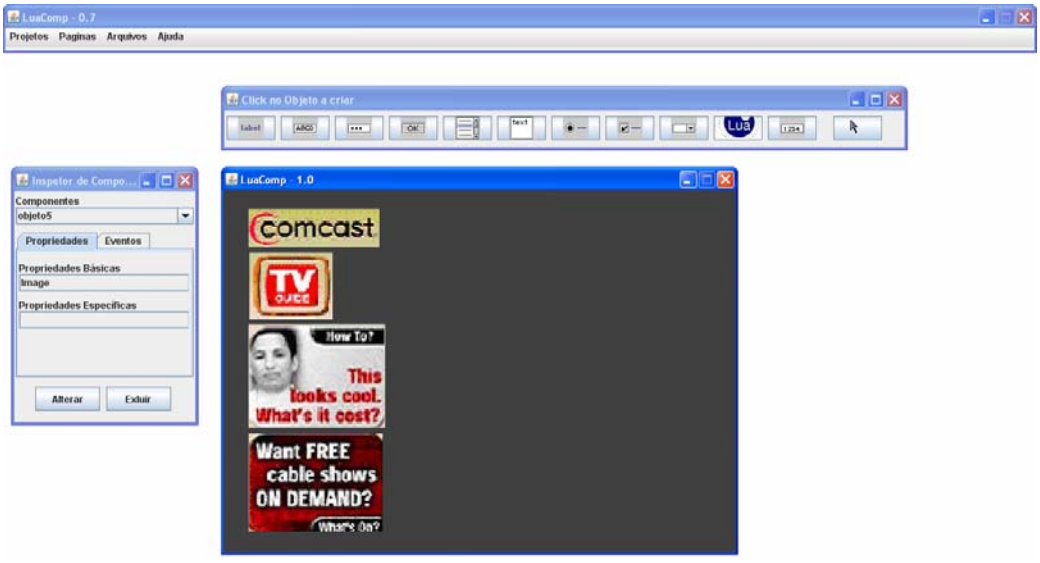


Figura 5.22 - Inserindo imagens no painel da página

Cada componente gráfico do LuaComp apresenta na janela Inspetor de Objetos todos os seus atributos ou propriedades que podem ser modificados e seus possíveis eventos tratáveis. A Figura 5.22 mostra todos os atributos e eventos de cada componente gráfico.

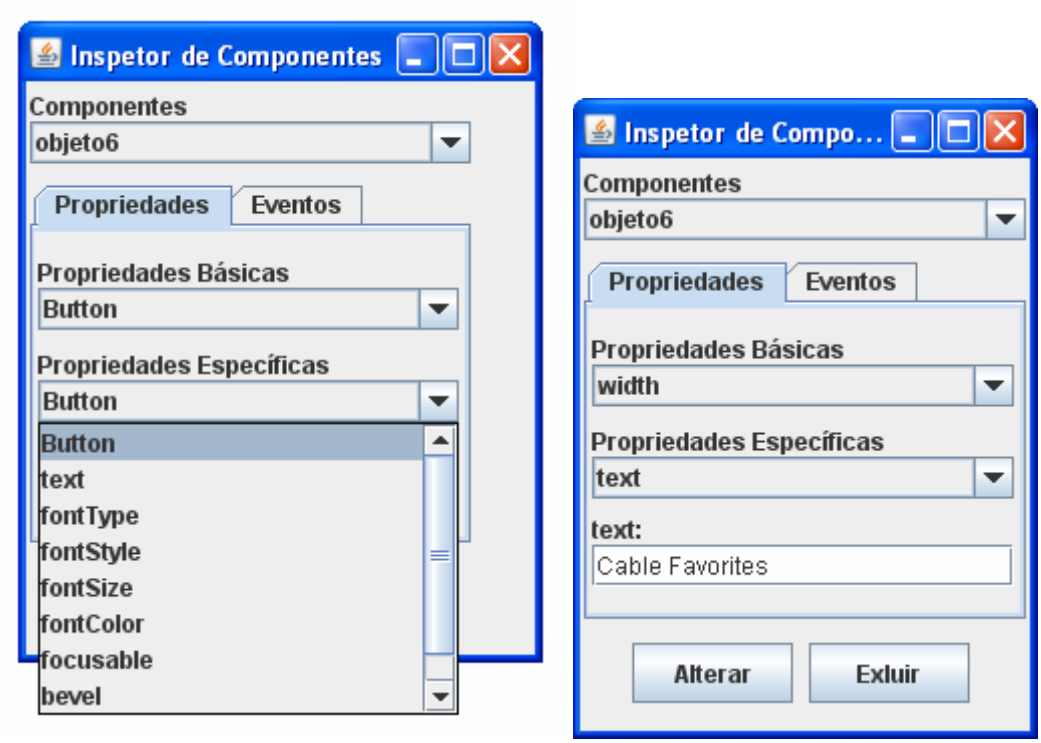
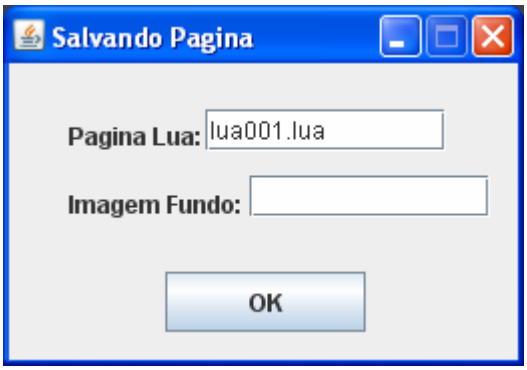


Figura 5.23 - Inspetor de componentes com seus atributos e eventos

Depois de o autor inserir os componentes de imagens, ele passa a inserir também os componentes do tipo botão e que servirão de links para as outras páginas da aplicação. Observe que na Figura 5.24, para ganhar tempo, o autor insere todos os botões e depois passa a modificar seus atributos na janela Inspetor de Objetos. A Janela Inspetor de Objetos, também possibilita a escolha do objeto a ter seus atributos modificados, sem precisar que o autor mexa nos componentes diretamente no painel.

Figura 5.24 - Inserindo botões

Após a inserção de todos os componentes, o autor pode salvar a página e se precisar de uma futura modificação, a visão textual dos arquivos XML facilitam uma rápida modificação e servem como atalho para ganho de tempo. A Figura 5.25 apresenta estas funcionalidades.



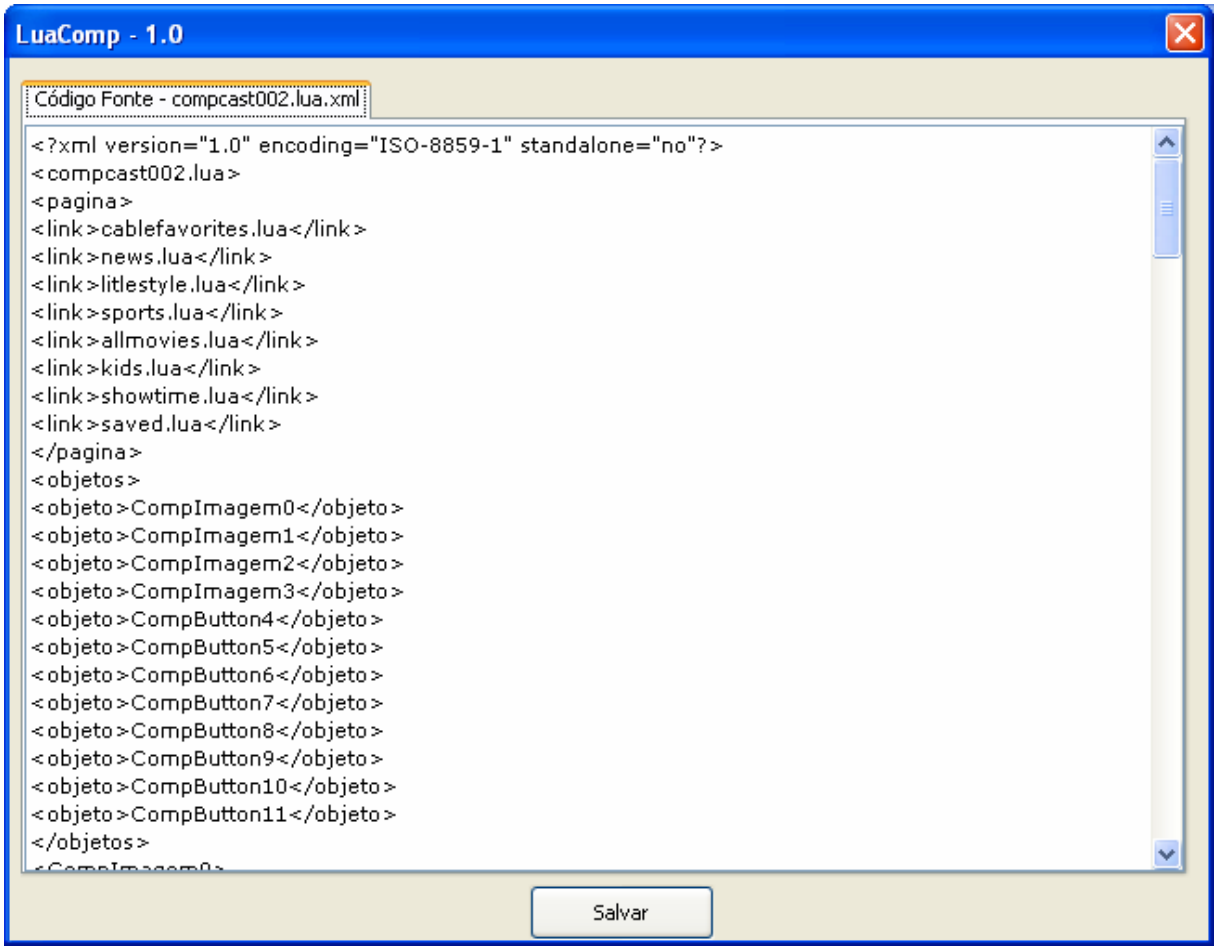


Figura 5.25 - Salvando uma página

Após essas ultimas alterações, o autor pode inserir a imagem de fundo e testar a aplicação no emulador Ginga-NCL. Observe na Figura 5.26 a seguir que o LuaComp possibilita a implementação WYSIWYG das páginas.

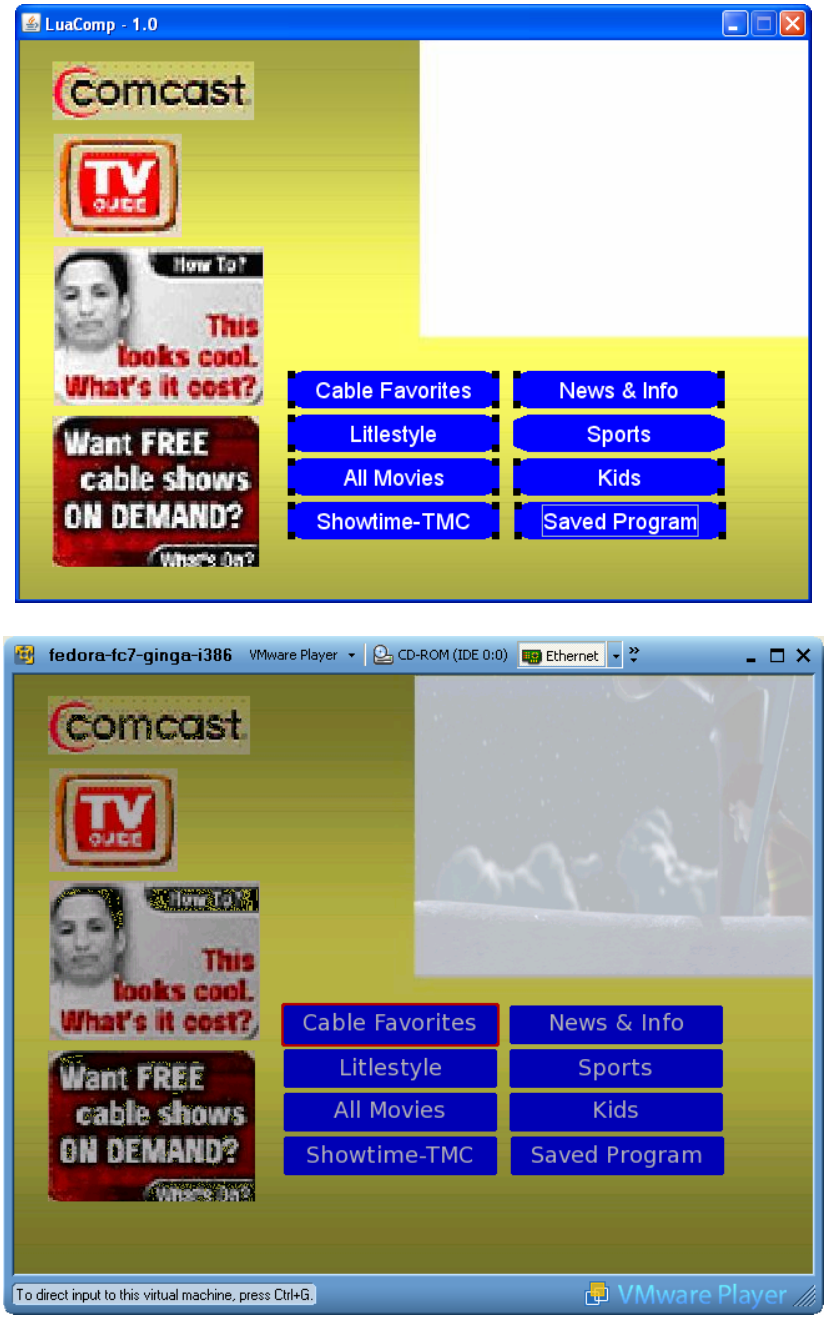


Figura 5.26 - Resultado final WYSIWYG

O LuaComp é uma ferramenta de autoria que possibilita uma criação de aplicações para TV digital com um grau de produtividade considerável e com as principais funcionalidades das maiorias da ferramentas de autorias já desenvolvidas.

5.5 ANÁLISE COMPARATIVA

Esta subseção destaca as características das principais ferramentas de autoria comparando com o LuaComp.

As ferramentas *Delphi* e *Visual Basic* são as mais completas ferramentas de autoria. Estas ferramentas estão em desenvolvimento até hoje. Estas ferramentas foram as primeiras a disponibilizar os componentes gráficos e não gráficos através do arrastar e colar. O Eclipse e o *NetBeans* seguem as mesma linha do *Delphi* e *Visual Basic* e são referência como ferramentas livres. As ferramentas de autoria como o JAME AUTHOR, *AltiComposer*, *Composer* são voltadas para edição de documentos hipermídias. As ferramentas de autoria para *Web* como o GRiNS, *LimSee2, Flash, DreamWeaver* e *FrontPage,* não podem ser desprezadas. A interatividade foi bastante impulsionada pela Internet e a as aplicações para TV Digital interativa, devido a sua limitação de recursos, não deixa de ser um paradigma para os usuários de TV Digital interativa.

Em comparação com o *Delphi, Visual Basic,* Eclipse e *NetBeans,* o LuaComp atende aos recursos gráficos básicos como as barras de componentes gráficos, inspetor de objetos ou componentes. O que o LuaComp não disponibiliza são os componentes não gráficos e isto se justifica pela não disponibilidade do completo Ginga-NCL. O Eclipse disponibilizou um *plugin* para o NCL e o Lua, mas com nenhum recurso de componentes gráficos para geração da aplicação imperativa. Este *plugin* serve apenas como módulo compilador e serve como apoio para a visão textual.

O Macromedia Flash serviria apenas como comparativo na implementação de sincronismo espacial e temporal, mas o LuaComp não tem ao implementação do sincronismo espacial/temporal como foco. O Flash não é uma ferramenta desenvolvida pata TV Digital.

O JAME Author talvez seja a ferramenta com recursos, tanto para código procedural como declarativo, mais harmonizados e de bom entendimento pelo autor. Esta ferramenta foi uma ótima referência para a construção do LuaComp. A programação visual para o foco é melhor trabalhado que o LuaComp, pois utiliza grafos para descrever a seqüência de foco e o componente de *link* entre as páginas. Esta ferramenta gera códigos MHP/OCAP e possui emulador.

Importante destacar que a facilidade destas ferramentas gerarem emuladores é devida ao uso da linguagem Java e sua facilidade de criar *XletView*.

O *Cardinal Studio*, da mesma forma que o JAME AUTHOR, apresenta os mesmos recursos, o único problema é a dificuldade de acesso a esta ferramenta para teste, o que não acontece com o JAME AUTHOR, que disponibiliza uma versão teste e um tutorial que reflete seus recursos. O *Cardinal Studio* é uma referência, mas de muito difícil acesso. O cardinal gera aplicações DVB/MHP. O *Cardinal Studio* gera aplicações hipermídia com tratamento de sincronismo espacial/temporal, mas demanda um conhecimento avançado para configuração do sincronismo.

O *AltiComposer* é a ferramenta que mais se aproxima ao modelo de produção baseado no perfil de produção de conteúdo da indústria de TV e cinema. Além de usar o Java, esta ferramenta traz como novidade a possibilidade do autor criar seus próprios componentes gráficos. O LuaComp tem como solução o uso de imagens como forma de disponibilizar novos componentes. Os componentes disponibilizados pelo LuaComp são os únicos necessários para uma interação de entrada e saída de dados. Os padrões de usabilidade para aplicações de TV Digital são muitos limitados e com poucas variações.

Com relação às ferramentas de autoria que disponibilizam os componentes gráficos apresentados no LuaComp, o JAME AUTHOR, *Cardinal Studio, AltiComposer* e *Macromedia Flash* dão também suporte a maioria dos componentes.

As ferramentas de autoria GRiNS, *LimSee2* e *Composer* são voltadas à aplicações hipermídia na qual o GRiNS e *LimSee2* usam a linguagem SMIL, e o *Composer* o NCL. Importante destacar que as linguagens base para geração das aplicações limitam a capacidade das ferramentas de autoria, pois elas devem disponibilizar as funcionalidades de cada linguagem. O Capítulo 2 destacou as principais características das linguagens.

O *Composer*, como única ferramenta adaptada ao *middleware* brasileiro, serve de referência para o LuaComp para comparação com os recursos de *layout* da mídia principal. Como o NCLua pode resolver todos os eventos das mídia NCL, o exploração dos recursos gráficos do *Composer* são aproveitados apenas quando a aplicação possui várias mídia com interação. Ele possui um emulador, mas somente com o Ginga-NCL. O emulador Ginga-NCL do *Composer* não disponibiliza o Ginga-NCLua. Este recurso está disponível apenas no *settop box* virtual disponibilizado pelo laboratório Telemidia da PUC-RJ. O LuaComp 1.0 não definiu como requisito gerar um emulador devido a dificuldade de acesso ao NCLua e também a não disponibilização da versão final do NCLua para instalação em qualquer sistema operacional. O LuaComp optou em ser desenvolvido em ambiente Windows devido a

instabilidade e incompletude do NCLua disponibilizado pela PUC-RJ. Este software apresenta muitos recursos, mas disponibiliza apenas os básicos, diferentemente do LuaComp que prefere disponibilizar todos os apresentados.

O LuaComp, para suprir a necessidade de programação de parte declarativa, disponibiliza na criação do projeto a programação do tipo de *layout* para a mídia/programa principal e sincronismo para entrada da mídia NCLua. O LuaComp possibilita a alteração textual do código NCL, mas gera códigos declarativos que atendem perfeitamente a demanda para aplicações híbridas. Uma inovação importante com relação às outras ferramentas é o uso de arquivos XML para geração e manutenção de *templates*.

FERRAMENTA	DECLARATIVO	PROCEDURAL	EMULADOR	XML
Eclipse		X		
JAME AUTHOR		X	X	
Cardinal Studio		X	X	
AltiComposer				
GRiNS	X			
LimSee2	X			
Composer	X		X	
LuaComp	X	X	X(*)	X

X(*) – Utilizando emulador do Ginga-NCL (Ginga Emulator)

Tabela 6.1 - Tabela comparativa entre ferramentas de autoria para TV Digital

Pode-se destacar a capacidade do LuaComp gerar aplicações hibridas (declarativas e procedurais), NCL e NCLua, com um tamanho consideravelmente menor que qualquer outro tipo de ferramenta de autoria hoje disponível.

5.6 CONCLUSÃO

O LuaComp é uma ferramenta de autoria para aplicações híbridas desenvolvida em Java. O LuaComp gera aplicações híbridas com NCL e Lua. O LuaComp tem uma aparência muito parecida com o *Delphi* e o *Visual Basic*. A qualidade visual dos componentes gráficos depende da melhoria do *framework* LuaOnTV.

O LuaComp além de diminuir bastante o tempo de produção das aplicações, possibilita a criação de *templates*, o gerenciamento e manutenção do projeto da aplicação

através da geração de arquivos XML. O XML é uma arquivo de fácil entendimento para o usuário não muito experiente e que pode ser modificado fora do LuaComp.

6 CONCLUSÕES

Antes da conclusão, é indispensável registrar as dificuldades de acesso ao Ginga-NCL, pois até hoje, estamos utilizando o primeiro emulador do Ginga-NCL lançado em meados de Outubro de 2007. Os outros emuladores foram lançados com erros e após testes na SET este emulador foi considerado confiável, enquanto não for lançado um emulador completo do Ginga-NCL.

O objetivo desta dissertação foi apresentar a versão 1.0 ou protótipo de uma ferramenta de autoria chamada LuaComp, que implementa aplicações híbridas. Esta iniciativa surgiu da necessidade e da morosa disponibilidade do Ginga de forma completa. Importante lembrar que o pai do Ginga-NCL, prof. Doutor Luis Fernando Soares, vem enfatizando a capacidade da linguagem Lua e, consequentemente, encorajando pesquisadores a desenvolver novas soluções procedurais para aplicações de TV Digital interativa.O LuaComp gera arquivos NCLua baseados no *framework* LuaOnTV [44]. O LuaComp pode ser considerado a primeira ferramenta de autoria para aplicações NClua.

O projeto do LuaComp como resultado de um bom projeto possibilita o desenvolvimento de futuros códigos Java sem mudança na arquitetura do *software*. O desenvolvimento de componentes não gráficos ficam sujeito à evolução do *framework* LuaOnTV e a liberação do Ginga de forma definitiva. O desenvolvimento dos componentes não gráficos, para o canal de interatividade, foi protelado devido a necessidade de encapsular todas as transações com base na norma sobre segurança. Esta norma deverá ser homologada um pouco antes da liberação do Ginga, e que deve ser em meados de julho de 2009.

A metodologia empregada envolveu um amplo estudo da literatura sobre ferramentas de autoria, uma importante pesquisa para os tipos de aplicações e interatividade nas TVs digitais. Estudos também sobre padrões de usabilidade para aplicações de TV Digital interativa, e principalmente o destaque do *framework* LuaOnTV, desenvolvido pelo laboratório de TV Digital da UnB.

O estudo das ferramentas de autoria disponíveis no mercado serviu como base para identificação dos principais recursos que deveriam estar disponíveis para o autor independentemente do tipo de aplicação a ser gerada. A pesquisa sobre tipos de aplicação, interatividade e padrões de usabilidade serviram como guia para o desenho dos componentes gráficos, e fundamentalmente para justificar a não utilização de muitos dos recursos de

interatividades utilizados em computadores, e que ficam fora do contexto para aplicações de TV Digital interativa.

Além do LuaComp ser uma ferramenta WYSIWYG, facilitando bastante a produtividade do autor, traz, como forte recurso e novidade nas ferramentas de autoria para TV Digital interativa, a utilização do recurso do XML e, consequentemente, a oferta de criação e uso de templates.

O foco principal do LuaComp é a manipulação dos componentes gráficos para entrada e saída de dados.

6.1 TRABALHO FUTURO

Como objetivos para trabalhos futuros, a versão LuaComp 2.0, além de melhorar as funcionalidades e a parte visual de seus componentes, partirá para o desenvolvimento dos componentes não gráficos relacionados à segurança e edição ao vivo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] - GUIMARÃES, R. L. Composer: um ambiente de autoria de documentos NCL para TV digital interativa. Dissertação de Mestrado. Rio de Janeiro: PUC-RJ, 2007. 106 p.

[2] - PISKE, Otávio Rodolfo; SEIDEL, Fábio André. Rapid application development Especialização em software livre. São Paulo: Centro Universitário Positivo UnicenP, 2006. 8 p.

[3] – PRESSMAN, Roger S. Engenharia de software. 6. ed. São Paulo: McGraw Hill, 2005. 654 p.

[4] - FARIAS, Adalberto Cajueiro de F. Ferramentas CASE: suporte, adoção e integração. Monografia de Engenharia de Software, Pernambuco: UFPE, 1999. 33 p.

[5] - FLECHER, Tom; HUNT, Jim. Software engineering and CASE: bridging the culture gap. New York: McGraw-Hill, Inc. 1993. 277 p.

[6] - WAKER, Robert Ari. Aplicações hipermídia e a gestão da produção. São Paulo: UNIP, 1997. 8 p. Disponível em http://www.abepro.org.br/biblioteca/ ENEGEP2000_E0111.PDF

[7] - SALGADO, Ana C. Sistemas hipermídia: hipertexto e banco de dados. Porto Alegre: UFRGS, 1992. 181 p.

[8] - PRIMO, Alex Fernando Teixeira. Quão interativo é o hipertexto? In: PRIMO, Alex Fernando Teixeira et al. (Orgs.). Da interface potencial à escrita coletiva. Fronteira: Estudos Midiáticos, São Leopoldo, v. 5, n. 2, p. 125-142, 2003.

[9] - PUGH, John; DOHERTY, Robert. Evaluation of integrated election software development environment. PMI Software Ltda., 2001. 56 p.

[10] - CANTU, Marco. Dominando o delphi 6: a bíblia. São Paulo: Makron Books, 2003. 934 p.

[11] - WORLD WILDE WEB CONSORTING. XHTML™ 1.0 - The extensible hypertext markup language (Second Edition). W3C Recommendation. 2002. Disponível em: http://www.w3.org/TR/xhtml1/. Acesso em: 16 fev. 2009.

[12] - JAME. Jame author 2.0 - Tutorial. Disponível em: http://www.jame.TV. Acesso em: 13 nov. 2008.

[13] - CARDINAL SYSTEMS. Cardinal studio 4.0. Disponível em: http://www.cardinal.fi. Acesso em: 14 nov. 2008.

[14] - ENTRETENIMENTO E INTERATIVIDADE PARA TV DIGITAL. Cardinal studio professional 4.0. Disponível em: http://www.eiTV.com.br. Acesso em: 15 nov. 2008.

[15] - RODRIGUES, Rogério Ferreira; SOARES, Luiz Fernando Gomes. Produção de conteúdo declarativo para TV digital. Rio de Janeiro: PUC-RJ, 2006. 16 p. Disponível em: www.ncl.org.br/documentos/SEMISH2006.pdf. Acesso em: 2 mar. 2009.

[16] - SOARES, Luiz Fernando Gomes, RODRIGUES, Rogério Ferreira, Moreno , Marcio Ferreira. Ginga-NCL: Environment for the execution of declarative applications. São Paulo: ISDTV-T Forum. 2006. Disponível em: www.ncl.org.br/documentos/JBCS2007.pdf . Acesso em: 2 mar. 2009.

[17] – SOUZA FILHO, Guido Lemos de. CUNHA LEITE, Luiz Eduardo, FREIRE BATISTA, Carlos Eduardo Coelho. Ginga-J: Environment for the execution of procedural applications São Paulo: ISDTV-T Forum. 2006. Disponível em www.sbc.org.br/bibliotecadigital/download.php? paper=625. Acesso em: 2 mar. 2009.

[18] – WORLD WILDE WEB CONSORTING. SMIL 2.1 - SMIL is a language for describing audiovisual presentations on the web. Disponível em: http://www.w3.org/TR/SMIL2/. Acesso em: 15 nov. 2008.

[19] – ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 15606-2:2008 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, 2008. 297 p.

[20] – BULTERMAN, D. C. A.; et al. GRiNS: a GRaphical INterface for creating and playing SMIL documents. In: WWW7 Conference, Computer Networks and ISDN Systems, v. 30, i. 1-7, p. 519-529, Brisbane, Australia. 1998. Disponível em: http://epubs.cclrc.ac.uk/bitstream/428/GRiNS_final.pdf. Acesso em: 15 nov. 2008.

[21] - RODRIGUES, Rogério Ferreira. Ambiente declarativo para sistemas que implementem o GEM. Rio de Janeiro: PUC-RJ, 2007. 12 p.

[22] – ALTICAST. AltiComposer 2.0 - USA. Disponível em: http://www.alticast.com/main.html. Acesso em: 15 nov. 2008

[23] – ADOBE SYSTEMS. Macromedia Flash MX Professional 7. Disponível em: http://www.adobe.com/products/player_census/flashplayer. Acesso em: 10 nov. 2008.

[24] – ECMA. Standard ECMA-376 Office Open XML File Formats. Dec./2006. Disponível em: http://www.ecma-international.org/publications/standards/Ecma-376.htm. Acesso em: 14 nov. 2008.

[25] - COELHO, Rogério Miguel; SOARES, Luiz Fernando Gomes. Integração de ferramentas gráficas e declarativas na autoria de arquiteturas modeladas através de grafos compostos. Dissertação de mestrado. Rio de Janeiro: PUC-Rio, 2004. 9 p.

[26] - SOARES, Luiz Fernando Gomes; RODRIGUES, Rogério Ferreira. Nested Context Language 3.0 - Part 8 – NCL Digital TV Profiles. 2006. 145 p. Disponível em http://www.ncl.org.br/documentos/NCL3.0-DTV.pdf. Acesso em 24 abr. 2007.

[27] – CASTELARI, Sueli Pissarra; FERREIRA, Hilcéa Santos. Sistema hipermídia adaptativo no ensino de sensoriamento remoto à distância. In: XIII SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO. Anais. Florianópolis: INPE, 21-26 abr./2007. p. 1.417-1.421.

[28] - SOARES, Luiz Fernando Gomes; RODRIGUES, Rogério Ferreira; MUCHALUAT-SAADE, Débora Christina. Modelo de contextos aninhados – versão 3.0. Relatório Técnico, Laboratório TeleMídia, Departamento de Informática. Rio de Janeiro: PUC-Rio, 2003.

[29] – RODRIGUES, Leandro Marques; RODRIGUES, Rogério Ferreira; MUCHALUAT SAADE, Débora Christina; SOARES, Luiz Fernando Gomes. Acrescendo facilidades NCM a documentos SMIL. In: V Simpósio Brasileiro de Sistemas Multimídia e Hipermídia - SBMídia99. Goiânia: SBMídia, 14-16 jun./1999. p. 193-212.

[30] – NETBEANS. Disponível em : http://www.netbeans.org. Acesso em: 16 fev. 2009.

[31] – NCL ECLIPSE. Disponível em: http://laws.deinf.ufma.br/~ncleclipse/. Acesso em: 16 fev. 2009.

[32] – LUAECLIPSE. Disponível em: http://luaeclipse.luaforge.net/. Acesso em: 16 fev. 2009.

[33] – XLETVIEW. Disponível em: http://xleTView.sourceforge.net. Acesso em: 16 fev. 2009.

[34] – GINGANCL PLAYER. Disponível em: www.ginga.org.br. Acesso em: 16 fev. 2009.

[35] – SCHWALB, Edward M. iTV handbook technologies and Standards. 2004. Disponível em: http://portal.acm.org/citation.cfm?id=1008213.1008241. Acesso em: 15 ago. 2008.

[36] – MORRIS, Steven; SMITH-CHAIGNEAU, Anthony. Interactive TV standards. 2005. 611 p. Disponível em: http://books.google.com.br/books?id=Tv9_WcYPXtwC&dq=InteracTVe+TV+standards&printsec=frontcover&source=bn&hl=pt-BR&ei=doqZSYDjC-CKmQfYyaGVCg&sa=X&oi=book_result&resnum=4&ct=result#PPP23,M1. Acesso em: 16 fev. 2009.

[37] - FERNANDES, Jorge; LEMOS, Guido; SILVEIRA, Gledson Elias. Introdução à televisão digital interativa: arquitetura, protocolos, padrões e práticas. In: XXIV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. XXIII Jornada de Atualização em Informática. Salvador, JAI-SBC, 2004. Disponível em: http://www.cic.unb.br/~jhcf/MyBooks/itvdi/slides-jai2004/IntroducaoATelevisaoDigitalInterativa_dia2.pdf. Acesso em: 16 fev. 2009.

[38] - MORENO, Marcio Ferreira. Um *middleware* declarativo para sistemas de TV digital interativa. Rio de Janeiro: PUC-Rio, 2006. 105 p. [40] – CPQD – Arquitetura de referencia – Sistema Brasileiro de Televisão Digital – fev/2006

[39] - BRASIL. *Ginga*. 2008. Disponível em: http://svn.softwarepublico.gov.br/trac/ginga/. Acesso em: 16 fev. 2009.

[40] - BRASIL. *Decreto nº 4.901, de 26 de novembro de 2003*. Institui o sistema brasileiro de televisão digital – SBTVD, e dá outras providências. Disponível em: http://www.planalto.gov.br/ccivil_03/Decreto/2003/D4901.htm. Acesso em: 16 fev. 2009.

[41] – PICCOLO, Lara Schibelsky Godoy, Arquitetura do Set-top Box para TV Digital Interativa. Unicamp, 2002. Disponível em *www.ic.unicamp.br/~rodolfo/Cursos/mo401/2s2005/Trabalho/039632-settopbox.pdf*

[42] – CALDER, Bart; COURTNEY, Jon; FOOTE, Bill; KYRNITSZKE, Linda; RIVAS, Davis; SAITO, Chihiro, VANLOO, James; YE, Tao. JavaTV API Technical Overview. Version 1.0, nov. 14, 2000. Disponível em: http://java.sun.com/javame/technology/javatv/docs/jtv-1_0-spec_overview.pdf. Acesso em: 2 mar. 2009.

[43] – WIKIPEDIA. ATSC standard. Disponível em: http://en.wikipedia.org/wiki/ATSC_Standards. Acesso em: 16 fev. 2009.

[44] – LuaOnTV – disponível em http://luaforge.net/projects/luaontv/

[45] – CONVERGÊNCIA DIGITAL (2008). Middleware ginga para TV digital. Disponível em: http://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?from_info_index=11&infoid=15594&sid=54. Acesso em: 10 dez. 2008.

[46] - SOARES, Luiz Fernando Gomes; SANT'ANNA, Francisco; CERQUEIRA, Renato. NCLua: objetivos imperativos lua na linguagem declarativa NCL. Rio de Janeiro: PUC-Rio, 2008. 8 p.

[47] – TECGRAF. Noções de Lua 3.1. 2008. Disponível em: http://www.tecgraf.puc-rio.br/lua/ftp/nocoes-3.1.pdf. Acesso em 2 mar. 2009.

[48] – LEMOS, Guido de Souza Filho; LEITE, Luiz Eduardo Cunha; BATISTA, Freire, Carlos Eduardo Coelho. Ginga-J: the procedural *middleware* for the brazilian digital TV system. In: JOURNAL OF THE BRAZILIAN COMPUTER SOCIETY. Porto Alegre, n. 4, v. 13, 2007. p. 47-56.

[49] – THE COMPUTER LANGUAGE BENCHMARKS GAME. Debian 2008: Veja e teste. Disponível em: http://shootout.alioth.debian.org/. Acesso em: 10 dez. 2008.

[50] – DE LUCA, Cristina. TV digital: foco no Java preocupa pai do Ginga. Convergência Digital, 15 dez. 2008. Disponível em:

http://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?infoid=17187&query=simple&search%5Fby%5Fauthorname=all&search%5Fby%5Ffield=tax&search%5Fby%5Fkeywords=any&search%5Fby%5Fpriority=all&search%5Fby%5Fsection=&search%5Fby%5Fstate=all&search%5Ftext%5Foptions=all&sid=54&text=luiz+fernando+lua. Acesso em: 16 fev. 2009.

[51] - BECKER, Valdecir; FORNARI, Augusto; HERWEG FILHO, Günter H.; MONTEZ, Carlos. Recomendações de usabilidade para TV digital interativa. In: II WORKSHOP DE TV DIGITAL. Título. Curitiba: Biblioteca Digital Brasileira da Computação, 2006. 12 p. 55 e 56

[52] - SILVA, Ricardo Pereira e; PRICE, Roberto Tom. O uso de técnicas de modelagem no projeto de frameworks orientados a objetos. In: 26TH INTERNATIONAL CONFERENCE OF THE ARGENTINE COMPUTER SCIENCE AND OPERATIONAL RESEARCH SOCIETY (26th JAIIO) / First Argentine Symposium on Object Orientation (ASOO'97). Buenos Aires, 1997. p. 87-94.

[53] - SILVA, Ricardo Pereira e. Suporte ao desenvolvimento e uso de frameworks e componentes. Porto Alegre: UFRGS, 2000. 262 p. Disponível em: http://www.inf.ufsc.br/~ricardo/download/tese.pdf. Acesso em: 8 dez. 2008.

[54] – LUA CANVAS. Módulo NCLua. Disponível em: http://www.telemidia.puc-rio.br/~francisco/ nclua/reference/canvas.html. Acesso em: 10 dez. 2008.

[55] – GÜELL, Natacha; SCHWABE, Daniel; BARBOSA, Simone Diniz Junqueira. Método de avaliação de usabilidade na web baseado em modelo e padrões de comportamento – Rio de Janeiro: PUC-RJ, 18 jul. 2001. 20 p.

[56] - CROCOMO, Fernando Antônio. TV digital e produção interativa: a comunidade recebe e manda notícias. Tese de doutorado. Santa Catarina: UFSC, 2004. 189 p.

[57] - BBCi & INTERACTIVE PROGRAMMES. Interactive television design: style guide. 2005. Disponível em: http://www.bbc.co.uk/guidelines/newmedia/desed/itv/iTV-Design_v1.pdf. Acesso em: 16 fev. 2009.

[58] – BROADBANDBANANAS. Disponível em: http://www.broadbandbananas.com/. Acesso em: 16 fev. 2009.

[59] - COMPUTERWORLD. Hélio Costa diz que falta engajamento da indústria para popularizar a TV Digital. 17 dez. 2008. Disponível em: http://computerworld.uol.com.br/negocios/2008/12/17/helio-costa-diz-que-falta-engajamento-da-industria-para-popularizar-tv-digital/. Acesso em: 23 dez. 2008.

[60] - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 15604:2007 Televisão digital terrestre – receptores. 2008. 78 p. Disponível em: http://www.abnt.org.br/imagens/Normalizacao_TV_Digital/ABNTNBR15604_2007Vc_2008.pdf. Acesso em: 16 fev. 2009.

[61] LERUSALIMSCHY, Roberto. A Evolução de lua. 2007. Disponível em: http://www.inf.puc-rio.br/~roberto/talks/luapyconf.pdf . Acesso em: 16 fev. 2009.

[62] AMATO, João Amato Neto. Análise da emergência da TV digital e seus impactos na cadeia produtiva eletroeletrônica brasileira. In: JOURNAL OF TECHNOLOGY MANAGEMENT & INNOVATION, v. 1, 2006.

[63] INATEL. Sistema brasileiro de TV digital opera com sucesso. 15 mai. 2006. Disponível em: http://www.inatel.br/imprensa/release/IntegracaoTVDigital.asp. Acesso em: 16 fev. 2009.

[64] LOPES, Denise Maria Moura da Silva. Sistema brasileiro de TV digital: caminhos percorridos e implantação. 2007. Disponível em: http://www.redealcar.jornalismo.ufsc.br/resumos/R0097-1.pdf. Acesso em: 20 fev. 2009.

[65] – WIKIPEDIA. MHP. Disponível em: http://en.wikipedia.org/wiki/ATSC. Acesso em: 16 fev. 2009.

[66] – SUN MICROSYSTEMS. JavaTV API. 2007. Disponível em: http://java.sun.com/products/javatv/index.jsp. Acesso em: 2 mar. 2009.

[67] COÊLHO, Andrino Soares de Souza. Uma especificação de desenvolvimento de serviços para televisão digital interativa. 2005. Disponível em: http://www.redealcar.jornalismo.ufsc.br/resumos/R0097-1.pdf. Acesso em: 20 dez. 2008.

[68] CARVALHO, Sandra Régia Chaves, ARAÚJO, Vitcor Teixeira. Emuladores para TV Digital - OpenMHP e Xletview. 2007. Disponível em: http://www.tvdi.inf.br/upload/artigos/artigo7.pdf. Acesso em: 2 mar. 2009.

[69] GUIMARÃES, Rodrigo Laiola, COSTA, Romualdo M. de Resende. Interatividade e sincronismo em TV digital. 2006. Disponível em: http://www.redealcar.jornalismo.ufsc.br/resumos/R0097-1.pdf. Acesso em: 10 dez. 2008.

[70] – GINGABRASIL-NCL3.0. Disponível em: http://en.wikipedia.org/wiki/ATSC_Standards. Acesso em: 16 fev. 2009.

[71] TAVARES, Tatiana Aires; SANTOS, Celso Alberto Saibel; ASSIS, Thiago Rocha de; PINHO, Clarissa Braga Bittencourt de; CARVALHO, Germano Mariniello de; COSTA, Clarissa Santana da. Ferramenta de apoio à educação infantil. 2005. Disponível em: http://bibliotecadigital.sbc.org.br/download.php?paper=939 Acesso em: 10 dez. 2008.

[72] – MENDES, Luciano Leonel. SBTVD: Uma visão sobre a TV digital no Brasil. In: T&C Amazônia, a. V, n. 12, out. /2007, p. 48-59.

[73] – TOME, Takashi. Tecnologias Digitais em Radiodifusão Aspectos Tecnológicos e Potencialidades do Uso. Disponível em: http://74.125.47.132/search?q=cache:40e1uJ2BSkYJ:www.direitoacomunicacao.org.br/novo/i

ndex.php%3Foption%3Dcom_docman%26task%3Ddoc_download%26gid%3D9+largura+de+banda+tv+digital+Mbit/s&hl=pt-BR&ct=clnk&cd=4&gl=br. Acesso em: 2 mar. 2009.

[74] LUAFORGE. Disponível em: http://luaforge.net/projects/luaontv/. Acesso em: 2 mar. 2009.