UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

ALOYSIO NANDI TISCOSKI

ESTUDO, MODELAGEM E ADAPTAÇÃO DE UM PLAYER GINGA-NCL PARA A CONSTRUÇÃO DE CONTEÚDO EM T-LEARNING

FLORIANÓPOLIS - SC
2010

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

ALOYSIO NANDI TISCOSKI

ESTUDO, MODELAGEM E ADAPTAÇÃO DE UM PLAYER GINGA-NCL PARA A CONSTRUÇÃO DE CONTEÚDO EM T-LEARNING

ORIENTADOR: PROF. DR. JOSÉ LEOMAR TODESCO
COORIENTADOR: PROF. DR. FERNANDO OSTUNI GAUTHIER

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de informação

FLORIANÓPOLIS - SC
2010

ALOYSIO NANDI TISCOSKI

ESTUDO, MODELAGEM E ADAPTAÇÃO DE UM PLAYER GINGA-NCL PARA A CONSTRUÇÃO DE CONTEÚDO EM T-LEARNING

Este trabalho de graduação foi julgado adequado para obtenção do grau de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo curso de Sistemas de Informação da Universidade Federal de Santa Catarina.

_____

Prof. Dr. José Leomar Todesco
Orientador

_____

Prof. Dr. Fernando Ostuni Gauthier
Coorientador

_____

Airton Zancanaro
Banca Examinadora

“I am against religion because it teaches us to be satisfied with not understanding the world. ”

Richard Dawkins

DEDICATÓRIA

Dedico este trabalho aos meus pais que sempre colocaram o meus estudos em primeiro lugar.

AGRADECIMENTOS

Agradeço em especial aos meus amigos Paul Eipper e Fernando Bunn, além de todo que direta ou indiretamente me ajudaram no desenvolvimento deste trabalho.

RESUMO

O sistema de difusão de sinais de televisão no Brasil está em pleno processo de transição, onde as tecnologias analógica e digital estão sendo transmitidas paralelamente. Um dos maiores objetivos da implantação da TV Digital no país é a inclusão social para milhões de brasileiros por meio do acesso à tecnologia digital, uma vez que a televisão está presente em mais de 90% lares brasileiros. Outro importante objetivo é a universalização da educação à distância pela disponibilização de aplicativos de T-Learning, que propiciarão educação a uma grande parte da população. A proposta deste trabalho é a construção um player NCL para simulação de um set-top box, que possibilite aos criadores de conteúdo para educação a distância testar o material gerado de uma maneira fácil, em ambiente de produção. Para atingir este objetivo foi criada uma ferramenta que através do protocolo SSH se comunica com o Ginga-NCL instalado em uma máquina virtual, onde é possível testar o conteúdo gerado.

Palavras Chave: TV Digital Interativa, Interatividade, Player NCL.

ABSTRACT

The system of broadcasting television signals in Brazil is in the process of transition, where the analog and digital technologies are being transmitted in parallel. A major goal of the deployment of Digital TV in this country is the social inclusion for millions of Brazilians by access to digital technology once the television is present in more than 90% of Brazilian households. Another important goal is the universalization of distance education through the provision of T-learning applications, which will provide education to a large population. The aim of this work is to build an NCL player to simulate a set-top box, which allows content creators to test distance education material generated in a easy way, in the production environment. To achieve this goal was created a tool who communicates with the Ginga-NCL installed on a virtual machine through the SSH protocol, where you can test the generated content.

Keywords: Interactive Digital TV, Interactivity, NCL Player.

LISTA DE SÍMBOLOS

STB - Set-top box;

TVD - TV Digital;

TVDI - TV Digital interativa;

NCL - Nested Context Language;

XML - Extensible Markup Language;

SBTVD - Sistema Brasileiro de Televisão Digital;

NCM - Nested Context Model;

EAD - Educação a distância;

SSH - Secure Shell;

VM - Máquina virtual (Virtual Machine);

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Segundo dados da PNAD (Pesquisa Nacional por Amostra de Domicílios) do IBGE (Instituto Brasileiro de Geografia e Estatística), realizada no ano de 2008, 95,1% dos lares brasileiros possuem pelo menos um aparelho de televisão, sendo assim o meio de transmissão de informação mais abrangente em nosso país, e para muitos brasileiros, é o único meio de adquirirem conhecimento, informação e cultura.

Com o advento da TV digital interativa (TVDI) muitos serviços que atualmente não são possíveis de serem implementados, devido as limitações tecnológicas da TV analógica tornam se viáveis. As possibilidades vão desde a criação de novos modelos de negócios seguindo para a abertura de novos horizontes na prestação de diversos tipos de serviços em diferentes seguimentos da sociedade, inclusive no âmbito educacional.

Uma importante característica advinda da digitalização, que tem incentivado muitos dos países no desenvolvimento de alternativas para implantação da mesma, é a possibilidade da “convergência” entre vários meios de comunicação, e mais a interatividade, a partir de um único equipamento (SANTOS, 2006).

O principal objetivo da implantação da TV Digital (TVD) no Brasil vai além de oferecer à população a possibilidade de receber em suas casas imagens e som de alta qualidade. O Decreto Presidencial No 4.901, de 26 de novembro de 2003, que instituiu o Sistema Brasileiro de Televisão Digital (SBTVD-T) destaca: “I - promover a inclusão social, a diversidade cultural do País e a língua pátria por meio do acesso à tecnologia digital, visando à democratização da informação; e II - propiciar a criação de rede universal de educação à distância”.

Um dos grandes desafios a ser superado com a chegada da TVD é a produção de conteúdo interativos de qualidade, bem como, a formação de mão-de-obra especializada para os diversos fins que exige essa nova tecnologia, isso em curto espaço de tempo, pois o sinal já está sendo transmitido em 26 cidades do país.

Por este motivo o departamento de Engenharia e Gestão do Conhecimento (EGC) da Universidade Federal de Santa Catarina (UFSC) está desenvolvendo uma

ferramenta para criação de conteúdo interativo para TV Digital, visando facilitar o trabalho das pessoas que criam aula para T-learning.

Este trabalho é um complemento ao trabalho e título: “Ferramenta de geração de conteúdo interativo para a TV Digital baseado em componentes”, que está sendo desenvolvido pelos alunos Natan Zeferino e Carlos Eduardo Machado.

1.1. OBJETIVO GERAL

Este trabalho tem como objetivo construir um player NCL para simulação de um set-top box, para que seja possível testar conteúdo gerado para educação a distância ainda em ambiente de produção.

1.2. OBJETIVOS ESPECÍFICOS

- Apresentar o estado da arte do middleware Ginga.
- Identificar o funcionamento da TV Digital;
- Implementar o player NCL;
- Testar e validar o player.

1.3. JUSTIFICATIVA

Utilizando a ferramenta proposta o criador de conteúdo para T-learning poderá testar as aulas geradas, em um ambiente familiar, pois ele será integrado ao sistema gerado no trabalho de título “Ferramenta de geração de conteúdo interativo para a TV Digital baseado em componentes”, sem a necessidade da compra de um set-top box físico e minimizará o trabalho, pois o player pegará automaticamente os arquivos a serem testados, não tendo a necessidade do usuário ter que copiar os arquivos para uma máquina virtual ou um STB a cada teste.

1.4. METODOLOGIA

Para a realização do presente trabalho, estipulam-se as seguintes etapas a serem realizadas:

1. Levantamento na literatura sobre as informações e tecnologias disponíveis relacionadas ao tema proposto;

2. Definição de tecnologias e ferramentas serão utilizadas na construção do sistema proposto;

3. Definição da arquitetura do sistema proposto;

4. Construção de um protótipo funcional;

5. Testes do protótipo.

1.5. ORGANIZAÇÃO DO TRABALHO

O presente capítulo faz uma contextualização do assunto relacionado a este trabalho e apresenta a problemática que serviu de subsídio à pesquisa realizada, bem como os objetivos estabelecidos, a delimitação do escopo, as justificativas e a metodologia empregada.

No capítulo dois, é apresentada uma fundamentação teórica sobre os principais conceitos envolvidos neste estudo, dentre eles: TV Digital, Ginga e Educação a distância.

No terceiro capítulo é feita a descrição das tecnologias e ferramentas utilizadas no desenvolvimento do sistema, são descritos alguns problemas encontrados na pesquisa e como foram solucionados, também é apresentada a modelagem do sistema e uma breve descrição da implementação.

Já no quarto e último capítulo, são apresentadas algumas considerações finais e alguns possíveis trabalhos que podem ser desenvolvidos a partir da proposta aqui apresentada.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados alguns conceitos que embasam o desenvolvimento deste trabalho, começando com uma contextualização de TV Digital, mostrando seus principais componentes, a situação atual da TV Digital no Brasil, é apresentado o middleware Ginga, uma conceitualização de educação a distancia e T-Learning, também são mostradas algumas tecnologias e ferramentas utilizadas no desenvolvimento do sistema proposto, como máquinas virtuais, Java e o Virtual Ginga-NCL.

2.1.TV DIGITAL

Os constantes avanços tecnológicos focaram a melhora da qualidade da imagem dos televisores, o tamanho das telas e dos novos layouts. Deste modo apareceram no mercado os aparelhos de TV de alta definição, como que preparando um ambiente básico para aportar uma nova forma de conteúdos televisivos que estaria por vir: os conteúdos digitais.

De acordo com Oliveira (2005, p.8), muitas são as novidades na TV digital com relação a analógica, quais sejam:

- Qualidade do som comparado ao Compact Disk (CD);
- Qualidade da imagem no padrão do Digital Vídeo Disk (DVD) player com 525 linhas;
- Aumento nas transmissões de áudio relacionadas aos programas, como a dublagem e a legenda em diversas línguas;
- Dados transmitidos junto aos programas;
- De acordo com as características geográficas locais, ampliação da área coberta o que possibilita flexibilidade nos parâmetros de transmissão;
- Novos serviços: comércio eletrônico, acesso à Internet, educação à distância, guia eletrônico de programação, entre outros;
- Ampliação do número de canais;

• Programação variada;

• Single Frequency Network (SFN), uma tecnologia capaz de definir, por exemplo, um único número de canal para todo o Brasil;

• Possibilidade de gravação digital com alto desempenho para conteúdos multimídias preferidos pelo usuário: filmes, programas, imagens e fotos;

• Número maior de aplicações e serviços interativos englobando áudio, vídeo, texto

• e os mais diversos tipos de elementos gráficos; e

• Possibilidade de seleção, por parte do telespectador, do conteúdo que deseja ver, como, quando e onde.

Atualmente 10 países já fizeram a transição completa da transmissão analógica para Digital, e muitos países tem planos para fazer esta mudança nos próximos anos, outros, como o Brasil, estão no meio deste processo.

2.2.COMPONENTES DA TV DIGITAL

Segundo Santos (2007) um sistema de TV Digital pode ser composto por uma emissora (estúdio) de TV digital que é onde o conteúdo é gerado e transmitido, um set-top box que recebe e converte os dados transmitidos pela emissora e por um aparelho analógico de TV, onde os dados recebidos pelo set-top box serão exibidos. (Figura 1)

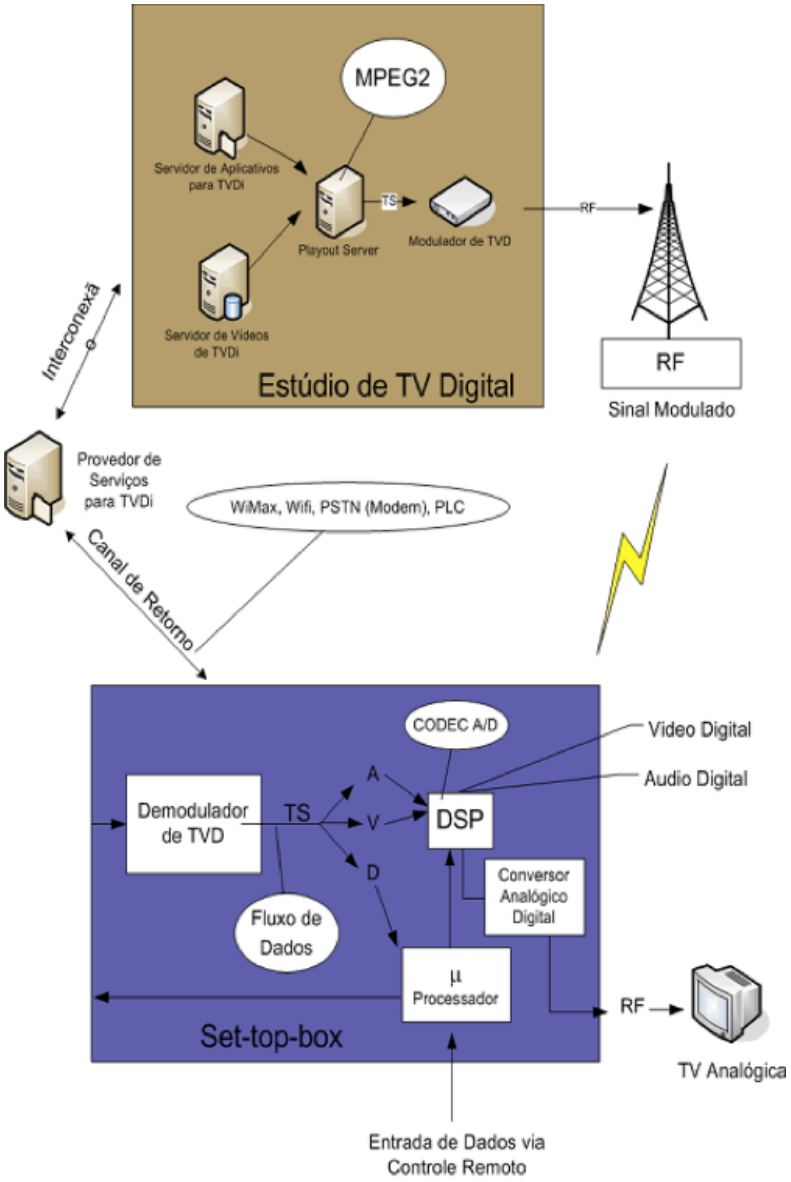

Figura 1: Visão Geral de um Sistema de TV Digital Fonte: Santos 2007

2.2.1.Emissoras

É quem transmite o conteúdo ao telespectador. elas devem possuir 2 servidores, sendo um de vídeos de TVDi e um outro para armazenar os sistemas aplicativos de TVDi que são associados aos vídeos. Os vídeos poderão ser disponibilizados em definição padrão (*Standard Definition* - SD) ou em alta definição (*High Definition* - HD), os vídeos são enviados para um servidor central antes de serem modulados.

É função do servidor de aplicativos gerar um *Transport Stream* (TS) que conterá vídeo, áudio e dados associados, enviá-los para um modulador que transforma estes dados em um sinal radiofrequência (*Radio Frequency* - RF). Este sinal será enviado a uma antena através de *broadcast* para as residências dos telespectadores.

O STB demodula os sinais de RF originados nas emissoras de TVD e extrai o fluxo de dados (áudio, vídeo e dados), ou TS. Os aplicativos são executados a partir de um micro processador existente nos STB e o telespectador poderá interagir com esses aplicativos através de um controle remoto ou outro dispositivo de entrada, como por exemplo, um teclado sem fio (Santos 2007).

2.2.2.Middleware

O middleware é a camada de software responsável pela interface entre os aplicativos digitais interativos e o sistema operacional do STB. Diversos padrões de middleware foram desenvolvidos, cada qual referente a um tipo específico de TV digital, que privilegia certos aspectos em relação a outros. Normalmente, nos ambientes de desenvolvimento para TV Digital usa-se dois paradigmas de programação: o paradigma procedural e o declarativo. O modelo procedural baseia-se na utilização de uma "máquina virtual", ou Virtual Machine (VM), na qual os aplicativos são executados, isso requer um maior poder de processamento por parte do STB, um exemplo disso é o uso da linguagem de programação Java no Ginga-J. Já o modelo declarativo, é caracterizado pelo uso de linguagens declarativas, como por exemplo a eXtensible Hypertext Markup Language (XHTML), Nesse modelo, quem defini o estilo de apresentação e os conteúdos é a linguagem de marcação com o objetivo de suprir as necessidades específicas do broadcast de dados.

Um middleware para aplicações de TV digital basicamente consiste de máquinas de execução das linguagens oferecidas e bibliotecas de funções, que permitem o desenvolvimento rápido e fácil de aplicações interativas para TV digital. Essas aplicações vão possibilitar, por exemplo, acesso à internet, operações bancárias, envio de mensagens para o canal de TV ao qual se está assistindo, entre outros (Santos 2007).

2.2.3.Canal de Retorno

É um canal de transmissão através do qual o telespectador pode enviar e receber informações personalizadas pelo Provedor de Serviços de TVDi. O provedor está ligado tanto ao estúdio de TV Digital quanto ao telespectador, podendo receber as solicitações originadas pelo telespectador e responder às requisições efetuadas.

O canal de retorno é um componente muito importante em um sistema de TVD, pois possibilita a execução de uma grande variedade de funcionalidades e serviços, além de permitir o recebimento e envio de informações de forma personalizada.

2.2.4.Set-top Box

O receptor digital é um dispositivo capaz de converter os sinais digitais enviados pela emissora para que possam ser assistidos nos aparelhos convencionais de TV. Esse dispositivo pode vir embutido no aparelho televisor ou ser adquirido à parte, dessa forma o receptor é chamado de STB. Este nome se deve ao detalhe de que o dispositivo ter um formato de uma caixa e, geralmente, ficar sob a TV (MONTEZ; BECKER, 2005).

2.3.TV DIGITAL NO BRASIL

O Sistema Brasileiro de TV Digital Terrestre foi baseado no sistema japonês, o Integrated Services Digital Broadcasting Terrestrial (ISDB-T) e é tecnicamente conhecido como Sistema Brasileiro de TV Digital Terrestre (SBTVD-T), ele proporciona uma série de diferenciais quando comparado aos outros sistemas atualmente utilizados no mundo. Esses diferenciais estão na junção da base técnica de transmissão do sistema japonês com os padrões de compressão digital de áudio e vídeo introduzidos pelo Brasil, que são mais modernos e eficientes do que os adotados por outros padrões, um outro grande diferencial é a utilização do middleware Ginga.

Na versão brasileira foram acrescentadas tecnologias desenvolvidas pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB).

Além disso, a especificidade do sistema brasileiro possibilita a transmissão de conteúdo de alta qualidade, tanto em termos de imagem como de som, permitindo ao mesmo tempo a recepção móvel e portátil dos sinais de TV digital. Para oferecer esses diferenciais, o SBTVD-T adotou o padrão MPEG-4, também conhecido como H.264, para codificação de vídeo, e o HE-AAC v2 para o áudio (DTV 2009).

Outros importantes diferenciais do SBTVD-T são a mobilidade e a interatividade. No caso da mobilidade é possível percebê-la na prática, uma vez que já estão à disposição do consumidor brasileiro diversos dispositivos móveis por meio dos quais se pode assistir à TV digital, como celulares, mini-televisores e receptores USB.

Em relação à interatividade, os documentos sobre o middleware Ginga da TV digital brasileira estão em processo de Consulta Nacional da ABNT. Para garantir que o Ginga esteja livre do pagamento de royalties.

A TV digital começou a ser transmitida no Brasil em 02 de dezembro de 2009 e tem previsão de ser concluída no dia 29 de junho de 2016, quando substituirá completamente a transmissão analógica, neste período a digitalização está sendo feita em etapas, começando pelas grandes capitais (Figura 2).

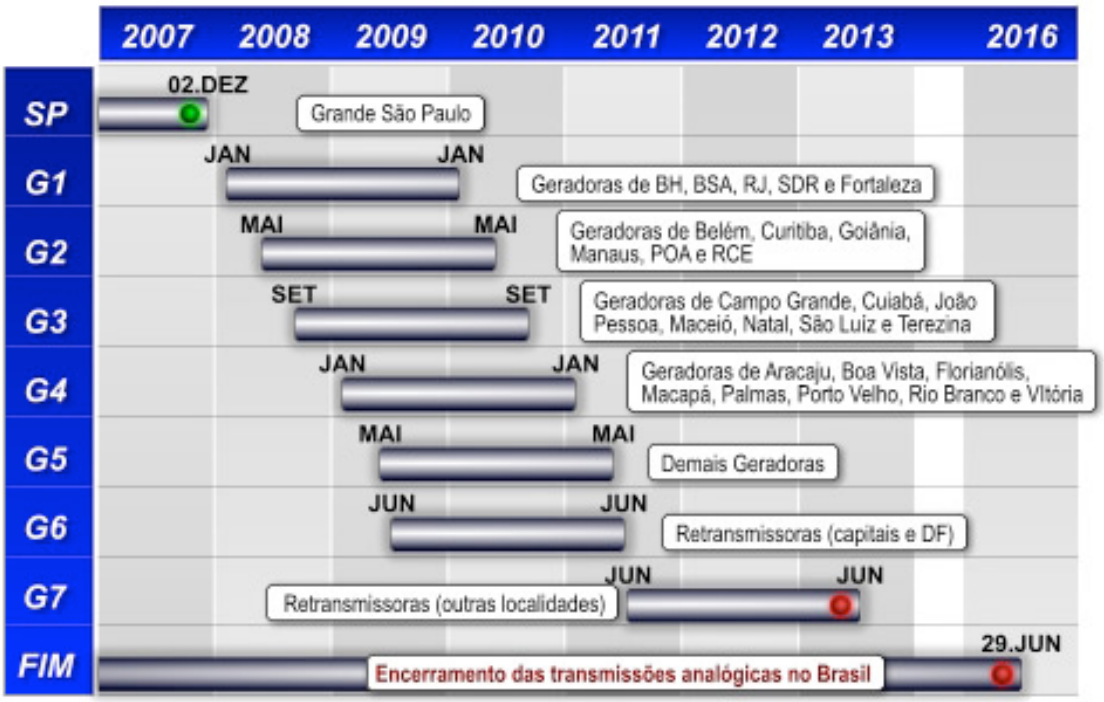

Figura 2: cronograma de implantação da TV Digital no Brasil. Fonte: DTV 2009

2.4.GINGA

Ginga é o middleware de especificação aberta adotado pelo Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) ele será instalado em conversores (set-top boxes) e em televisores. É uma camada de software intermediária, entre o sistema operacional e as aplicações. Sua principais funções são: tornar as aplicações independentes do sistema operacional da plataforma de hardware utilizados e oferecer um melhor suporte ao desenvolvimento de aplicações. Ou seja, é o Ginga que dá suporte a interatividade.

O Ginga é constituído por um conjunto de tecnologias padronizadas e inovações brasileiras que o tornam a especificação de middleware mais avançada do mundo atualmente e a melhor solução para os requisitos do país. O Ginga é o resultado de vários anos de pesquisas realizadas pela Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB).

O sistema é subdividido em dois subsistemas principais interligados (Ginga-NCL e Ginga-J), que permitem o desenvolvimento de aplicações seguindo dois paradigmas de programação diferentes. Dependendo das funcionalidades requeridas no projeto de cada aplicação, um paradigma será mais adequado do que o outro (DTV 2009).

2.4.1.Ginga-NCL

O Ginga-NCL foi desenvolvido pela PUC-Rio com o objetivo de prover uma infra-estrutura de apresentação para aplicações declarativas escritas na linguagem NCL (Nested Context Language), que é uma aplicação XML com facilidades para a especificação de aspectos de interatividade, sincronismo espaço-temporal entre objetos de mídia, adaptabilidade, suporte a múltiplos dispositivos e suporte à produção ao vivo de programas interativos não-lineares (Ginga-NCL 2009).

Apesar de ser um ambiente declarativo, o Ginga-NCL ainda permite executar aplicativos em uma linguagem procedural chamada LUA, também desenvolvida pela PUC-Rio e considerada pelo mercado como uma excelente solução para execução de aplicações multimídia, devido à sua ótima relação performance x leveza. De acordo com as normas da ABNT (ABNT, NBR 15605-5:2008) esse ambiente é obrigatório nas configurações Ginga em receptores de sinal SBTVD-T fixos, móveis e portáteis (celulares).

2.4.2.Ginga-J

O Ginga-J foi desenvolvido pela UFPB para prover uma infra-estrutura de execução de aplicações baseadas na linguagem Java, com facilidades especificamente voltadas para o ambiente de TV digital.

É uma máquina de execução de aplicativos desenvolvidos com a linguagem Java e foi desenvolvido pela Universidade Federal da Paraíba (UFPB). Um ambiente apto a executar código Java é extremamente importante não só pelo grande poder de desenvolvimento de aplicações complexas que o Java oferece como também pelo enorme contingente de analistas, técnicos e programadores especializados

nessa linguagem, no mundo e especialmente no Brasil. Esse ambiente é chamado de Ginga-J e é obrigatório em configurações Ginga em receptores de sinal SBTVD-T fixos e móveis, porém não é obrigatório em receptores portáteis.

O ambiente Ginga-J utiliza API's (Application Program Interfaces) do pacote Java-DTV, desenvolvido pela Sun Microsystems a pedido do Governo Brasileiro, de forma que todas as API's do conhecido padrão GEM estivessem inclusas no pacote, acrescido de outras API's específicas do Ginga. Isso permitiu um pacote de API's que atendesse aos requisitos do sistema SBTVD-T e sem a cobrança de royalties, resultando no desenvolvimento de set top boxes e aparelhos de TV bem mais acessíveis (DTV 2009).

2.5.NCL

A linguagem NCL (Nested Context Language) é uma linguagem declarativa para autoria de documentos hipermídia baseados no modelo conceitual NCM (Nested Context Model), que é utilizado para especificação de documentos hipermídia com sincronização temporal e espacial entre seus objetos de mídia. A linguagem foi desenvolvida utilizando uma estrutura modular, seguindo os princípios adotados pelo W3C. Sendo assim, seus módulos para especificação de conectores e templates de composição, chamados XConnector e XTemplate respectivamente, podem ser incorporados a outras linguagens existentes (NCL 2009).

A utilização de linguagens declarativas como NCL em aplicações 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 (SANT'ANNA 2008).

Por este motivo algumas tarefas como processamentos matemáticos, manipulações 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, necessitam do auxílio de uma linguagem imperativa.

2.6.EDUCAÇÃO A DISTANCIA

Existem vários conceitos para educação a distância (EaD), um deles é o da Associação Brasileira de Educação a Distância (ABED, 2006), que conceitua EAD como sendo “a modalidade de educação em que as atividades de ensino-aprendizagem são desenvolvidas majoritariamente (e em bom número de casos exclusivamente) sem que alunos e professores estejam presentes no mesmo lugar à mesma hora.”

Um outro exemplo é o Decreto-Lei Brasileiro nº 5.622, de 20 de dezembro de 2005, onde o termo é conceituado de maneira diferente:

> “ Art. 1o. – Para os fins deste Decreto, caracteriza-se a educação a distância como modalidade educacional na qual a mediação didático pedagógica no processos de ensino e aprendizagem ocorre com a utilização de meios de tecnologia de informação e comunicação, com estudantes e professores desenvolvendo atividades educativas em lugares ou tempos diversos.”

Existem, ainda, outros conceitos, como mostra o pesquisador Nunes (1994):

• Dohmem (1967, apud, NUNES, 1994) diz que a educação a distância é um método de auto-estudo onde o aluno adquire conhecimentos com base no material de estudo que ele tem acesso, sendo o seu desenvolvimento acompanhado e supervisionado através de professores;

• Peters (1973, apud, NUNES, 1994) afirma que educação à distância é uma maneira sistematizada de partilhar conhecimento, habilidades e atitudes através do uso extensivo de meios de comunicação;

• Holmberg (1977, apud, NUNES, 1994) sob o termo EAD ficam encobertos diversos tipos de estudos, em que seus vários níveis não se encontram supervisionados direta e imediatamente por tutores presenciais;

• Perry e Rumble (1987, apud, NUNES, 1994) apresentam a comunicação bidirecional como característica básica para a educação à distância, quando não se encontrarem juntos, no mesmo local, aluno e professor;

A pesquisadora Belloni (2002) considera a educação a distância, como parte de um processo mais amplo de inovação educacional, que se refere à integração das novas Tecnologias de Informação e Comunicação (TIC) nos processos educacionais.

2.6.1.T-learning

Segundo Bates (2003), T-Learning é o tipo de educação à distância baseado em televisão interativa. Esse termo é também apresentado como a convergência entre a TV digital interativa (TVI) e o E-Learning, sendo mais tarde conhecida como a utilização da tecnologia computacional para apoiar a formação e a realização de atividades educacionais.

Ainda em Bates (2003), o T-Learning dará possibilidade de os telespectadores terem acesso a diversos materiais didáticos, do tipo: filmes, imagens, hipertexto, etc., a partir de suas casas, escolas, local de trabalho ou nos centros comunitários.

2.7.MÁQUINA VIRTUAL

O conceito de Máquina virtual (VM) foi originalmente definido por Popek e Goldberg (1974) como “uma duplicata isolada e eficiente de uma máquina real”, atualmente também são inclusas as máquinas virtuais que não tem uma ligação direta com nenhum hardware real.

Uma característica marcante de uma máquina virtual é que o software que roda nela é limitado aos recursos e abstrações providos por pela mesma, ele não pode ultrapassar este “mundo virtual”. As máquinas virtuais são separadas em dois grandes grupos, baseado no seu uso e no grau de ligação com a máquina real. Uma máquina virtual de de sistema proporciona um plataforma completa que suporta a execução de um sistema operacional completo, já a máquina virtual de processo é desenhada para rodar apenas um programa, o que significa que ela suporta apenas um processo (ICOER 2008).

2.7.1.Máquina Virtual de Sistema

Máquinas virtuais de sistema, também conhecidas como máquina virtual de hardware, permitem a partilha dos recursos físicos de uma máquina real entre diferentes máquinas virtuais, cada uma rodando seu próprio sistema operacional, a camada de software que proporciona a virtualização é chamado de monitor de máquina virtual, ou hypervisor.

Principais Vantagens das Máquinas Virtuais de Sistema:

• Múltiplos sistemas operacionais podem coexistir no mesmo computador, cada um fortemente isolado do outro;

• A máquina virtual pode proporcionar um conjunto de instruções (ISA), que é um pouco diferente daquela da máquina real;

• provisionamento de aplicação, manutenção, alta disponibilidade e recuperação de desastres.

Principal desvantagem das Máquinas Virtuais de Sistema:

• A máquina virtual é menos eficiente que uma máquina real quando acessa o hardware indiretamente.

Múltiplas VMs cada um rodando seu próprio sistema operacional (sistema operacional convidado) são freqüentemente usados em consolidação de servidores, onde os diferentes serviços que geralmente rodavam em máquinas individuais, a fim de evitar interferências são executados em VMs separadas na mesma máquina física. Este uso é freqüentemente chamado de isolamento de qualidade de serviço (isolamento de QoS) (SMITH; NAIR, 2005).

A utilização de máquinas virtuais de sistema também é um atrativo veículo para distribuição de software, principalmente daqueles que possuem alta complexidade de compilação, instalação ou personalização junto ao hardware e serviços.

2.7.2.Máquina Virtual de Processo

As máquinas virtuais de processo, também chamada de máquina virtual de aplicação, roda uma aplicação normal dentro do sistema operacional e suporta um único processo. Ela é criada quando o processo é iniciado e destruída logo após o seu término. Sua finalidade é fornecer um ambiente de programação independente de plataforma que abstrai detalhes do hardware subjacente ou sistema operacional e permite que um programa seja executado da mesma forma em qualquer plataforma.

Este tipo de VM se tornou popular com a linguagem de programação Java, que é implementada utilizando a máquina virtual Java, outros exemplos são a máquina virtual Parrot, que serve como uma camada de abstração para várias linguagens interpretadas, e o .NET Framework, que é executado em uma máquina virtual chamada de Common Language Runtime (SMITH; NAIR 2005).

Existem vários softwares que permitem a criação de máquinas virtuais de sistema, dentre eles, podemos citar o VMware Workstation. as máquinas virtuais criadas por ele, podem ser rodadas atreves do VMware Player, este é grátis e existe versão para Linux e Windows.

2.8.VMWARE

VMware Workstation é uma suíte de software de máquina virtual para computadores x86 e x86-64. Este conjunto de software permite aos usuários configurar várias máquinas x86 e x86-64 e usar uma ou mais destas máquinas virtuais simultaneamente com o sistema operacional hospedeiro. Cada instância da máquina virtual pode executar seu próprio sistema operacional, tais como Windows, Linux, BSDs, e outros. Em termos simples, VMware Workstation permite que uma máquina física possa executar múltiplos sistemas operacionais simultaneamente.

Além de fazer uma ponte para os adaptadores de rede existentes, dispositivos de CD-ROM, discos rígidos e dispositivos USB, o VMware Workstation também é capaz de simular alguns hardwares. Por exemplo, pode montar um arquivo ISO

como um drive de CD-ROM, arquivos .vmdk como discos rígidos e também pode configurar o seu adaptador de rede para usar Network Address Translation (NAT), através da máquina física, ao invés de pontes (o que requerem um endereço IP para cada máquina virtual na rede) (ICOER 2008).

O VMware Workstation foi o software utilizado para a criação do Virtual Ginga-NCL, que é uma máquina virtual com o Ginga-NCL instalado, onde é possível testar programas feitos para TV digital sem a necessidade de um set-top box.

2.9.VIRTUAL GINGA-NCL

O Set-top Box Virtual Ginga-NCL é uma máquina virtual que utiliza o sistema operacional Linux, distribuição Fedora Core 7 otimizado para o desenvolvimento do Middleware Ginga e para a execução do Ginga-NCL versão C++ (Implementação de Referência). Dessa forma, foram retirados programas como todo ambiente gráfico X/ GNOME/KDE, e suas ferramentas, como pode ser visto na Figura 3. Todas estas mudanças foram feitas para criar um ambiente o mais próximo possível ao de um set-top box de desenvolvimento real.

O Virtual Ginga-NCL foi construído com o intuito de facilitar o processo de distribuição e implantação do Ginga-NCL versão C++, a versão do player NCL que conta com os mais avançados recursos de apresentação de aplicações declarativas, melhor desempenho e maior proximidade de uma implementação embarcada em set-top boxes reais.

Figura 3: Virtual Ginga-NCL rodando através do VMWare Fusion em uma máquina utilizando o sistema operacional Mac OS X 10.6. Fonte: Imagem capturada pelo Autor.

A máquina virtual fedora-fc7-ginga-i386 foi criada e configurada pela equipe do Laboratório TeleMídia da PUC-Rio utilizando o software VMWare Workstation 6 (trial) (VIRTUAL GINGA-NCL 2006).

A comunicação do usuário com o Virtual Ginga-NCL é feita através do protocolo SSH, que permite a troca de mensagens entre a máquina física e a máquina virtual fedora, onde está instalado o Virtual Ginga-NCL.

2.10.MVC

Para a modelagem das classes do sistema proposto, será utilizado o padrão de projeto Model-View-Controller (MVC), que é um padrão de alto nível bastante antigo, onde há a preocupação com a arquitetura global de um aplicativo e os objetos são classificados de acordo com as funções gerais que desempenham em um sistema. É também um padrão complexo na medida em que inclui vários padrões mais elementares.

Programas orientados a objeto se beneficiam de várias formas, adaptando o MVC em seu design, pois os objetos nestes programas tendem a ser mais reutilizáveis e suas interfaces tendem a ser melhor definidas. Os programas em geral são mais adaptáveis às mudanças nos requisitos, em outras palavras, eles são mais facilmente extensíveis do que programas que não são baseados em MVC (MVC 2010).

2.10.1.Papeis e Relacionamentos de Objetos MVC

O padrão de projeto MVC considera que existem três tipos de objetos: objetos do modelo, objetos de visão (view) e objetos de controle. O padrão define os papéis que cada tipo de objeto desempenha na aplicação e suas linhas de comunicação. Ao projetar uma aplicação, um passo importante é escolher ou criar classes personalizadas para objetos de que se enquadram em um desses três grupos. Cada um dos três tipos de objetos é separado dos outros por fronteiras abstratas e se comunica com objetos de outros tipos através desses limites.

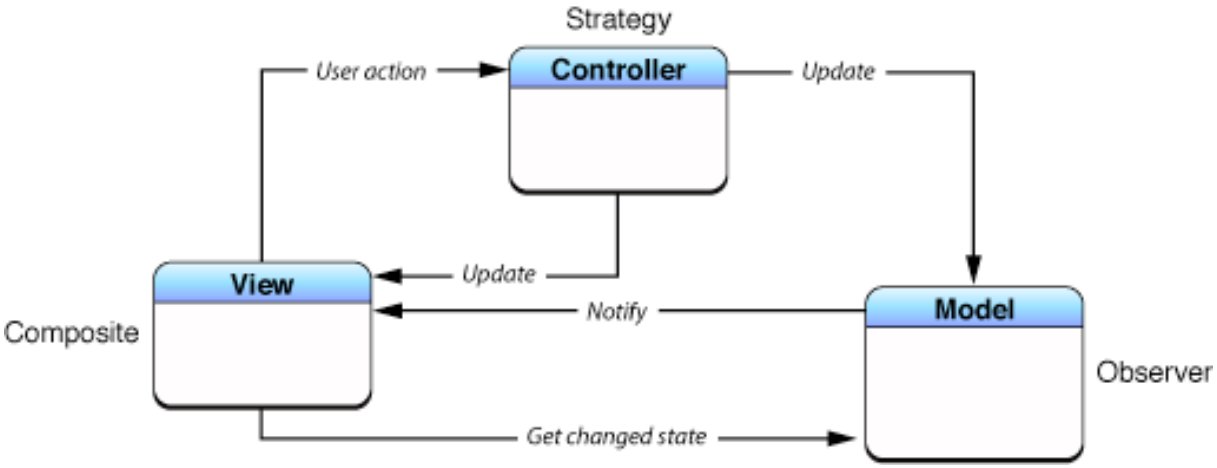

Figura 4: Versão tradicional do MVC. Fonte: MVC 2010

2.10.2.Modelo (Model)

Objetos de modelo representam conhecimentos específicos e conhecimentos especializados. Eles armazenam dados de um aplicativo e definem a lógica que manipula os dados. Uma aplicação MVC bem concebida tem todos os seus dados importantes encapsuladas em objetos de modelo. Os dados que fazem parte do estado persistente da aplicação (se esse estado persistente é armazenado em arquivos ou bancos de dados) deve residir no modelo de objetos, uma vez que os dados são carregados na aplicação. Porque eles representam o conhecimento e expertise de domínio relacionado a um problema específico, elas tendem a ser reutilizáveis.

Idealmente, um objeto de modelo não tem nenhuma ligação explícita com a interface de usuário usada para apresentar e editá-lo. Na prática, esta separação nem sempre é a melhor coisa, e há espaço para a flexibilidade aqui, mas em geral, um modelo de objeto não deve se preocupar com questões de interface e de apresentação (MVC 2010).

2.10.3.Visão (View)

Um objeto de visão sabe como exibir, e pode permitir que os usuários editem os dados do modelo do aplicativo. A visão não deve ser responsável por armazenar

os dados que está exibindo. Isso não significa que a visão nunca armazena os dados que são exibidos. A exibição de dados pode armazenar em cache ou fazer truques semelhantes, por razões de desempenho. Um objeto de exibição pode ser responsável por exibir apenas uma parte de um modelo de objeto, ou um modelo de objeto inteiro, ou mesmo muitos objetos de modelo diferente.

Objetos de visão tendem a ser reutilizáveis e configuráveis, e garantem a coerência entre as aplicações. A view deve garantir que está mostrando o modelo corretamente. Por isso, geralmente precisa saber sobre as alterações no modelo. Como os objetos do modelo não deve ser vinculado a objetos de visão específicos, necessita-se de uma forma genérica, que indique que eles mudaram (MVC 2010).

2.10.4.Controle (Controller)

Um objeto controlador atua como intermediário entre objetos de visão da aplicação e seus objetos de modelo. Os controladores são muitas vezes responsável por garantir que a view tenha acesso aos objetos do modelo necessários e agir como o canal através do qual as views tenham tenham conhecimento sobre as alterações no modelo. Controlador de objetos também pode realizar configurações, coordenar tarefas de uma aplicação e gerenciar o ciclo de vida de outros objetos.

Em um típico projeto MVC, quando o usuário entra um valor ou indicar uma escolha por um objeto de visão, o valor ou a escolha é comunicada a um objeto controlador. O objeto controlador pode interpretar a entrada do usuário, de algum modo específico e então pode dizer a um modelo de objeto o que fazer com essa entrada, por exemplo, "adicionar um novo valor" ou "excluir o registro atual". Alguns objetos de controle também pode contar uma visão de objeto mudar sua aparência ou comportamento, quando um modelo muda (MVC 2010).

3. PROJETO DO PROTOTIPO

Neste capítulo será apresentado o problema que motivou este trabalho, além as etapas realizadas antes da implementação do sistema proposto, que mostraram o caminho seguido e o amadurecimento da estrutura do sistema.

3.1.DESCRIÇÃO DO PROBLEMA

Atualmente não existe uma maneira simples para um desenvolvedor de conteúdo para educação a distância testar os seus aplicativos, geralmente utiliza-se uma máquina virtual com o Ginga-NCL instalado e todos os comandos para copiar os arquivos para a máquina virtual e depois rodar os arquivos .ncl são feitos manualmente através do SSH, o que se torna um processo demorado diminuindo a produtividade. O objetivo deste trabalho é automatizar este processo a fim de facilitar e agilizar o trabalho do criador de conteúdo.

3.2.PESQUISA

A proposta inicial deste projeto era instalar o Ginga-NCL em uma máquina virtual rodando uma das distribuição de Linux mais utilizadas, o Ubuntu, para que o programa desenvolvido se comunicasse diretamente com o Ginga-NCL. Após um bom tempo de estudos foram feitas várias investidas na tentativa se instalar o Ginga-NCL diretamente em uma máquina virtual rodando Ubuntu, a última, e a que chegou-se mais longe, foi abortada, pois houveram vários problemas na compilação dos módulos do Ginga-NCL, muitos dos quais necessitavam de versões mais antigas de bibliotecas e programas do que as disponibilizadas na versão do Ubuntu em uso, o que fez com que a compilação falhasse em diversos estágios, alguns destes problemas foram resolvidos, porem outros necessitariam de mudanças que não estão no escopo desse projeto.

Com o conhecimento adquirido durante os estudos, percebeu-se que esta abordagem não era a mais interessante, pois, além da dificuldade de instalação, era

muito limitada, uma vez que o usuário da ferramenta teria que rodar tudo de dentro da máquina virtual, e isso traz algumas dificuldades, como:

- O usuário teria que copiar, por linha de comando ou algum outro software, os arquivos da sua máquina host para a máquina virtual, e para um usuário não familiarizado com isto não é algo trivial;

- Para distribuir o sistema implementado neste projeto teria-se que disponibilizar um arquivo .vdi, que é a imagem do disco da máquina virtual, e na ultima tentativa de instalar o Ginga-NCL na máquina Virtual, este arquivo ultrapassou os 7GB de espaço em disco, o que dificultaria a distribuição;

- Dificuldade na atualização do Ginga-NCL, pois para atualizar o player seria necessário compilar o Ginga-NCL, o que exige um conhecimento avançado em Linux.

Por estas razões decidiu-se mudar a abordagem, utilizando então o Virtual Ginga-NCL para rodar os arquivos NCL e criando um software na linguagem Java que será instalado no sistema host do usuário e irá comunicar-se com o Virtual Ginga-NCL através do protocolo SSH para controlar a execução do conteúdo e copiar os arquivos necessários para a máquina virtual. Algumas vantagens da abordagem atual são:

- A Ferramenta é multiplataforma, pois é escrita em Java, e o Virtual Ginga-NCL pode rodar tanto no Linux e no Windows, através do VMWare Player, quanto no Mac OS X, através do VMWare Fusion;

- Maior escalabilidade, pois caso saia uma nova versão do Ginga-NCL, também será disponibilizado uma nova versão do Virtual Ginga-NCL, o que facilita muito a atualização;

- Mais fácil de configurar, pois o usuário terá apenas que configurar o Virtual Ginga-NCL, que é uma tarefa fácil, uma vez que conta com um manual detalhado na página do projeto, e executar o .jar do sistema implementado neste projeto.

3.3.REQUISITOS GERAIS DO SISTEMA

- Interpretar arquivos NCL;
- Exibir arquivos NCL;
- Ser fiel ao que o STB mostrará;
- Interface Parecida com a do Ginga Hiper;
- Fácil de usar;
- Controle Virtual;

3.4.MODELAGEM DO SISTEMA

Com o intuito de facilitar o entendimento geral do sistema a ser desenvolvido neste trabalho, antes da implementação de fato, foi feita a modelagem do mesmo, através de diagrama de casos de uso, Diagrama de classes e mock-up[1] da interface gráfica.

3.4.1.Diagrama de Casos de Uso

A partir da elaboração de um diagrama de casos de uso (Figura 5) foi possível levantar os requisitos básicos necessários para o desenvolvimento da aplicação.

[1] Molde para teste em tamanho natural feito com a finalidade de experimentar, ou seja, saber onde são os pontos fracos e fortes em uma idéia, um projeto.

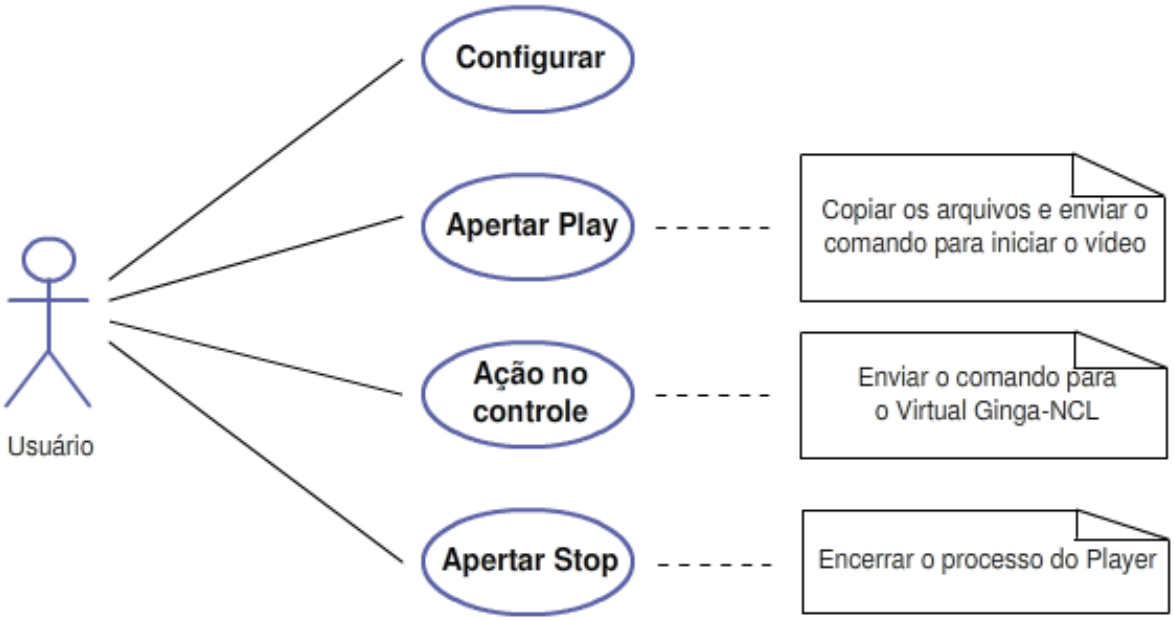

Figura 5: Diagrama de casos de uso da aplicação Fonte: Elaborado pelo autor.

Apenas um tipo usuário foi identificado para uso da aplicação, sendo este usuário denominado no diagrama como “Usuário”. Este usuário é a pessoa que deseja testar a aplicação gerada para TV Digital.

A partir da definição do usuário e próximo passo foi definir macro processo envolvidos. Quatro foram observados:

- **Configurar:** É o processo onde são fornecidas as informações básicas para que se possa rodar um projeto, são elas: ip da máquina virtual, usuário do sistema instalado na máquina virtual, senha deste usuário e o caminho da pasta do projeto onde encontram-se os arquivos que serão rodados.

- **Iniciar Exibição:** Neste processo a pasta do projeto é copiada para a máquina virtual e logo em seguida é enviado o comando para iniciar a exibição do conteúdo.

- **Controlar:** Este processo é iniciado a cada interação do usuário com o sistema, como clicar no botão vermelho, clicar no menu ou outra, enquanto o

conteúdo está sendo exibido, ele envia a ação do usuário para a máquina virtual que irá executá-la.

• **Finalizar Exibição:** Neste processo será enviado um comando para máquina virtual, para que ela finalize o processo do player.

3.4.2.Modelagem de Classes

Com o término da elaboração do diagrama de casos de uso o próximo passo foi a elaboração do diagrama de classes da aplicação. Estruturalmente a aplicação segue o padrão de projeto MVC, para que cada classe com o seu papel, permitindo um maior reaproveitamento de código, além da melhor organização. Na Figura 6 está representada a modelagem das classes.

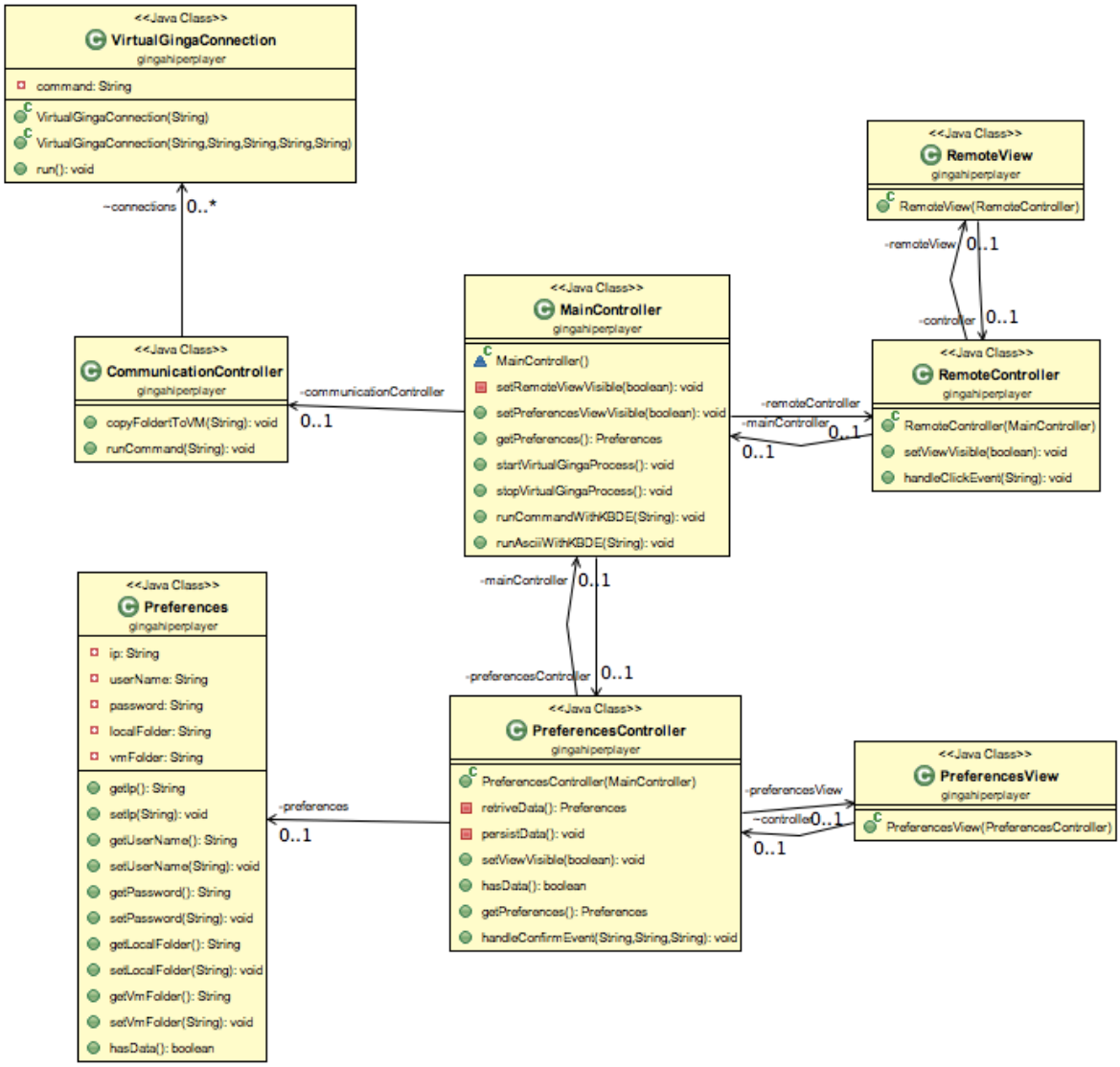

Figura 6: Diagrama de classes da aplicação Fonte: elaborado pelo autor.

3.4.3.Modelagem de Interface

Com o diagrama de classes pronto, foi possível fazer a modelagem das interfaces gráficas do sistema, a modelagem foi feita através de mock-up, que são modelos nos quais a interface real do sistema será baseada.

• **RemoteView:** É a tela principal do programa, é através dela que o usuário interage com o Virtual Ginga-NCL. Como pode ser visto na Figura 7, ela tem o design semelhante ao de um controle remoto convencional para ser mais intuitivo ao usuário.

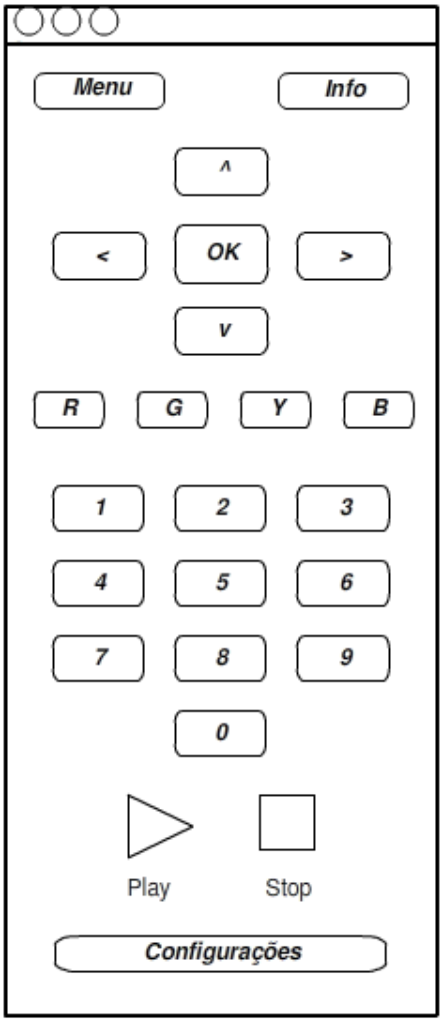

Figura 7: Mock-up da interface gráfica principal do sistema. Fonte: Elaborada pelo autor

• **PreferencesView:** Nesta tela é feita a configuração inicial das variáveis para que o sistema possa acessar o Virtual Ginga-NCL e rodar o projeto adequadamente. Esta interface possui um design simples, e por isto intuitivo.

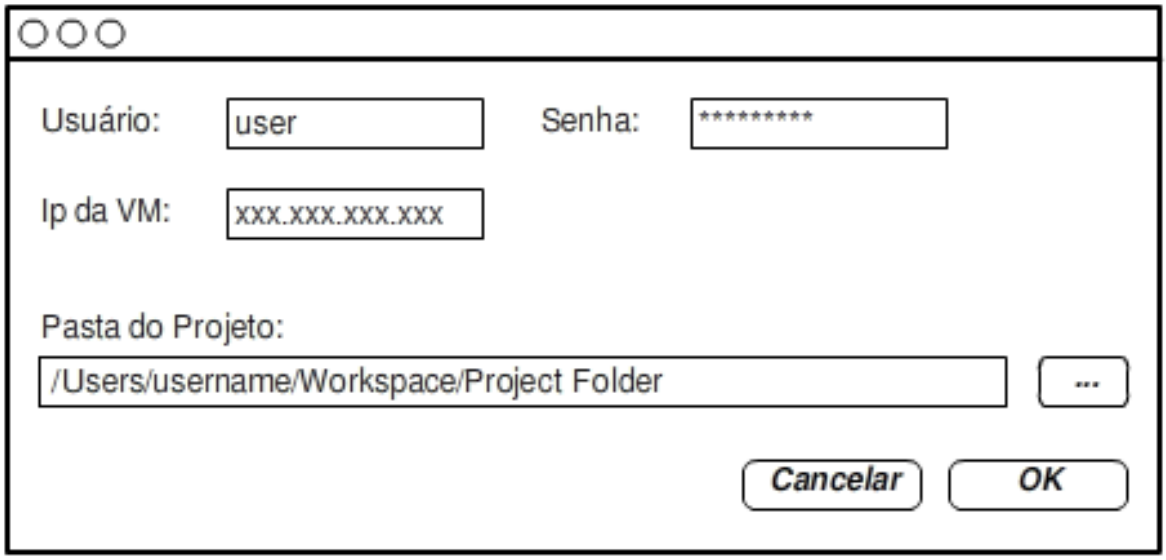

Figura 8: Mock-up da interface de configurações do sistema. Fonte: Elaborada pelo autor

3.5.FERRAMENTAS E TECNOLOGIAS UTILIZADAS

As ferramentas e tecnologias utilizadas no desenvolvimento do projeto são:

3.5.1.Java

A linguagem de programação Java, foi originalmente desenvolvida por James Gosling na Sun Microsystems, e lançada em 1995 como componente central da plataforma Java. Sua sintaxe deriva muito das linguagens C e C++, mas tem um modelo de objeto mais simples e menos recursos de baixo nível. Normalmente as aplicações escritas em Java são compiladas para bytecode (arquivos .class) que pode ser rodado em qualquer Java Virtual Machine (JVM), independente da arquitetura do computador (JAVA 2000).

Java é uma linguagem de propósito geral, baseada em classes, orientada a objetos, utiliza o conceito de concorrência e projetada para ter o menor numero de dependências de implementação, desta forma permite os desenvolvedores escreverem o código uma vez e rodarem em qualquer lugar (JAVA 2010).

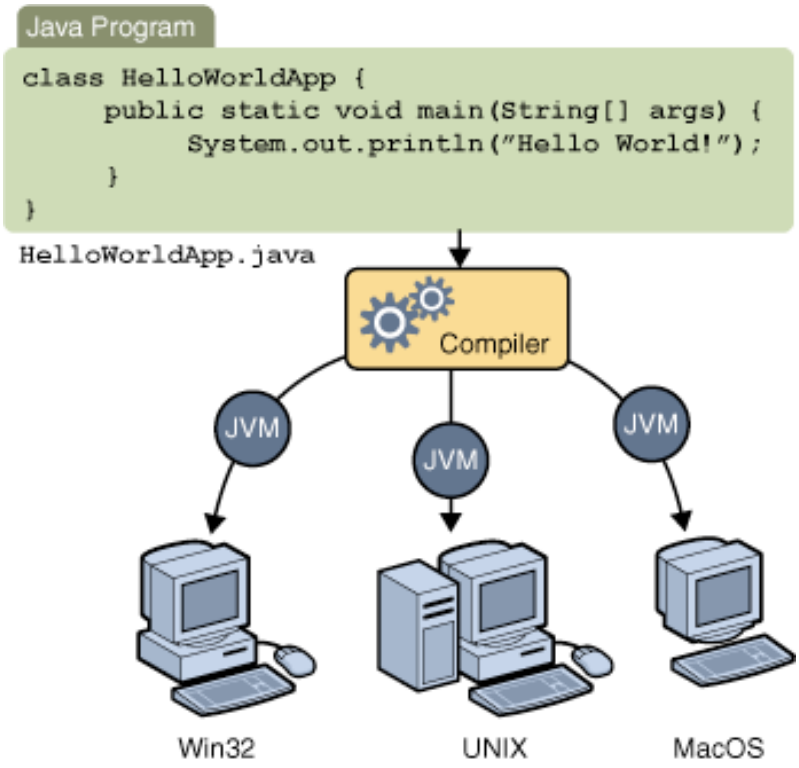

Figura 9: Através do Java, a mesma aplicação é capaz de rodar em diferentes ambientes. Fonte: JAVA 2010

Java é considerado por muitos como uma das linguagens de programação mais influentes do século 20, e aparece como a segunda linguagem mais utilizada no ano de 2010 (TIOBE 2010).

A aplicação será escrita na linguagem Java, pois esta é multi-plataforma, ou seja, um programa escrito em Java pode ser executado em qualquer sistema operacional que tenha uma máquina virtual Java implementada (Java 2010). Além de ser bastante popular, sendo atualmente a segunda linguagem mais utilizada no mundo (TIOBE 2010).

3.5.2.Netbeans

A IDE escolhida para o desenvolvimento do projeto foi o Netbeans 6.9 beta, pois é uma ferramenta bastante completa, fácil de ser manuseada e gratuita da própria Oracle, que é quem mantem o Java atualmente (NETBEANS 2010)

3.5.3.SSH

O Secure Shell ou SSH é um protocolo de rede que permite a troca de dados utilizando um canal seguro entre dois dispositivos de rede. Inicialmente utilizado no GNU/Linux e sistemas baseados no Unix para acessar contas shell, o SSH foi criado para substituir o Telnet e outros shells remotos inseguros, que enviam informações, como senhas, em forma de texto, deixando estas informações suscetíveis à análise de pacotes. A encriptação utilizada no SSH proporciona confidencialidade e integridade dos dados, mesmo em uma rede insegura, como a internet.

O SSH utiliza uma criptografia de chave pública para autenticar o computador remoto, e permite que o computador remoto autentique o usuário, se necessário.

O SSH utiliza o modelo cliente-servidor, e normalmente é utilizado para efetuar login em um computador remoto e executar comandos, mas ele também suporta tunelamento, forwarding de portas TCP e conexões X11, podendo transferir arquivos utilizando os protocolos SFTP e SCP.

Um programa cliente SSH é normalmente usado para estabelecer conexões, através da porta TCP 22, a um servidor SSH que aceita conexões remotas. Ambos são comumente encontrados na maioria dos sistemas operacionais modernos, incluindo Mac OS X, Linux, FreeBSD, Solaris e OpenVMS (SSH 2006).

A comunicação com o Virtual Ginga-NCL será feito através do protocolo SSH, por ser um protocolo seguro (SSH 2006) e por ser recomendado o uso na própria tela inicial do Virtual GingaNCL, como pode ser visto na Figura 3.

3.5.4.VMware

O software que rodará a máquina virtual será o VMWare, pois máquina Virtual fedora-fc7-ginga-i386 foi feita utilizando o VMWare Workstation 6. Uma vantagem de se utilizar o VMWare é o fato de ser possível rodar a máquina virtual tanto no Linux e no Windows, através do VMWare Player, quanto no Mac OS X, através do VMWare Fusion (VIRTUAL GINGA-NCL 2006).

3.5.5.KBDE

O KBDE é um emulador de teclado criado por Vaery Reznic, escrito na linguagem de programação C e distribuído sob a licença GPL, O kbde tem como objetivo simular via software um teclado real, através de um comando no terminal, por exemplo:

kbde --ascii="ls" --key=Enter --background

O comando anterior digita “ls” no terminal e envia o comando “Enter” (KBDE 2007).

O KBDE será instalado máquina virtual e tratará os comandos realizados pelo usuário através do controle remoto, como clique no botão “OK”, clique no botão vermelho entre outros. Estes comandos serão enviados através do protocolo SSH.

3.5.6.Virtual Ginga-NCL

É a implementação do player NCL na linguagem C++, e roda numa máquina virtual Linux, distribuição Fedora, Foi escolhido, pois conta com os mais avançados recursos de apresentação de aplicações declarativas, melhor desempenho e maior proximidade de uma implementação embarcada em set-top boxes reais (VIRTUAL GINGA-NCL 2006).

4. APRESENTAÇÃO DO PROTÓTIPO

Para fim de facilitar o entendimento, implementação do sistema pode ser dividida em duas partes, a implementação do protótipo em si, e a instalação e a configuração da máquina virtual que fará a simulação do set-top box.

4.1. MÁQUINA VIRTUAL

Para que o sistema funcione como planejado, é necessário a instalação do emulador de teclado virtual, o KBDE, na máquina virtual que simula o set-top box, ele que receberá os comandos enviados pelo usuário através do sistema instalado na máquina host do usuário.

O objetivo inicial era utilizar o Virtual Ginga-NCL para fazer a simulação do set-top box, mas ao tentar instalar o KBDE na máquina virtual percebe-se que o source utilizado para a compilação do kernel não foi disponibilizado com a mesma, o que impossibilitou a compilação do simulador de teclado virtual, para tentar contornar este problema foi utilizado o source original do fedora, mas novamente não foi obtido sucesso, pois o source utilizado no Virtual Ginga-NCL foi customizado. Como última tentativa, foi enviado um email para o laboratório Telemidia solicitando o source, mas não foi obtida nenhuma resposta.

Como não foi possível o uso do Virtual Ginga-NCL, a solução foi utilizar uma máquina virtual disponibilizada no site: http://manoelcampos.com/2010/01/28/virtualbox-vm-gingancl-0112-openginga-beta-1/, a máquina roda ubuntu 8.10 e já vem com o Ginga-NCL instalado e roda sobre o VirtualBox. Nesta máquina foi possível compilar e instalar o KBDE, para que o sistema funcione corretamente, também foram feitas algumas configurações para facilitar a utilização pelo usuário final, como login automático, configuração da interface de rede para acesso por SSH, entre outras.

4.2.SISTEMA

O sistema foi implementado na linguagem de programação Java, utilizando o padrão de projeto MVC. Para aumentar o reuso do código, não foram feitas ligações entre o Modelo e a Visão, ou seja, toda a comunicação entre elas é feita através da classe de Controle, além disso, para melhor estruturar o código, foram criadas três classes de controle: MainController, RemoteController e PreferencesController.

Visando facilitar a utilização do sistema pelo usuário, foi adicionada a funcionalidade de iniciar a máquina virtual automaticamente ao iniciar o programa, e para isto foram necessário algumas informações não contempladas na modelagem inicial do sistema, estas informações foram adicionadas na janela de configurações e são:

- **Nome da VM:** Nome da máquina virtual necessário para a inicialização automática da máquina virtual ao iniciar o sistema;
- **Arquivo Principal:** É o arquivo por onde o simulador do set-top box iniciará a execução do programa.
-

4.3.REQUISITOS PARA USO

O sistema foi desenvolvido utilizando-se a linguagem de programação multiplataforma Java. Os softwares desenvolvidos nesta linguagem podem ser executados em qualquer sistema operacional que possibilite a instalação da Máquina Virtual Java (Java Virtual Machine - JVM). Como exemplos de sistemas operacionais que apresentam este suporte estão o Microsoft Windows, o Linux, o Mac OS X, dentre outros.

4.4.APRESENTAÇÃO DOS RESULTADOS

Neste tópico será apresentado o sistema desenvolvido neste trabalho demonstrando as interfaces visuais e mostrando passo a passo as ações que o usuário final fará para testar os seus arquivos NCL.

4.4.1.Interfaces Visuais

O sistema possui duas janelas, a janela principal (Figura 10), onde o usuário realiza a interação com a máquina virtual que simula o set-top box, e uma outra (Figura 11) aonde ele configura o sistema para que ele possa fazer a comunicação co a VM corretamente.

Figura 10: Janela principal do sistema. Fonte: Elaborada pelo autor.

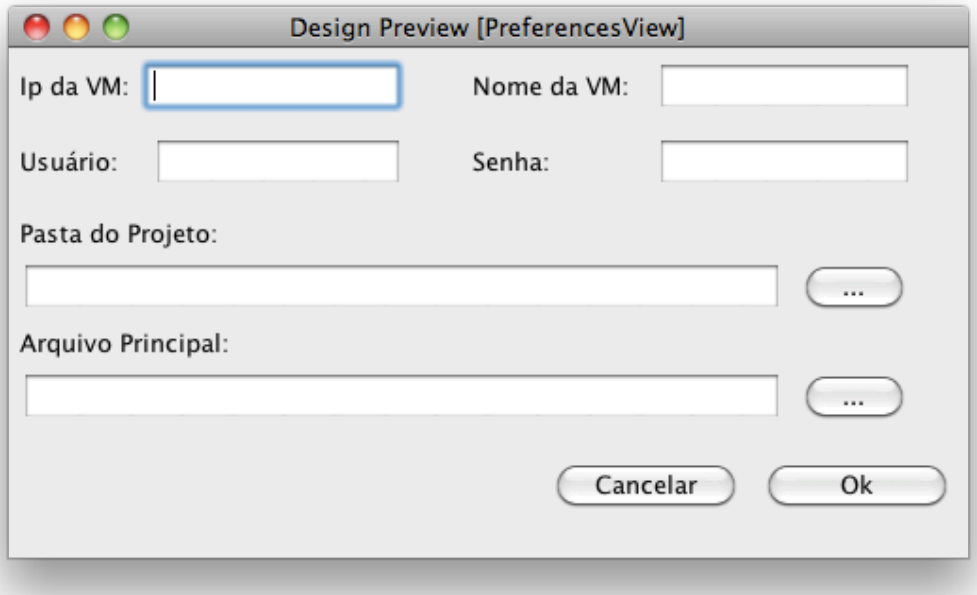

Figura 11: Janela de configuração do sistema. Fonte: Elaborada pelo autor.

4.4.2. Validação do Sistema

Neste tópico será feita uma breve descrição dos passos que o usuário deve fazer para utilizar o sistema desenvolvido neste trabalho e logo depois será feita a validação do mesmo utilizando um programa NCL.

Para utilizar o sistema primeiramente o usuário terá que configurar a máquina virtual que simulará o set-top box, para que o programa possa comunicar-se com ela, Este passo será feito somente na primeira vez que o sistema for utilizado. O VirtualBox deve ser configurado com os seguintes dados (Figura 12):

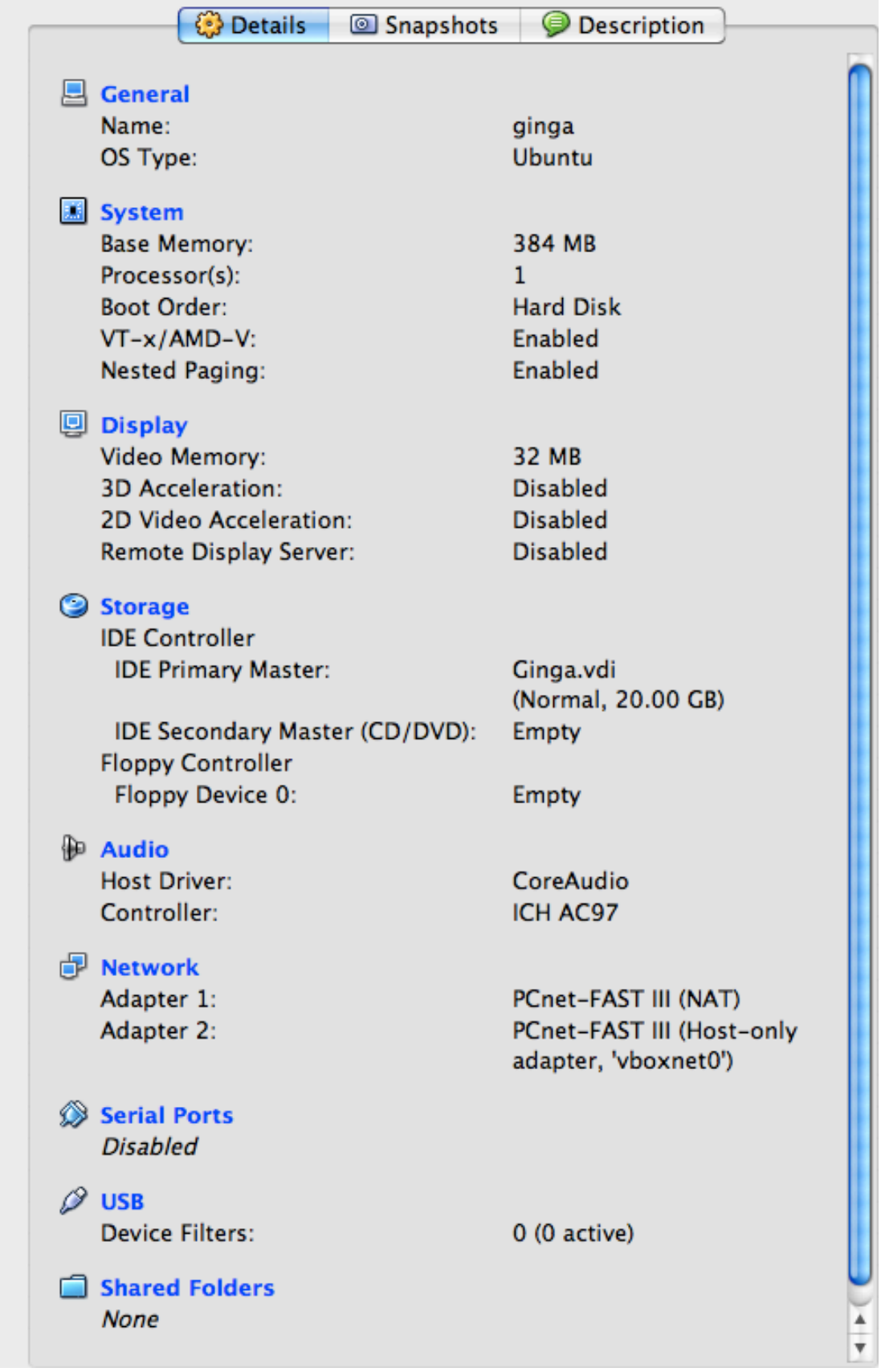

Figura 12: Janela de configuração do VirtualBox. Fonte: Elaborada pelo autor.

Os passos seguintes, o usuário irá executar todas vezes que desejar utilizar o sistema. Ele deve iniciar a máquina virtual configurada no passo anterior e assim que a mesma estiver iniciada aparecerá a tela mostrada na figura 13, então o usuário deve abrir o sistema criado neste trabalho utilizando o arquivo .jar disponibilizado.

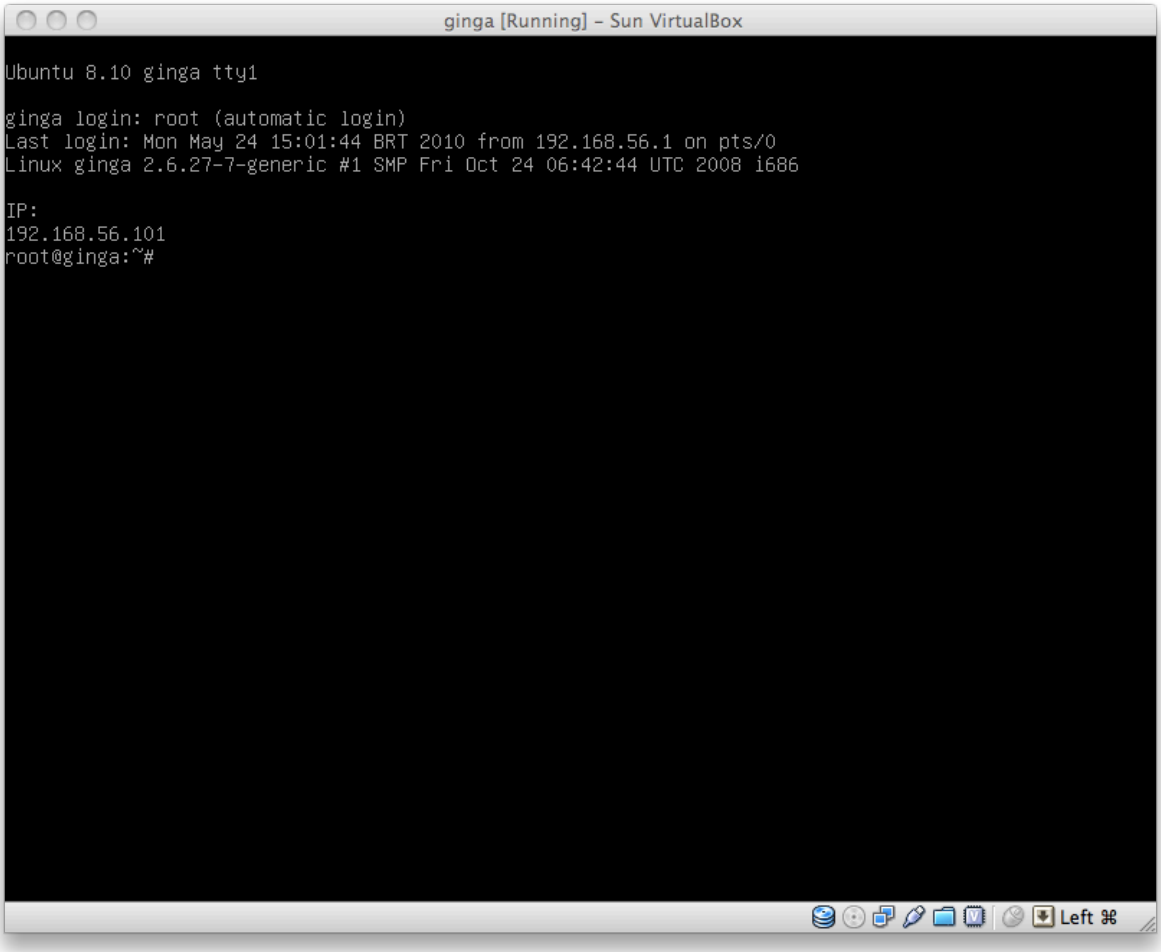

Figura 13: Tela inicial da máquina virtual. Fonte: Capturada pelo autor.

Ao abrir o sistema será mostrada a tela inicial (Figura 10), que é por onde o usuário controlará a máquina virtual, por ter o layout idêntico ao de um controle remoto, esta tela é muito intuitiva e fácil de ser utilizada.

A primeira vez que o usuário iniciar o sistema, ele deve entrar na tela de configurações (figura 14) para setar as informações necessárias para o funcionamento, são elas:

- **Ip da VM:** É o IP da máquina virtual,

- **Nome da VM:** O nome da máquina virtual configurada no VirtualBox, neste caso, como pode ser visto na figura 13, o nome da máquina é “ginga”;

- **Usuário:** O usuário que será utilizado para acessar a VM por SSH, que para esta máquina virtual é root.

- **Senha:** A senha de acesso para o usuário anterior, neste caso root;

- **Pasta do Projeto:** A pasta onde encontram-se os arquivos NCL e os outros arquivos necessários, como vídeos;

- **Arquivo principal:** O arquivo por onde será iniciada a execução.

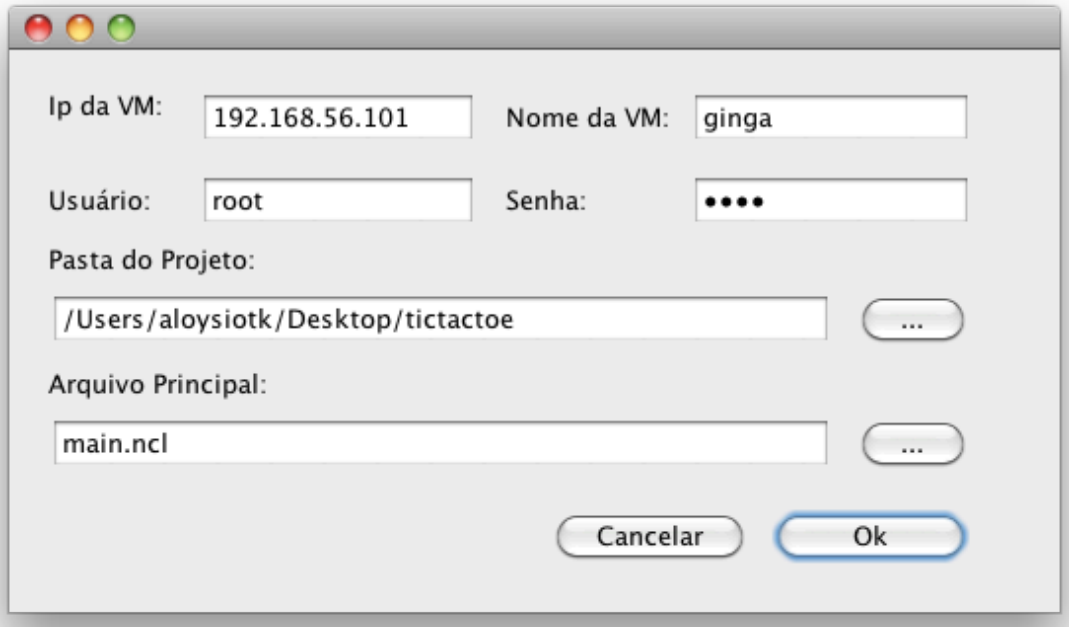

Figura 14: Tela de configurações do sistema preenchida. Fonte: Elaborada pelo autor.

Tendo configurado o sistema a primeira vez, o sistema salvará estas configurações para as próximas vezes que for executado.

Com estes passos feitos o sistema está configurado e pronto para ser utilizado, basta clicar no botão “play” na tela principal do sistema, que o arquivo começará a rodar como pode ser visto na figura 15.

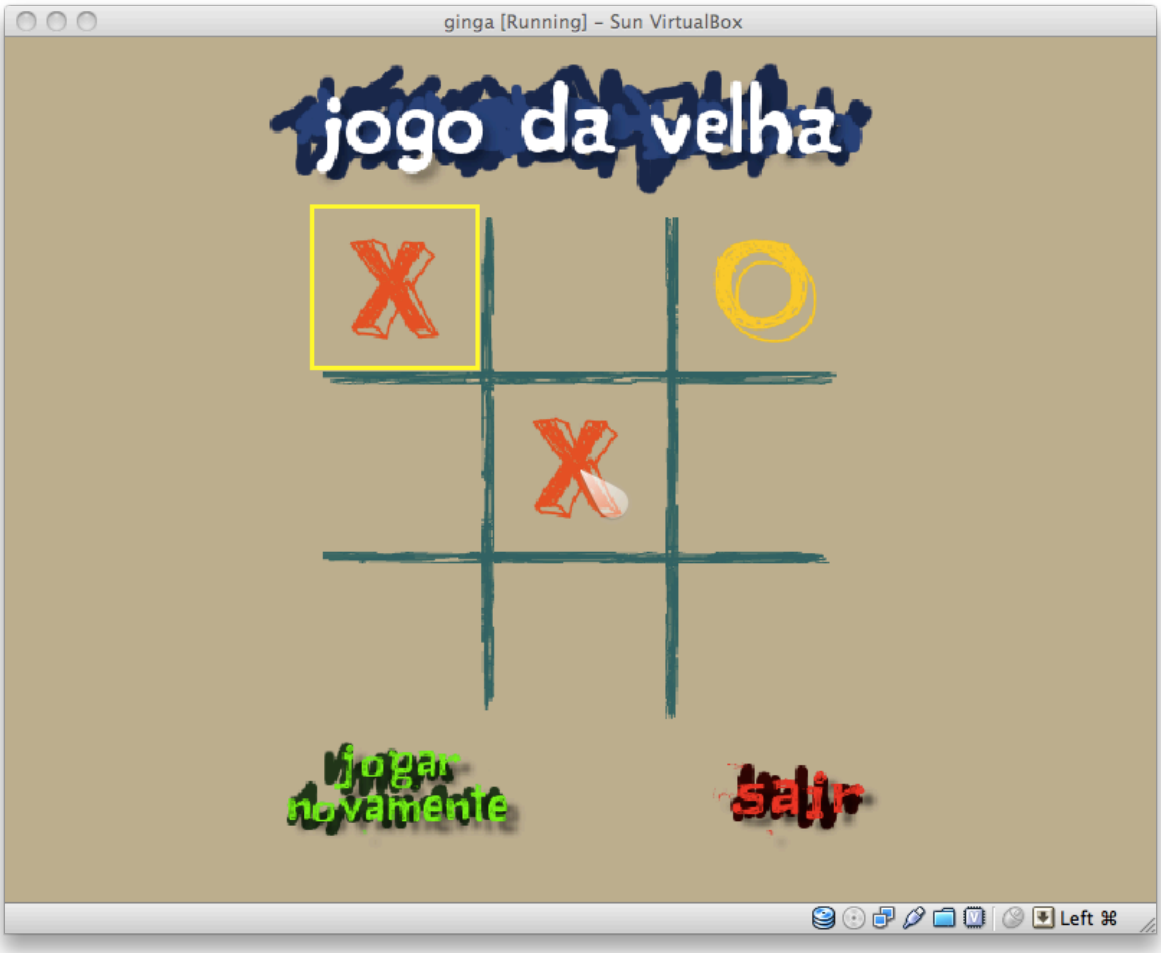

Figura 15: Programa NCL de Jogo da Velha sendo rodado na máquina virtual. Fonte: Capturada pelo autor.

5. CONCLUSÕES

A finalidade deste trabalho foi desenvolver uma ferramenta para que as pessoas que produzem conteúdo para t-learning possam, de uma maneira fácil, testar o material gerado, tendo em vista que muitas vezes estas pessoas não tem um conhecimento avançado na área de informática.

Durante o desenvolvimento deste trabalho, várias dificuldades foram encontradas devido a falta de materiais com informações técnicas sobre o Ginga-NCL, foram feitas várias tentativas de esclarecer essas dúvidas através de emails, fóruns e perguntas em lista de discussões, mas nenhuma delas obteve sucesso. Por este motivo foram feitas várias abordagens para atingir o objetivo deste trabalho.

A primeira abordagem foi a instalação do Ginga-NCL em uma máquina virtual rodando Ubuntu para que o programa que o programa desenvolvido se comunicasse diretamente com o Ginga-NCL. Ela foi abortada, pois houveram vários problemas na compilação dos módulos do Ginga-NCL, alguns resolvidos porem outros necessitariam de mudanças que não estão no escopo deste projeto.

A segunda foi utilizar o Virtual Ginga-NCL para rodar os arquivos NCL e criar um software na linguagem Java que seria instalado no sistema host do usuário e se comunicaria com o Virtual Ginga-NCL através do protocolo SSH para controlar a execução do conteúdo e copiar os arquivos necessários para a máquina virtual, mas ao tentar instalar o KBDE na máquina virtual percebeu-se que o source utilizado para a compilação do kernel não foi disponibilizado com a mesma, o que impossibilitou a compilação do simulador de teclado virtual, então foi enviado um email para o laboratório Telemidia solicitando o source, mas não foi obtida nenhuma resposta, o que viola a licença do kernel do linux, uma vez que ele é distribuído sob a licença GPL (General Public License) e ela diz que ao distribuir um programa licenciado pela GPL deve-se garantir que o usuário tenha acesso ao código, como pode-se ver no trecho retirado da própria licença:

> "... if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights."

A terceira abordagem e a que obteve sucesso, é basicamente igual a segunda, mas utilizando, no lugar do Virtual Ginga-NCL, uma máquina virtual encontrada na internet já com o Ginga-NCL instalada. Nesta máquina foi possível compilar e instalar o KBDE para fazer a comunicação com o Ginga-NCL.

Mesmo com todos os problemas encontrados, considerou-se que os objetivos estabelecidos para este trabalho foram atingidos, seguindo a metodologia apresentada. Foi apresentado o funcionamento básico da TV digital interativa, apresentado o estado da arte do middleware de TV digital brasileiro, o Ginga, tendo feito o embasamento teórico necessário, foi feita a modelagem do sistema e com a mesma madura, foi feita a implementação. Para testar e validar o sistema utilizou-se vários programas NCL disponibilizados online por desenvolvedores.

5.1.TRABALHOS FUTUROS

Os objetivos deste trabalho foram alcançados, porém a ferramenta desenvolvida ainda pode ser melhorada, para trabalhos futuros sugerimos:

- Enviar da imagem gerada pela máquina virtual através de VNC ou HTTP, o que possibilitaria a criação de um servidor, possivelmente com várias máquinas virtuais rodando, onde o gerador de conteúdo se conectaria para testar o seu material;

- Utilização do Virtual Ginga-NCL como simulador de set-top box

- Navegação entre nós do NCL;

6. REFERENCIAS

ABED (2006). Associação Brasileira de Educação a Distância, São Paulo. Disponível em: < http://www.abed.org.br>. Acessado em 16 de Novembro de 2009.

ABNT NBR 15602. Televisão digital terrestre: Codificação de vídeo, áudio e multiplexação. In: Normas Técnicas, ABNT. Rio de Janeiro, 2008.

ABNT, NBR 15605-5:2008. Televisão digital terrestre: codificação de dados e especificação de transmissão para radiodifusão digital. In: Normas Técnicas, ABNT, 121. Rio de Janeiro, 2008.

Aprenda NCL. Disponível em: <http://clube.ncl.org.br/?q=node/27> Acessado em: 20 de Outubro de 2009.

BARBOSA, S. D. J; SOARES, L. F. G. TV digital interativa no Brasil se faz com Ginga: fundamentos, padrões, autoria declarativa e usabilidade. Rio de Janeiro: PUC-Rio, 2008.

BATES, P. J. (2003). t-learning Study: A study into TV-based interactive learning to the home, Final Report, pjb Associates, UK, 2003.

BELLONI, M. L. (2002). Ensaio sobre a educação a distância no Brasil. Campinas: Revista Educação & Sociedade, ano XXIII, n. 78, abril / 2002. Disponível em: <http://www.scielo.br/pdf/es/v23n78/a08v2378.pdf>.

BITTENCOURT, Fernando. TV aberta brasileira DIGITAL: QUALIDADE E INTERATIVIDADE / IEL.NC, 2007.

COSENTINO, Laércio. Software: a essência da TV digital. TV DIGITAL: QUALIDADE E INTERATIVIDADE / IEL.NC.– Brasília : IEL/NC, 2007.

CROCOMO, Fernando. TV digital e produção interativa: a comunidade manda notícias. Florianópolis: Ed. da UFSC, 2007. 178 p., il.

DE SOUZA, FLÁVIO RICARDO DIAS; Estudo de Viadilidade de Educação a Distância Através de TV Digital Interativa. Recife: UFPE 2009.

DTV (2009); Cronograma de implantação da TV digital no Brasil. Disponível em: <http://www.dtv.org.br> Acessado em: 28 de Novembro de 2009.

GINGA (2009); TV Interativa se faz com Ginga. Disponível em: <http://www.ginga.org.br/>. Acessado em: 25 de Novembro de 2009.

GINGA-NCL (2009); Sobre o Ginga-NCL. 2009. Disponível em: <http://www.gingancl.org.br>. Acessado em: 28 de Novembro de 2009.

ICOER (2008); The Need for a Virtual Machine , Dept. of Computer Engineering, ICOER, Pune. Disponível em: <http://www.blog.unsri.ac.id/userfiles/23386952-Virtual-Machine.doc> Acessado em: 7 de maio de 2010.

JAVA (2000); Java Language Specification. Disponível em: <http://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601> Acessado em: 9 de Maio de 2010.

JAVA (2010); About the Java Technology. Disponível em: <http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html> Acessado em: 9 de Maio de 2010.

KBDE (2007); Man Page kbde. Disponível em: <http://kbde.sourceforge.net/kbde/man/kbde.1.html> Acessado em: 10 de Maio de 2010.

MONTEZ, C.; BECKER, V. (2005). TV Digital Interativa: conceitos, desafios e perspectivas para o Brasil. 2. ed. Florianópolis: Ed. da UFSC, 2005.

MVC (2010); Cocoa Design Patterns. Disponível em: <http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/CocoaFundamentals/CocoaDesignPatterns/CocoaDesignPatterns.html> Acessado em: 10 de Maio de 2010.

NCL (2009); Nested Context Language. Disponível em: <http://www.ncl.org.br>. Acessado em: 30 de Novembro de 2009.

NETBEANS (2010). NetBeans IDE 6.9 Beta Release Information. Disponível em : <http://netbeans.org/community/releases/69/> Acessado em: 18 de Abril de 2010.

NUNES, I. B. (1994). Noções de educação à distância. Revista de Educação à Distância nrs. 4/5, Dez./93-Abr/1994. Brasília; Instituto Nacional de Educação à Distância.

OLIVEIRA, C. T. de. (2005). Um estudo sobre os padrões de middleware para televisão digital interativa. Fortaleza: CEFET-CE, 2005. Monografia (Tecnólogo em Telemática), Centro Federal de Educação Tecnológica do Ceará. Ceará, 2005.

PNAD (2008); Pesquisa Nacional por Amostra de Domicílios. Disponível em: <http://www.ibge.gov.br/home/estatistica/populacao/trabalhoerendimento/pnad2008/sintesepnad2008.pdf>. Acessado em: 28 de Novembro de 2009.

POPEK, Gerald J.; GOLDBERG, Robert P. “Formal Requirements for Virtualizable Third Generation Architectures”, 1974.

SANT’ANNA, F.; CERQUEIRA, R.; SOARES, L. F. G. NCLua - Objetos Imperativos Lua na Linguagem Declarativa NCL. Rio de Janeiro: PUC-Rio, 2008.

SANTOS, A. C. O. dos. (2006). A digitalização da TV no Brasil: a sociedade civil organizada e a opinião pública a respeito do sistema brasileiro de TV digital – SBTVD. São Paulo: USP, 2006. Tese (Doutorado em Ciências da Comunicação) Escola de Comunicação e Artes, Universidade de São Paulo. São Paulo, 2006.

SANTOS, D. T. dos. (2007). Estudo de aplicativos de TVDi para educação a Distância. Campinas: UNICAMP, 2007. Dissertação (Mestrado em Engenharia Elétrica) Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas. São Paulo, 2007.

SMITH, James E.; NAIR, Ravi. "The Architecture of Virtual Machines". Computer (IEEE Computer Society), 2005.

SSH (2006); The Secure Shell (SSH) Authentication Protocol. Disponível em: <http://tools.ietf.org/html/rfc4252>. Acessado em: 20 de Abril de 2010.

TIOBE (2010); TIOBE Programming Community Index for April 2010. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acessado em: 28 de Abril de 2010.

VIRTUAL GINGA-NCL (2010). Detalhando o Set-top Box Virtual Ginga-NCL. Disponível em: <http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/gingancl/xowiki/gingancl_vm>. Acessado em: 28 de Abril de 2010.

7. ANEXOS

7.1.CÓDIGO FONTE DA APLICAÇÃO

MainController.java

```
package gingahiperplayer;

import java.io.IOException;

public class MainController {

    private CommunicationController communicationController;
    private PreferencesController preferencesController;
    private RemoteController remoteController;

    MainController() {
        try {
            Runtime.getRuntime().exec("VirtualBox --startvm ginga");
        } catch (IOException ex) {

        }

        communicationController = new CommunicationController();
        preferencesController = new PreferencesController(this);
        remoteController = new RemoteController(this);

        this.setRemoteViewVisible(true);
    }

    private void setRemoteViewVisible(boolean visible) {
       remoteController.setViewVisible(visible);
    }

    public void setPreferencesViewVisible() {
        preferencesController.setViewVisible();
    }

    public Preferences getPreferences() {
        return preferencesController.getPreferences();
    }

    public void startVirtualGingaProcess() {
        Preferences prefs = preferencesController.getPreferences();

        communicationController.runCommand("rm -r " + prefs.getVmFolder() + "/");

        communicationController.copyFoldertToVM(prefs.getIp(), prefs.getUserName(),
prefs.getPassword(), prefs.getLocalFolder(), prefs.getVmFolder());
```

```
        communicationController.playNclFile("/misc/launcher.sh " + prefs.getVmFolder()
+ "/" + prefs.getMainFile(), prefs.getIp(), prefs.getUserName(), prefs.getPassword());
    }

    public void stopVirtualGingaProcess() {
        Preferences prefs = preferencesController.getPreferences();

        communicationController.runCommand("killall -KILL ginga && sleep 1 && dfbg -
c 00000000");
    }

    public void  runCommandWithKBDE(String command) {
        Preferences prefs = preferencesController.getPreferences();

        communicationController.runCommand("kbde -d 100 -k " + command);
    }

    public void  runAsciiWithKBDE(String ascii) {
        Preferences prefs = preferencesController.getPreferences();

        communicationController.runCommand("kbde -d -a " + ascii);
    }

    public boolean isConnected() {
        return communicationController.isConnected();
    }

    public void changeConnectionData(String ip, String user, String pass) {
        communicationController.changeConnectionData(ip, user, pass);
    }
}
```

PreferencesViewController.java

```
package gingahiperplayer;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class PreferencesController {

   private static final String filename = "data.ghi";
```

```
    private Preferences preferences;
    private PreferencesView preferencesView;
    private MainController mainController;

    public PreferencesController(MainController controller) {
        this.mainController = controller;
        this.preferencesView = new PreferencesView(this);
        this.preferences = this.retriveData();

    }

    private Preferences retriveData() {

        if (new File(filename).exists()) {
            Preferences preferenceRetrived = null;
            FileInputStream fis = null;
            ObjectInputStream in = null;
            try {
                fis = new FileInputStream(filename);
                in = new ObjectInputStream(fis);
                preferenceRetrived = (Preferences) in.readObject();
                in.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            } catch (ClassNotFoundException ex) {
                ex.printStackTrace();
            }

            if (preferenceRetrived == null) {
                return new Preferences();
            } else {
                this.mainController.changeConnectionData(preferenceRetrived.getIp(),
preferenceRetrived.getUserName(), preferenceRetrived.getPassword());
                return preferenceRetrived;
            }
        } else {
            return new Preferences();
        }
    }

    private void persistData() {

        FileOutputStream fos = null;
        ObjectOutputStream out = null;

        try {
            fos = new FileOutputStream(filename);
            out = new ObjectOutputStream(fos);
            out.writeObject(preferences);
            out.close();
```

```
        } catch (IOException ex) {
            ex.printStackTrace();
        }

    }

    public void setViewVisible() {
        this.preferencesView.setVisibleWIthValues(preferences.getIp(),
preferences.getVmName(), preferences.getUserName(), preferences.getPassword(),
preferences.getLocalFolder(), preferences.getMainFile());
    }

    public boolean hasData() {
        return preferences.hasData();
    }

    public Preferences getPreferences() {
        return this.preferences;
    }

    public void handleConfirmEvent(String ip, String vmName, String userName,
String password, String localPath, String mainFile) {
        this.preferences.setIp(ip);
        this.preferences.setVmName(vmName);
        this.preferences.setUserName(userName);
        this.preferences.setPassword(password);
        this.preferences.setLocalFolder(localPath);
        this.preferences.setMainFile(mainFile);

        if (preferences.hasData()) {
            this.mainController.changeConnectionData(preferences.getIp(),
preferences.getUserName(), preferences.getPassword());
            this.persistData();
        }
    }
}
```

Preferences.java

```
package gingahiperplayer;

import java.io.File;
import java.io.Serializable;

public class Preferences implements Serializable {

    private static final String vmPath = "/misc/ncl30/";
```

```
private String ip;
private String vmName;
private String userName;
private String password;
private String localFolder;
private String vmFolder;
private String mainFile;

/**
 * @return the Virtual Machine ip
 */
public String getIp() {
    return ip;
}

/**
 * @param ip the Virtual Machine ip to set
 */
public void setIp(String ip) {
    this.ip = ip;
}

/**
 * @return the vmName
 */
public String getVmName() {
    return vmName;
}

/**
 * @param vmName the vmName to set
 */
public void setVmName(String vmName) {
    this.vmName = vmName;
}

/**
 * @return the userName
 */
public String getUserName() {
    return userName;
}

/**
 * @param userName the userName to set
 */
public void setUserName(String userName) {
    this.userName = userName;
}
```

```
/**
 * @return the password
 */
public String getPassword() {
    return password;
}

/**
 * @param password the password to set
 */
public void setPassword(String password) {
    this.password = password;
}

/**
 * @return the localFolder
 */
public String getLocalFolder() {
    return localFolder;
}

/**
 * @param localFolder the localFolder to set
 */
public void setLocalFolder(String localFolder) {
    this.localFolder = localFolder;

    File file = new File(localFolder);

    this.vmFolder = vmPath + file.getName();
}

/**
 * @return the vmFolder
 */
public String getVmFolder() {
    return vmFolder;
}

/**
 * @return the mainFile
 */
public String getMainFile() {
    return mainFile;
}

/**
 * @param mainFile the mainFile to set
 */
public void setMainFile(String mainFile) {
```

```
        this.mainFile = mainFile;
    }

    public boolean hasData() {
        boolean hasIp = !(this.ip.compareToIgnoreCase("") == 0);
        boolean hasVmName = !(this.vmName.compareToIgnoreCase("") == 0);
        boolean hasUserName = !(this.userName.compareToIgnoreCase("") == 0);
        boolean hasPassword = !(this.password.compareToIgnoreCase("") == 0);
        boolean hasLocalFolder = !(this.localFolder.compareToIgnoreCase("") == 0);
        boolean hasVmFolder = !(this.vmFolder.compareToIgnoreCase("") == 0);
        boolean hasMainFile = !(this.mainFile.compareToIgnoreCase("") == 0);

        return hasIp && hasVmName && hasUserName && hasPassword &&
hasLocalFolder && hasVmFolder && hasMainFile;
    }
}
```

PreferencesView.java

```
package gingahiperplayer;

import java.io.File;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class PreferencesView extends javax.swing.JFrame {

    PreferencesController controller;

    public PreferencesView(PreferencesController controller) {
        this.controller = controller;

        initComponents();

         this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);

    }

    public void showMessage(String message) {
         JOptionPane.showMessageDialog(this, message);
    }

    public void setVisibleWIthValues(String ip, String vmName, String user, String
pass, String localFolder, String mainFile) {
        this.jTextField1.setText(ip);
        this.jTextField2.setText(vmName);
        this.jTextField3.setText(user);
```

```
        this.jPasswordField1.setText(pass);
        this.jTextField4.setText(localFolder);
        this.jTextField5.setText(mainFile);

        this.setVisible(true);
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        jLabel2 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jLabel5 = new javax.swing.JLabel();
        jButton4 = new javax.swing.JButton();
        jTextField5 = new javax.swing.JTextField();
        jLabel6 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField();
        jLabel1 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel2.setText("Ip da VM:");

        jLabel4.setText("Pasta do Projeto:");

        jButton1.setText("...");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Cancelar");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton3.setText("Ok");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            jButton3ActionPerformed(evt);
        }
    });

    jLabel5.setText("Arquivo Principal:");

    jButton4.setText("...");
    jButton4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton4ActionPerformed(evt);
        }
    });

    jLabel6.setText("Nome da VM:");

    jLabel1.setText("Usuário:");

    jLabel3.setText("Senha:");

    org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout
(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(layout.createSequentialGroup()
            .addContainerGap()
            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
                .add(layout.createSequentialGroup()
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
                        .add(jLabel2)
                        .add(jLabel1))
                    .add(18, 18, 18)
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING, false)
                        .add(jTextField3)
                        .add(jTextField1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
142, Short.MAX_VALUE))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
                        .add(jLabel6)
                        .add(jLabel3))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING, false)
                        .add(jPasswordField1)
                        .add(jTextField2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
144, Short.MAX_VALUE))
```

```
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 1,
Short.MAX_VALUE))
                .add(jLabel4)
                .add(jLabel5)
                .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.TRAILING)
                        .add(layout.createSequentialGroup()
                            .add(jButton2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 106,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                            .add(jButton3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 106,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                        .add(layout.createSequentialGroup()
                            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING, false)
                                .add(jTextField5)
                                .add(jTextField4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 398, Short.MAX_VALUE))
                            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
                                .add(jButton1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 63,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                .add(jButton4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 63,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))))
                    .add(43, 43, 43)))
            .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.BASELINE)
                        .add(jTextField1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(jLabel6)
                        .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
```

```
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(jLabel2))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.LEADING)
                .add(layout.createSequentialGroup()
                    .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.BASELINE)
                        .add(jLabel1)
                        .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(jLabel3))
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                    .add(jLabel4))
                .add(jPasswordField1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(jTextField4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jButton1))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(jLabel5)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(jTextField5, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jButton4))
            .add(18, 18, 18)
            .add(layout.createParallelGroup
(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(jButton3)
                .add(jButton2))
            .add(23, 23, 23))
        );

        pack();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
```

```
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

        boolean hasIp = !(this.jTextField1.getText().compareToIgnoreCase("") == 0);
        boolean hasVmName = !(this.jTextField2.getText().compareToIgnoreCase("") ==
0);
        boolean hasUserName = !(this.jTextField3.getText().compareToIgnoreCase("")
== 0);
        boolean hasPassword = !(new String(this.jPasswordField1.getPassword
()).compareToIgnoreCase("") == 0);
        boolean hasLocalFolder = !(this.jTextField4.getText().compareToIgnoreCase("")
== 0);
        boolean hasMainFile = !(this.jTextField5.getText().compareToIgnoreCase("") ==
0);

        if (hasIp && hasVmName && hasUserName && hasPassword &&
hasLocalFolder && hasMainFile) {
            controller.handleConfirmEvent(this.jTextField1.getText(),
this.jTextField2.getText(), this.jTextField3.getText(), new String
(this.jPasswordField1.getPassword()), this.jTextField4.getText(),
this.jTextField5.getText());
            this.setVisible(false);
        } else {
            JOptionPane.showMessageDialog(this, "Preencha todos os campos...",
"Configurações", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
         JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

        fileChooser.setAcceptAllFileFilterUsed(false);

        fileChooser.showDialog(this, null);
        jTextField4.setText( fileChooser.getSelectedFile().getPath());
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

        File currentFolder = new File(jTextField4.getText());

        JFileChooser fileChooser = new JFileChooser();

        if (currentFolder.exists()) {
            fileChooser.setCurrentDirectory(currentFolder);
        }

        fileChooser.showDialog(this, null);
```

```
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);


         jTextField5.setText( fileChooser.getSelectedFile().getName());
    }


    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPasswordField jPasswordField1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;

}
```

RemoteController.java

```
package gingahiperplayer;

public class RemoteController {

    private RemoteView remoteView;
    private MainController mainController;

    public RemoteController(MainController controller) {
        this.mainController = controller;
        this.remoteView = new RemoteView(this);
    }

    public void setViewVisible(boolean visible) {
        this.remoteView.setVisible(visible);
    }

    public void handleClickEvent(String event) {

        if (RemoteHelper.getCommandByAction(event).compareToIgnoreCase("config")
== 0) {
```

```
            this.mainController.setPreferencesViewVisible();
        } else {
            if (mainController.isConnected()) {
                this.vmMessage(event);
            } else {
                this.remoteView.showMessage("Não foi possível conectar com a máquina
virtual.\nVerifique se a mesma está ligada ou se as configurações estão corretas.");
            }
        }
    }

    private void vmMessage(String event) {

        if (RemoteHelper.getCommandByAction(event).compareToIgnoreCase("play")
== 0) {

            this.mainController.startVirtualGingaProcess();

        } else if (RemoteHelper.getCommandByAction(event).compareToIgnoreCase
("stop") == 0) {

            this.mainController.stopVirtualGingaProcess();

        } else if (RemoteHelper.getCommandByAction(event).length() > 1) {

            this.mainController.runCommandWithKBDE
(RemoteHelper.getCommandByAction(event));

        } else {

            this.mainController.runAsciiWithKBDE(RemoteHelper.getCommandByAction
(event));

        }
    }
}
```

RemoteHelper.java

```
package gingahiperplayer;

import java.util.HashMap;

public class RemoteHelper {

        private static HashMap <String, String> eventToCommand = new HashMap
<String, String>();
```

```
    static {
        eventToCommand.put("0", "F5");
        eventToCommand.put("1", "F6");
        eventToCommand.put("2", "ArrowU");
        eventToCommand.put("3", "ArrowD");
        eventToCommand.put("4", "ArrowL");
        eventToCommand.put("5", "ArrowR");
        eventToCommand.put("6", "Enter");
        eventToCommand.put("7", "F1");
        eventToCommand.put("8", "F2");
        eventToCommand.put("9", "F3");
        eventToCommand.put("10", "F4");
        eventToCommand.put("11", "1");
        eventToCommand.put("12", "2");
        eventToCommand.put("13", "3");
        eventToCommand.put("14", "4");
        eventToCommand.put("15", "5");
        eventToCommand.put("16", "6");
        eventToCommand.put("17", "7");
        eventToCommand.put("18", "8");
        eventToCommand.put("19", "9");
        eventToCommand.put("20", "0");
        eventToCommand.put("21", "play");
        eventToCommand.put("22", "stop");
        eventToCommand.put("23", "config");
    }

    public static String getCommandByAction(String action) {
        return eventToCommand.get(action);
    }

}
```

RemoteView.java

```
package gingahiperplayer;

import javax.swing.JOptionPane;

public class RemoteView extends javax.swing.JFrame {

    private RemoteController controller;

    private RemoteView() {
        initComponents();
    }
```

```
    public RemoteView(RemoteController controller) {
        this.controller = controller;

        initComponents();
    }

    public void showMessage(String message) {
         JOptionPane.showMessageDialog(this, message);
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jButton5 = new javax.swing.JButton();
        jButton6 = new javax.swing.JButton();
        jButton7 = new javax.swing.JButton();
        jButton8 = new javax.swing.JButton();
        jButton9 = new javax.swing.JButton();
        jButton10 = new javax.swing.JButton();
        jButton11 = new javax.swing.JButton();
        jButton12 = new javax.swing.JButton();
        jButton22 = new javax.swing.JButton();
        jButton23 = new javax.swing.JButton();
        jButton24 = new javax.swing.JButton();
        jButton25 = new javax.swing.JButton();
        jButton26 = new javax.swing.JButton();
        jButton27 = new javax.swing.JButton();
        jButton28 = new javax.swing.JButton();
        jButton29 = new javax.swing.JButton();
        jButton30 = new javax.swing.JButton();
        jButton31 = new javax.swing.JButton();
        jButton13 = new javax.swing.JButton();
        jButton14 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(51, 51, 51));
        setResizable(false);
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jButton1.setText("Menu");
        jButton1.setSelectedIcon(new javax.swing.ImageIcon("/Users/aloysiotk/Public/
Drop Box/Untitled-1.png")); // NOI18N
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
```

```
        });
        getContentPane().add(jButton1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 80, -1));

        jButton2.setText("Info");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(141, 10, 80, -1));

        jButton3.setText("OK");
        jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent evt) {
                jButton3MousePressed(evt);
            }
        });
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(83, 80, 55, 40));

        jButton4.setText("^");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(83, 40, 55, 32));

        jButton5.setText(">");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(146, 84, 55, 32));

        jButton6.setText("<");
        jButton6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
```

```
        });
        getContentPane().add(jButton6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 84, 55, 32));

        jButton7.setText("v");
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(83, 128, 55, 32));

        jButton8.setBackground(new java.awt.Color(255, 0, 51));
        jButton8.setText("R");
        jButton8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton8ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 170, 50, -1));

        jButton9.setText("G");
        jButton9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton9ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(57, 170, 50, -1));

        jButton10.setText("Y");
        jButton10.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton10ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton10, new
org.netbeans.lib.awtextra.AbsoluteConstraints(114, 170, 50, -1));

        jButton11.setText("B");
        jButton11.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton11ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton11, new
org.netbeans.lib.awtextra.AbsoluteConstraints(170, 170, 50, -1));
```

```
        jButton12.setText("Configurações");
        jButton12.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton12ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton12, new
org.netbeans.lib.awtextra.AbsoluteConstraints(15, 450, 190, -1));

        jButton22.setText("1");
        jButton22.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton22ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton22, new
org.netbeans.lib.awtextra.AbsoluteConstraints(15, 220, 57, 37));

        jButton23.setText("2");
        jButton23.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton23ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton23, new
org.netbeans.lib.awtextra.AbsoluteConstraints(82, 220, 57, 37));

        jButton24.setText("3");
        jButton24.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton24ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton24, new
org.netbeans.lib.awtextra.AbsoluteConstraints(149, 220, 57, 37));

        jButton25.setText("4");
        jButton25.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton25ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton25, new
org.netbeans.lib.awtextra.AbsoluteConstraints(15, 260, 57, 37));

        jButton26.setText("5");
        jButton26.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton26ActionPerformed(evt);
            }
```

```
        });
        getContentPane().add(jButton26, new
org.netbeans.lib.awtextra.AbsoluteConstraints(82, 260, 57, 37));

        jButton27.setText("6");
        jButton27.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton27ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton27, new
org.netbeans.lib.awtextra.AbsoluteConstraints(149, 260, 57, 37));

        jButton28.setText("7");
        jButton28.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton28ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton28, new
org.netbeans.lib.awtextra.AbsoluteConstraints(15, 300, 57, 37));

        jButton29.setText("8");
        jButton29.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton29ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton29, new
org.netbeans.lib.awtextra.AbsoluteConstraints(82, 300, 57, 37));

        jButton30.setText("9");
        jButton30.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton30ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton30, new
org.netbeans.lib.awtextra.AbsoluteConstraints(149, 300, 57, 37));

        jButton31.setText("0");
        jButton31.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton31ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton31, new
org.netbeans.lib.awtextra.AbsoluteConstraints(82, 340, 57, 37));

        jButton13.setText("Stop");
```

```
        jButton13.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton13ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton13, new
org.netbeans.lib.awtextra.AbsoluteConstraints(115, 400, 80, -1));

        jButton14.setText("Play");
        jButton14.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton14ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton14, new
org.netbeans.lib.awtextra.AbsoluteConstraints(25, 400, 80, -1));

        pack();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("0");
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("1");
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("2");
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("3");
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("4");
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("5");
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("6");
    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        this.controller.handleClickEvent("7");
```

```
}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("8");
}

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("9");
}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("10");
}

private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("11");
}

private void jButton23ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("12");
}

private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("13");
}

private void jButton25ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("14");
}

private void jButton26ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("15");
}

private void jButton27ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("16");
}

private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("17");
}

private void jButton29ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("18");
}

private void jButton30ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("19");
}
```

```
private void jButton31ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("20");
}

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("21");
}

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("22");
}

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
    this.controller.handleClickEvent("23");
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new RemoteView().setVisible(true);
        }
    });
}

private javax.swing.JButton jButton1;
private javax.swing.JButton jButton10;
private javax.swing.JButton jButton11;
private javax.swing.JButton jButton12;
private javax.swing.JButton jButton13;
private javax.swing.JButton jButton14;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton22;
private javax.swing.JButton jButton23;
private javax.swing.JButton jButton24;
private javax.swing.JButton jButton25;
private javax.swing.JButton jButton26;
private javax.swing.JButton jButton27;
private javax.swing.JButton jButton28;
private javax.swing.JButton jButton29;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton30;
private javax.swing.JButton jButton31;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JButton jButton9;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
```

```
    private javax.swing.JLabel jLabel3;
}
```

VirtualGingaCommand.java

```
package gingahiperplayer;

import java.io.*;

import ch.ethz.ssh2.Connection;
import ch.ethz.ssh2.Session;
import ch.ethz.ssh2.StreamGobbler;

public class VirtualGingaCommand extends Thread {

    private String command;
    private String ip;
    private String user;
    private String pass;


    public VirtualGingaCommand(String command, String ip, String user, String pass)
{
        this.command = command;
        this.ip = ip;
        this.user = user;
        this.pass = pass;
    }

    @Override
    public void run() {
        try {
            Connection conn = new Connection(ip);

            conn.connect();

            boolean isAuthenticated = conn.authenticateWithPassword(user, pass);

            if (isAuthenticated == false) {
                throw new IOException("Authentication failed.");
            }

            Session sess = conn.openSession();

            sess.execCommand(command);

            InputStream stdout = new StreamGobbler(sess.getStdout());
```

```
            BufferedReader br = new BufferedReader(new InputStreamReader(stdout));

            while (true) {
                String line = br.readLine();
                if (line == null) {
                    break;
                }
            }

            sess.close();

            conn.close();

        } catch (IOException e) {
            e.printStackTrace(System.err);
            System.exit(2);
        }
    }
}
```

Main.java

```
package gingahiperplayer;

public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        MainController mainController = new MainController();
    }

}
```

CommunicationController.java

```
package gingahiperplayer;

import com.jscape.inet.scp.Scp;
import com.jscape.inet.ssh.util.SshParameters;

import ch.ethz.ssh2.Connection;
import ch.ethz.ssh2.Session;
import ch.ethz.ssh2.StreamGobbler;
import java.io.BufferedReader;
```

```
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class CommunicationController {

    private Connection conn;
    private boolean connected;

     public CommunicationController() {
        connected = false;
     }

    public void changeConnectionData(String ip, String user, String pass) {
       if (this.isConnected()){
            conn.close();
       }

       conn = new Connection(ip);

        try {
            conn.connect();

            boolean isAuthenticated = conn.authenticateWithPassword(user, pass);

            if (isAuthenticated == false) {
                connected = false;
                throw new IOException("Authentication failed.");
            } else {
                connected = true;
            }

        } catch (IOException ex) {
            connected = false;
            ex.printStackTrace(System.err);
        }
    }

    public void copyFoldertToVM(String hostname, String username, String password,
String source, String destination)  {

        try {

        SshParameters params = new SshParameters(hostname, username,
password);

        Scp scp = new Scp(params);
```

```
            scp.connect();

            scp.uploadDir(new File(source), destination);

            scp.disconnect();
             } catch (Exception e) {
                e.printStackTrace(System.err);
            }
        }


        public void runCommand(String command) {
            try {

                Session sess = conn.openSession();

                sess.execCommand(command);

                InputStream stdout = new StreamGobbler(sess.getStdout());

                BufferedReader br = new BufferedReader(new InputStreamReader(stdout));

                while (true) {
                    String line = br.readLine();
                    if (line == null) {
                        break;
                    }
                }

                sess.close();

            } catch (IOException e) {
                connected = false;
                e.printStackTrace(System.err);
            }
        }

        public void playNclFile(String command, String ip, String user, String pass) {
            VirtualGingaCommand newConnection = new VirtualGingaCommand(command,
    ip, user, pass);
            newConnection.start();
        }


        public boolean isConnected() {
            return connected;
        }
    }
```

7.2.ARTIGO

Estudo, Modelagem e Adaptação de um Player Ginga-NCL Para a Construção de Conteúdo em T-Learning

Aloysio Nandi Tiscoski[1]

[1]Departamento de Informática e Estatística - INE
Universidade Federal de Santa Catarina - UFSC
Caixa Postal 476 - 88.040-900 - Florianópolis - SC - Brasil

aloysio@inf.ufsc.br

Abstract. *The system of broadcasting television signals in Brazil is in the process of transition, where the analog and digital technologies are being transmitted in parallel. A major goal of the deployment of Digital TV in this country is the social inclusion for millions of Brazilians by access to digital technology once the television is present in more than 90% of Brazilian households. Another important goal is the universalization of distance education through the provision of T-learning applications, which will provide education to a large population. The aim of this paper is to present the completion of course work with this same title, which is proposed to construct an NCL player to simulate a set-top box, which allows content creators to test distance education material in a easy way, in the production environment. To achieve this goal was created a tool who communicates with the Ginga-NCL installed on a virtual machine through the SSH protocol, where you can test the generated content.*

Resumo. *O sistema de difusão de sinais de televisão no Brasil está em pleno processo de transição, onde as tecnologias analógica e digital estão sendo transmitidas paralelamente. Um dos maiores objetivos da implantação da TV Digital no país é a inclusão social para milhões de brasileiros por meio do acesso à tecnologia digital, uma vez que a televisão está presente em mais de 90% lares brasileiros. Outro importante objetivo é a universalização da educação à distância pela disponibilização de aplicativos de T-Learning, que propiciarão educação a uma grande parte da população. O objetivo deste artigo é a apresentação do trabalho de conclusão de curso de mesmo título deste, que tem como proposta a construção um player NCL para simulação de um set-top box, que possibilite aos criadores de conteúdo para educação a distância testar o material gerado de uma maneira fácil, em ambiente de produção. Para atingir este objetivo foi criada uma ferramenta que através do protocolo SSH se comunica com o Ginga-NCL instalado em uma máquina virtual, onde é possível testar o conteúdo gerado.*

1. Introdução

Segundo dados da PNAD (Pesquisa Nacional por Amostra de Domicílios) do IBGE (Instituto Brasileiro de Geografia e Estatística), realizada no ano de 2008, 95,1% dos lares brasileiros possuem pelo menos um aparelho de televisão, sendo assim o meio de transmissão de informação mais abrangente em nosso país, e para muitos brasileiros, é o único meio de adquirirem conhecimento, informação e cultura.

Com o advento da TV digital interativa (TVDI) muitos serviços que atualmente não são possíveis de serem implementados, devido as limitações tecnológicas da TV analógica tornam se viáveis. As possibilidades vão desde a criação de novos modelos de negócios seguindo para a abertura de novos horizontes na prestação de diversos tipos de serviços em diferentes seguimentos da sociedade, inclusive no âmbito educacional.

Uma importante característica advinda da digitalização, que tem incentivado muitos dos países no desenvolvimento de alternativas para implantação da mesma, é a possibilidade da "convergência" entre vários meios de comunicação, e mais a interatividade, a partir de um único equipamento (SANTOS, 2006).

O principal objetivo da implantação da TV Digital (TVD) no Brasil vai além de oferecer à população a possibilidade de receber em suas casas imagens e som de alta qualidade. O Decreto Presidencial No 4.901, de 26 de novembro de 2003, que instituiu o Sistema Brasileiro de Televisão Digital (SBTVD-T) destaca: "I - promover a inclusão social, a diversidade cultural do País e a língua pátria por meio do acesso à tecnologia digital, visando à democratização da informação; e II - propiciar a criação de rede universal de educação à distância".

Um dos grandes desafios a ser superado com a chegada da TVD é a produção de conteúdo interativos de qualidade, bem como, a formação de mão-de-obra especializada para os diversos fins que exige essa nova tecnologia, isso em curto espaço de tempo, pois o sinal já está sendo transmitido em 26 cidades do país.

2. TV Digital

Os constantes avanços tecnológicos focaram a melhora da qualidade da imagem dos televisores, o tamanho das telas e dos novos layouts. Deste modo apareceram no mercado os aparelhos de TV de alta definição, como que preparando um ambiente básico para aportar uma nova forma de conteúdos televisivos que estaria por vir: os conteúdos digitais.

De acordo com Oliveira (2005, p.8), muitas são as novidades na TV digital com relação a analógica, quais sejam:

- Qualidade do som comparado ao Compact Disk (CD);
- Qualidade da imagem no padrão do Digital Vídeo Disk (DVD) player com 525 linhas;
- Aumento nas transmissões de áudio relacionadas aos programas, como a dublagem e a legenda em diversas línguas;
- Dados transmitidos junto aos programas;
- De acordo com as características geográficas locais, ampliação da área coberta o que possibilita flexibilidade nos parâmetros de transmissão;
- Novos serviços: comércio eletrônico, acesso à Internet, educação à distância, guia eletrônico de programação, entre outros;
- Ampliação do número de canais;
- Programação variada;

- Single Frequency Network (SFN), uma tecnologia capaz de definir, por exemplo, um único número de canal para todo o Brasil;
- Possibilidade de gravação digital com alto desempenho para conteúdos multimídias preferidos pelo usuário: filmes, programas, imagens e fotos;
- Número maior de aplicações e serviços interativos englobando áudio, vídeo, texto
- e os mais diversos tipos de elementos gráficos; e
- Possibilidade de seleção, por parte do telespectador, do conteúdo que deseja ver, como, quando e onde.

Atualmente 10 países já fizeram a transição completa da transmissão analógica para Digital, e muitos países tem planos para fazer esta mudança nos próximos anos, outros, como o Brasil, estão no meio deste processo.

2.1. Componentes da TV Digital

Segundo Santos (2007) um sistema de TV Digital pode ser composto por uma emissora (estúdio) de TV digital que é onde o conteúdo é gerado e transmitido, um set-top box que recebe e converte os dados transmitidos pela emissora e por um aparelho analógico de TV, onde os dados recebidos pelo set-top box serão exibidos.

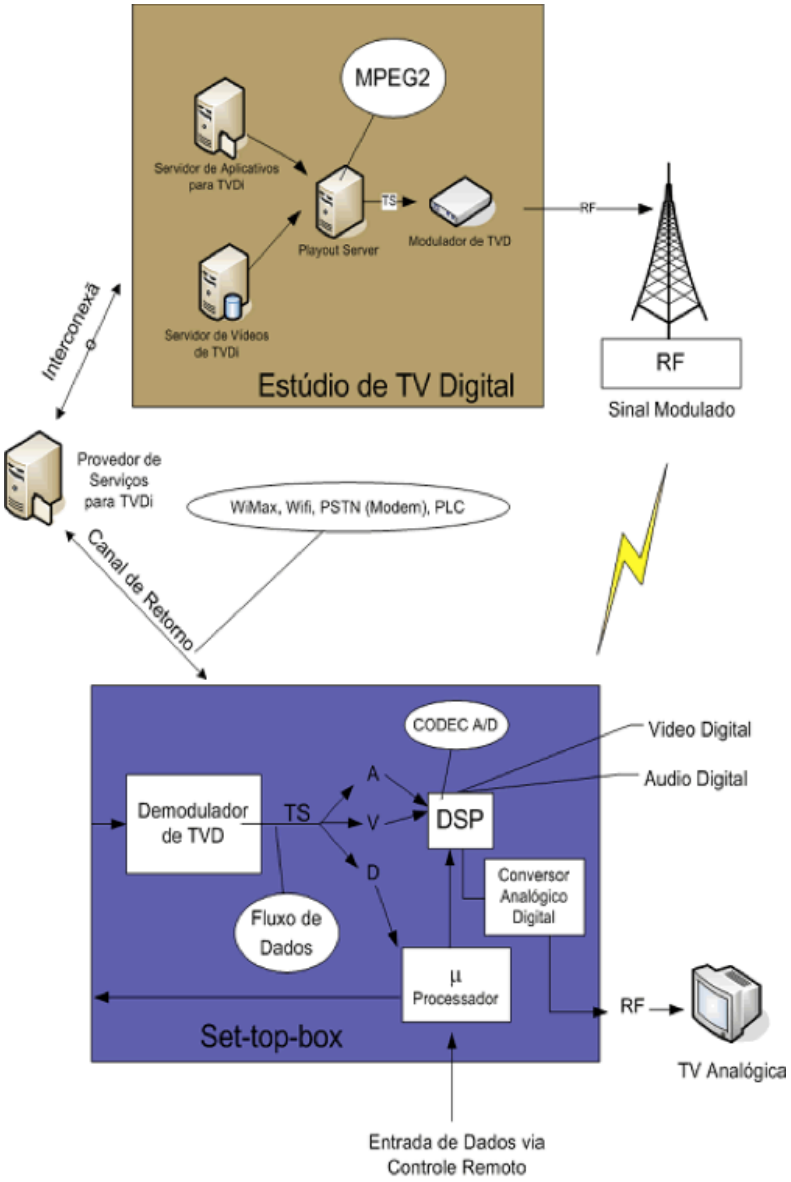

Figura 1. Visão Geral de um Sistema de TV Digital.

2.1.1. Emissoras

É quem transmite o conteúdo ao telespectador. elas devem possuir 2 servidores, sendo um de vídeos de TVDi e um outro para armazenar os sistemas aplicativos de TVDi que são associados aos vídeos. Os vídeos poderão ser disponibilizados em definição padrão (Standard Definition - SD) ou em alta definição (High Definition - HD), os vídeos são enviados para um servidor central antes de serem modulados.

É função do servidor de aplicativos gerar um Transport Stream (TS) que conterá vídeo, áudio e dados associados, enviá-los para um modulador que transforma estes dados em um sinal radiofrequência (Radio Frequency - RF). Este sinal será enviado a uma antena através de broadcast para as residências dos telespectadores.

O STB demodula os sinais de RF originados nas emissoras de TVD e extrai o fluxo de dados (áudio, vídeo e dados), ou TS. Os aplicativos são executados a partir de um micro

processador existente nos STB e o telespectador poderá interagir com esses aplicativos através de um controle remoto ou outro dispositivo de entrada, como por exemplo, um teclado sem fio (Santos 2007).

2.1.2. Middleware

O middleware é a camada de software responsável pela interface entre os aplicativos digitais interativos e o sistema operacional do STB. Diversos padrões de middleware foram desenvolvidos, cada qual referente a um tipo específico de TV digital, que privilegia certos aspectos em relação a outros. Normalmente, nos ambientes de desenvolvimento para TV Digital usa-se dois paradigmas de programação: o paradigma procedural e o declarativo. O modelo procedural baseia-se na utilização de uma "máquina virtual", ou Virtual Machine (VM), na qual os aplicativos são executados, isso requer um maior poder de processamento por parte do STB, um exemplo disso é o uso da linguagem de programação Java no Ginga-J. Já o modelo declarativo, é caracterizado pelo uso de linguagens declarativas, como por exemplo a eXtensible Hypertext Markup Language (XHTML), Nesse modelo, quem defini o estilo de apresentação e os conteúdos é a linguagem de marcação com o objetivo de suprir as necessidades específicas do broadcast de dados.

Um middleware para aplicações de TV digital basicamente consiste de máquinas de execução das linguagens oferecidas e bibliotecas de funções, que permitem o desenvolvimento rápido e fácil de aplicações interativas para TV digital. Essas aplicações vão possibilitar, por exemplo, acesso à internet, operações bancárias, envio de mensagens para o canal de TV ao qual se está assistindo, entre outros (Santos 2007).

2.1.3. Canal de Retorno

É um canal de transmissão através do qual o telespectador pode enviar e receber informações personalizadas pelo Provedor de Serviços de TVDi. O provedor está ligado tanto ao estúdio de TV Digital quanto ao telespectador, podendo receber as solicitações originadas pelo telespectador e responder às requisições efetuadas.

O canal de retorno é um componente muito importante em um sistema de TVD, pois possibilita a execução de uma grande variedade de funcionalidades e serviços, além de permitir o recebimento e envio de informações de forma personalizada.

2.1.4 Set-top Box

O receptor digital é um dispositivo capaz de converter os sinais digitais enviados pela emissora para que possam ser assistidos nos aparelhos convencionais de TV. Esse dispositivo pode vir embutido no aparelho televisor ou ser adquirido à parte, dessa forma o receptor é chamado de STB. Este nome se deve ao detalhe de que o dispositivo ter um formato de uma caixa e, geralmente, ficar sob a TV (MONTEZ; BECKER, 2005).

2.2. TV Digital no Brasil

O Sistema Brasileiro de TV Digital Terrestre foi baseado no sistema japonês, o Integrated Services Digital Broadcasting Terrestrial (ISDB-T) e é tecnicamente conhecido como Sistema Brasileiro de TV Digital Terrestre (SBTVD-T), ele proporciona uma série de diferenciais quando comparado aos outros sistemas atualmente utilizados no mundo. Esses diferenciais estão na junção da base técnica de transmissão do sistema japonês com os padrões de compressão digital de áudio e vídeo introduzidos pelo Brasil, que são mais modernos e eficientes do que os adotados por outros padrões, um outro grande diferencial é a utilização do middleware Ginga.

Na versão brasileira foram acrescentadas tecnologias desenvolvidas pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB).

Além disso, a especificidade do sistema brasileiro possibilita a transmissão de conteúdo de alta qualidade, tanto em termos de imagem como de som, permitindo ao mesmo tempo a recepção móvel e portátil dos sinais de TV digital. Para oferecer esses diferenciais, o SBTVD-T adotou o padrão MPEG-4, também conhecido como H.264, para codificação de vídeo, e o HE-AAC v2 para o áudio (DTV 2009).

Outros importantes diferenciais do SBTVD-T são a mobilidade e a interatividade. No caso da mobilidade é possível percebê-la na prática, uma vez que já estão à disposição do consumidor brasileiro diversos dispositivos móveis por meio dos quais se pode assistir à TV digital, como celulares, mini-televisores e receptores USB.

Em relação à interatividade, os documentos sobre o middleware Ginga da TV digital brasileira estão em processo de Consulta Nacional da ABNT. Para garantir que o Ginga esteja livre do pagamento de royalties.

A TV digital começou a ser transmitida no Brasil em 02 de dezembro de 2009 e tem previsão de ser concluída no dia 29 de junho de 2016, quando substituirá completamente a transmissão analógica, neste período a digitalização está sendo feita em etapas, começando pelas grandes capitais.

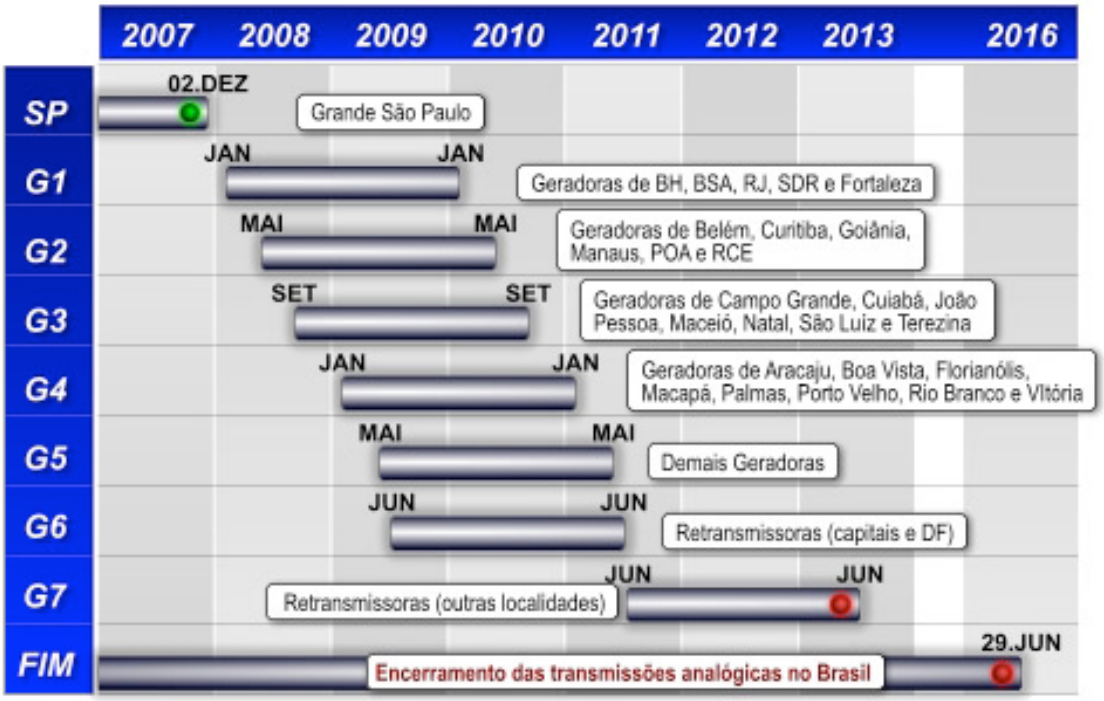

Figura 2. Cronograma de implantação da TV Digital no Brasil.

2.3. Ginga

Ginga é o middleware de especificação aberta adotado pelo Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) ele será instalado em conversores (set-top boxes) e em televisores. É uma camada de software intermediária, entre o sistema operacional e as aplicações. Sua principais funções são: tornar as aplicações independentes do sistema operacional da plataforma de hardware utilizados e oferecer um melhor suporte ao desenvolvimento de aplicações. Ou seja, é o Ginga que dá suporte a interatividade.

O Ginga é constituído por um conjunto de tecnologias padronizadas e inovações brasileiras que o tornam a especificação de middleware mais avançada do mundo atualmente e a melhor solução para os requisitos do país. O Ginga é o resultado de vários anos de pesquisas realizadas pela Universidade Católica do Rio de Janeiro (PUC-Rio) e pela Universidade Federal da Paraíba (UFPB).

O sistema é subdividido em dois subsistemas principais interligados (Ginga-NCL e Ginga-J), que permitem o desenvolvimento de aplicações seguindo dois paradigmas de programação diferentes. Dependendo das funcionalidades requeridas no projeto de cada aplicação, um paradigma será mais adequado do que o outro (DTV 2009).

3. Educação a Distância

Existem vários conceitos para educação a distância (EaD), um deles é o da Associação Brasileira de Educação a Distância (ABED, 2006), que conceitua EAD como sendo “a modalidade de educação em que as atividades de ensino-aprendizagem são desenvolvidas

majoritariamente (e em bom número de casos exclusivamente) sem que alunos e professores estejam presentes no mesmo lugar à mesma hora."

Um outro exemplo é o Decreto-Lei Brasileiro nº 5.622, de 20 de dezembro de 2005, onde o termo é conceituado de maneira diferente:

" Art. 1o. – Para os fins deste Decreto, caracteriza-se a educação a distância como modalidade educacional na qual a mediação didático pedagógica no processos de ensino e aprendizagem ocorre com a utilização de meios de tecnologia de informação e comunicação, com estudantes e professores desenvolvendo atividades educativas em lugares ou tempos diversos."

Existem, ainda, outros conceitos, como mostra o pesquisador Nunes (1994):

- Dohmem (1967, apud, NUNES, 1994) diz que a educação a distância é um método de auto-estudo onde o aluno adquire conhecimentos com base no material de estudo que ele tem acesso, sendo o seu desenvolvimento acompanhado e supervisionado através de professores;

- Peters (1973, apud, NUNES, 1994) afirma que educação à distância é uma maneira sistematizada de partilhar conhecimento, habilidades e atitudes através do uso extensivo de meios de comunicação;

- Holmberg (1977, apud, NUNES, 1994) sob o termo EAD ficam encobertos diversos tipos de estudos, em que seus vários níveis não se encontram supervisionados direta e imediatamente por tutores presenciais;

- Perry e Rumble (1987, apud, NUNES, 1994) apresentam a comunicação bidirecional como característica básica para a educação à distância, quando não se encontrarem juntos, no mesmo local, aluno e professor;

A pesquisadora Belloni (2002) considera a educação a distância, como parte de um processo mais amplo de inovação educacional, que se refere à integração das novas Tecnologias de Informação e Comunicação (TIC) nos processos educacionais.

3.1. T-learning

Segundo Bates (2003), T-Learning é o tipo de educação à distância baseado em televisão interativa. Esse termo é também apresentado como a convergência entre a TV digital interativa (TVI) e o E-Learning, sendo mais tarde conhecida como a utilização da tecnologia computacional para apoiar a formação e a realização de atividades educacionais.

Ainda em Bates (2003), o T-Learning dará possibilidade de os telespectadores terem acesso a diversos materiais didáticos, do tipo: filmes, imagens, hipertexto, etc., a partir de suas casas, escolas, local de trabalho ou nos centros comunitários.

4. Pesquisa

A proposta inicial do projeto era instalar o Ginga-NCL em uma máquina virtual rodando uma das distribuição de Linux mais utilizadas, o Ubuntu, para que o programa desenvolvido se comunicasse diretamente com o Ginga-NCL. Após um bom tempo de estudos foram feitas

várias investidas na tentativa se instalar o Ginga-NCL diretamente em uma máquina virtual rodando Ubuntu, a última, e a que chegou-se mais longe, foi abortada, pois houveram vários problemas na compilação dos módulos do Ginga-NCL, muitos dos quais necessitavam de versões mais antigas de bibliotecas e programas do que as disponibilizadas na versão do Ubuntu em uso, o que fez com que a compilação falhasse em diversos estágios, alguns destes problemas foram resolvidos, porem outros necessitariam de mudanças que não estão no escopo do projeto.

Com o conhecimento adquirido durante os estudos, percebeu-se que esta abordagem não era a mais interessante, pois, além da dificuldade de instalação, era muito limitada, uma vez que o usuário da ferramenta teria que rodar tudo de dentro da máquina virtual, e isso traz algumas dificuldades, como:

- O usuário teria que copiar, por linha de comando ou algum outro software, os arquivos da sua máquina host para a máquina virtual, e para um usuário não familiarizado com isto não é algo trivial;
- Para distribuir o sistema implementado neste projeto teria-se que disponibilizar um arquivo .vdi, que é a imagem do disco da máquina virtual, e na ultima tentativa de instalar o Ginga-NCL na máquina Virtual, este arquivo ultrapassou os 7GB de espaço em disco, o que dificultaria a distribuição;
- Dificuldade na atualização do Ginga-NCL, pois para atualizar o player seria necessário compilar o Ginga-NCL, o que exige um conhecimento avançado em Linux.

Por estas razões decidiu-se mudar a abordagem, utilizando então o Virtual Ginga-NCL para rodar os arquivos NCL e criando um software na linguagem Java que será instalado no sistema host do usuário e irá comunicar-se com o Virtual Ginga-NCL através do protocolo SSH para controlar a execução do conteúdo e copiar os arquivos necessários para a máquina virtual. Algumas vantagens da abordagem atual são:

- A Ferramenta é multiplataforma, pois é escrita em Java, e o Virtual Ginga-NCL pode rodar tanto no Linux e no Windows, através do VMWare Player, quanto no Mac OS X, através do VMWare Fusion;
- Maior escalabilidade, pois caso saia uma nova versão do Ginga-NCL, também será disponibilizado uma nova versão do Virtual Ginga-NCL, o que facilita muito a atualização;
- Mais fácil de configurar, pois o usuário terá apenas que configurar o Virtual Ginga-NCL, que é uma tarefa fácil, uma vez que conta com um manual detalhado na página do projeto, e executar o .jar do sistema implementado no projeto.

Ao tentar instalar o KBDE no Virtual Ginga-NCL percebe-se que o source utilizado para a compilação do kernel não foi disponibilizado com a mesma, o que impossibilitou a compilação do simulador de teclado virtual, para tentar contornar este problema foi utilizado o source original do fedora, mas novamente não foi obtido sucesso, pois o source utilizado no Virtual Ginga-NCL foi customizado.

Como não foi possível o uso do Virtual Ginga-NCL, a solução foi utilizar uma máquina virtual disponibilizada no site: http://manoelcampos.com/2010/01/28/virtualbox-vm-ginganclcl-0112-openginga-beta-1/, a máquina roda ubuntu 8.10 e já vem com o Ginga-NCL instalado e roda sobre o VirtualBox. Nesta máquina foi possível compilar e instalar o KBDE,

para que o sistema funcione corretamente, também foram feitas algumas configurações para facilitar a utilização pelo usuário final, como login automático, configuração da interface de rede para acesso por SSH, entre outras.

5. Sistema

O sistema foi implementado na linguagem de programação Java, utilizando o padrão de projeto MVC. Para aumentar o reuso do código, não foram feitas ligações entre o Modelo e a Visão, ou seja, toda a comunicação entre elas é feita através da classe de Controle, além disso, para melhor estruturar o código, foram criadas três classes de controle: MainController, RemoteController e PreferencesController.

Visando facilitar a utilização do sistema pelo usuário, foi adicionada a funcionalidade de iniciar a máquina virtual automaticamente ao iniciar o programa.

Por ter sido desenvolvido na linguagem de programação multiplataforma Java, o software pode ser executados em qualquer sistema operacional que possibilite a instalação da Máquina Virtual Java (Java Virtual Machine - JVM). Como exemplos de sistemas operacionais que apresentam este suporte estão o Microsoft Windows, o Linux, o Mac OS X, dentre outros.

5. Conclusão

A finalidade do trabalho foi desenvolver uma ferramenta para que as pessoas que produzem conteúdo para t-learning possam, de uma maneira fácil, testar o material gerado, tendo em vista que muitas vezes estas pessoas não tem um conhecimento avançado na área de informática.

Durante o desenvolvimento do trabalho, várias dificuldades foram encontradas devido a falta de materiais com informações técnicas sobre o Ginga-NCL, foram feitas várias tentativas de esclarecer essas dúvidas através de emails, fóruns e perguntas em lista de discussões, mas nenhuma delas obteve sucesso. Por este motivo foram feitas várias abordagens para atingir o objetivo deste trabalho.

A primeira abordagem foi a instalação do Ginga-NCL em uma máquina virtual rodando Ubuntu para que o programa que o programa desenvolvido se comunicasse diretamente com o Ginga-NCL. Ela foi abortada, pois houveram vários problemas na compilação dos módulos do Ginga-NCL, alguns resolvidos porem outros necessitariam de mudanças que não estão no escopo deste projeto.

A segunda foi utilizar o Virtual Ginga-NCL para rodar os arquivos NCL e criar um software na linguagem Java que seria instalado no sistema host do usuário e se comunicaria com o Virtual Ginga-NCL através do protocolo SSH para controlar a execução do conteúdo e copiar os arquivos necessários para a máquina virtual, mas ao tentar instalar o KBDE na máquina virtual percebeu-se que o source utilizado para a compilação do kernel não foi disponibilizado com a mesma, o que impossibilitou a compilação do simulador de teclado virtual.

A terceira abordagem e a que obteve sucesso, é basicamente igual a segunda, mas utilizando, no lugar do Virtual Ginga-NCL, uma máquina virtual encontrada na internet já com o Ginga-NCL instalada. Nesta máquina foi possível compilar e instalar o KBDE para fazer a comunicação com o Ginga-NCL.

Mesmo com todos os problemas encontrados, considerou-se que os objetivos estabelecidos para o trabalho foram atingidos, seguindo a metodologia apresentada.

Referências

ABED (2006). Associação Brasileira de Educação a Distância, São Paulo. Disponível em: < http://www.abed.org.br>. Acessado em 16 de Novembro de 2009.

BATES, P. J. (2003). t-learning Study: A study into TV-based interactive learning to the home, Final Report, pjb Associates, UK, 2003.

BELLONI, M. L. (2002). Ensaio sobre a educação a distância no Brasil. Campinas: Revista Educação & Sociedade, ano XXIII, n. 78, abril / 2002. Disponível em: <http://www.scielo.br/pdf/es/v23n78/a08v2378.pdf>.

DTV (2009); Cronograma de implantação da TV digital no Brasil. Disponível em: <http://www.dtv.org.br> Acessado em: 28 de Novembro de 2009.

MONTEZ, C.; BECKER, V. (2005). TV Digital Interativa: conceitos, desafios e perspectivas para o Brasil. 2. ed. Florianópolis: Ed. da UFSC, 2005.

NUNES, I. B. (1994). Noções de educação à distância. Revista de Educação à Distância nrs. 4/5, Dez./93-Abr/1994. Brasília; Instituto Nacional de Educação à Distância.

OLIVEIRA, C. T. de. (2005). Um estudo sobre os padrões de middleware para televisão digital interativa. Fortaleza: CEFET-CE, 2005. Monografia (Tecnólogo em Telemática), Centro Federal de Educação Tecnológica do Ceará. Ceará, 2005.

PNAD (2008); Pesquisa Nacional por Amostra de Domicílios. Disponível em: <http://www.ibge.gov.br/home/estatistica/populacao/trabalhoerendimento/pnad2008/sintesepnad2008.pdf>. Acessado em: 28 de Novembro de 2009.

SANTOS, A. C. O. dos. (2006). A digitalização da TV no Brasil: a sociedade civil organizada e a opinião pública a respeito do sistema brasileiro de TV digital – SBTVD. São Paulo: USP, 2006. Tese (Doutorado em Ciências da Comunicação) Escola de Comunicação e Artes, Universidade de São Paulo. São Paulo, 2006.

SANTOS, D. T. dos. (2007). Estudo de aplicativos de TVDi para educação a Distância. Campinas: UNICAMP, 2007. Dissertação (Mestrado em Engenharia Elétrica) Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas. São Paulo, 2007.