ARQUITETURA E IMPLEMENTAÇÃO DO SISTEMA INTERNET-DTV

C. A. Queiroz alequeiroz@yahoo.com
T. A. Tavares tati@dimap.ufrn.br
V. C. de Paula virginia@dimap.ufrn.br
G. Lemos de Souza guido@dimap.ufrn.br

Grupo NatalNet - Departamento de Informática e Matemática Aplicada - DIMAp
Universidade Federal do Rio Grande do Norte - UFRN
Campus Universitário – Lagoa Nova – 59072-970, Natal - RN

Resumo

Este artigo descreve a modelagem, descrição arquitetural e implementação de um sistema de TV Digital na Internet denominado *Internet-DTV*. Propomos uma solução arquitetural de software capaz de refletir o dinamismo inerente à aplicação, explorando assim, todo o potencial de especificação e abstração de uma ADL (do inglês "***A**rchitecture **D**escription **L**anguage*") e de uma linguagem de programação multi-plataforma. Para tanto, adotamos o Acme para descrever e o Java para implementar o sistema. O artigo apresenta algumas considerações básicas envolvendo aspectos fundamentais sobre TV Digital, bem como, suas principais funções e modelagem; além de abordar os conceitos de arquitetura de software e agentes inteligentes. Finalmente, partimos para especificação do sistema em Acme, e concluímos salientando alguns detalhes de implementação e os resultados obtidos na implantação deste sistema de TV Digital.

Palavras Chave: Arquitetura de Software, TV Digital, Agentes, Dinamismo, *Acme*, Java.

1. Introdução

Mundialmente, o sinal mais utilizado para transmissão dos diferentes tipos de mídia é o sinal analógico. Porém, de acordo com Milenkovic [06], a tecnologia de DTV (do inglês, ***D**igital **Tele**vision*) que se destaca hoje, comercialmente, em sistemas híbridos, os quais utilizam sinais analógicos e digitais, deverá mudar este cenário. Nos Estados Unidos, por exemplo, a transmissão aberta (*broadcast*) e os pontos de origem num sistema de comunicações (*headend*) estão enviando seus sinais digitalmente.

Em um sistema de transmissão aberta de TV Digital, vídeo e áudio são codificados e comprimidos na origem, onde ocorre a captura do fluxo que será transmitido, utilizando o padrão MPEG 2, no perfil principal, nível alto para HDTV [09] desenvolvido pelo grupo MPEG (***M**oving **P**ictures **E**xpert **G**roup*)[06]. A transmissão é feita em diversas opções de redes de transporte: radiofusão, redes de TV a cabo, etc.

Em nossa pesquisa, que se insere no escopo do projeto Internet 2-BR, o sistema de transporte é uma rede de alta velocidade com IP sobre ATM. O fato de termos a nosso dispor uma rede de alta velocidade interligando equipamentos com display e capacidade de processamento, além de uma geradora de TV (TV Universitária da UFRN) nos motivou a investigar e implementar uma aplicação de transmissão de DTV em nossa rede.

A complexidade envolvida no desenvolvimento de um sistema que empregue tal tecnologia, nem sempre é levada em consideração na fase de projeto. Desta forma, os requisitos pretendidos no projeto podem não ser atendidos em sua plenitude, ou até mesmo, esquecidos na implementação. A arquitetura de software se propõe a preencher esta lacuna entre requisitos e implementação, provendo mecanismos que possibilitem ao desenvolvedor considerar requisitos especificados, como também, estruturar de forma modularizada a aplicação [14]. Assim, até a realização de simulações pode se tornar viável mesmo em fase de projeto do sistema, o que torna a especificação da arquitetura um instrumento que além de descrever, pode validar os requisitos do sistema.

Deste modo, utilizamos conceitos de arquitetura de software para descrever as características do sistema *Internet-DTV*, como por exemplo, os protocolos de comunicação, ou até mesmo, o padrão de interface com o usuário que o sistema pode apresentar. Além disso, considerando que esta é uma aplicação complexa e que não há um padrão a ser seguido para especificação de arquiteturas deste tipo, uma importante contribuição deste artigo é a especificação de uma arquitetura complexa e dinâmica, através de uma ADL de referência. Por sua vez, esta especificação voltada ao escopo do Projeto ProTeM-RNP NatalNet, constitui uma importante ferramenta de referência para especificação de arquiteturas análogas.

Adotamos na implementação, por se tratar de um sistema de TV Digital que manipula fluxo de dados multimídia, a API Java JMF (*Java Media FrameWork*) da Sun Microsystems, por esta oferecer suporte aos padrões de mídia digital existentes.

A seguir abordaremos, então, a descrição e implemetação de um sistema de TV Digital, que consiste, em linhas gerais, em um conjunto de equipamentos e de software que permitam ao usuário, por intermédio de um *browser*, receber os sinais de TV digitalizados a partir de uma máquina servidora.

Nossa proposta apresenta a descrição arquitetural do sistema de TV Digital do Projeto ProTeM-RNP NatalNet. Este projeto faz parte dos consórcios de redes metropolitanas de alta velocidade do projeto RNP2, que visa a implantação de uma rede metropolitana de alta velocidade na cidade de Natal.

A principal aplicação da NatalNet será o suporte a difusão de vídeo gerado ao vivo ou sob demanda pela Televisão Universitária - TVU. Dentro deste contexto, se enquadra esta aplicação, cujo objetivo é a implementação de um sistema de TV Digital que disponibilize a programação ao vivo da TVU para os usuários da Rede NatalNet e usuários que acessem o serviço via Internet.

2. Arquitetura de Software e ADLs

A arquitetura de software de um sistema define a estrutura de alto nível, ou seja, denota sua estrutura organizacional bruta como uma coleção de componentes interativos [14].

Com o intuito de termos descrições arquiteturais mais precisas, um grupo de pesquisadores propôs notações formais para representar e analisar arquiteturas de software [01]. Estas notações são conhecidas como Linguagens de Descrição de Arquiteturas, fornecem uma estrutura conceitual e uma sintaxe para caracterizar a arquitetura, procurando atenuar a lacuna existente entre o projeto e a implementação de um software. Além disso, algumas delas possuem ferramentas para execução de atividades como: parsing, unparsing, apresentação, compilação, análise e simulação. Atualmente, existem vários exemplos de ADLs as quais preocupam-se com o design da arquitetura, dentre os principais podemos destacar [04]: Aesop, Adage, Meta-H, C2, Rapide, SADL, UniCon, Wright, etc. Todavia, cada uma delas provê capacidades distintas.

2.1. *Acme* - uma proposta de padronização

A proliferação de propostas incompatíveis de ADLs fez surgir a necessidade de se definir uma linguagem de referência para a descrição de arquiteturas de software, que permita a integração entre especificações de arquitetura escritas em diferentes ADLs. Esta ADL deve ser genérica e compatível com as demais ADLs existentes.

O *Acme* [01] é uma ADL que surgiu a partir de um modelo de referência que supre as necessidades de padronização e trocas entre as ADLs. Hoje, *Acme* é uma ADL que estabelece uma linguagem de troca possibilitando a comunicação entre diferentes ADLs. Seu sucesso apoia-se na necessidade de estabelecer uma linguagem comum e genérica de descrição de arquiteturas.

Em [02] são descritos os elementos básicos para construção de uma arquitetura em *Acme*, são eles: componentes, portas, conectores, papéis (*roles*), sistemas, representações e propriedades.

Além das definições estruturais propostas pela *Acme*, ela também descreve uma notação textual e uma notação gráfica que são utilizadas na descrição de uma arquitetura de software. Estas notações são suportadas pelo ambiente de desenvolvimento de projetos *Acme*: o *AcmeStudio* [03]. O *AcmeStudio* é um visualizador e editor de projetos *Acme* que permite ao arquiteto de software descrever arquiteturas de software através de uma interface própria que suporta visualização em modo texto e gráfico [03].

O *AcmeStudio* provê uma interface gráfica que permite a criação, manipulação e edição de descrições de arquiteturas *Acme*. A interface gráfica implementada na ferramenta segue o padrão Windows o que a torna bastante familiar e amigável à grande maioria dos usuários. Também é oferecido ao usuário a possibilidade de visualizar seu projeto de modos diferentes, um textual e outro gráfico, os quais são sempre atualizados de forma sincronizada.

3. Modelagem do Problema

Uma visão simplificada da aplicação pode ser visualizada na Figura 01, onde o fluxo gerado pela TVU é digitalizado e disponibilizado para os usuários da rede NatalNet ou Internet através de uma máquina servidora.

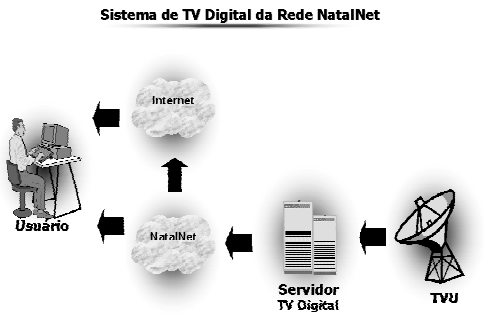

Figura 01 - Sistema de TV Digital da Rede NatalNet.

De fato, uma única máquina servidora é responsável pela captura do sinal da TVU, já a transmissão do fluxo gerado para os clientes não é uma atividade tão elementar. Um ponto chave na transmissão do fluxo é determinar o número de usuários que o sistema deve atender. No entanto, prever este número é uma tarefa difícil já que o sistema encontra-se disponível via Internet. Desta forma o sistema pode passar de um estado de pouca carga para sobrecarga da rede, ou do próprio servidor de TV em um intervalo de tempo muito curto. Basta que se inicie a transmissão de um programa que interesse muito ao público.

Na tentativa de solucionar este problema propomos uma arquitetura de software visando otimizar a utilização tanto da rede quanto do servidor durante a transmissão dos fluxos para os clientes. Procuramos embasar nossa solução em uma outra área de pesquisa da Ciência da Computação, a área de Inteligência Artificial (IA), mais especificamente, no recente campo de Agentes Inteligentes.

Os agentes inteligentes [05,11,12,13] se diferenciam dos softwares em geral por apresentarem como características: mobilidade, autonomia e a habilidade de interação sem a presença do usuário.

No escopo deste trabalho, o emprego de agentes constitui uma importante ferramenta na otimização do oferecimento do serviço. Agentes poderão definir as condições ideais para o bom funcionamento da rede e utilização do servidor. Para tanto, será definida uma arquitetura dinâmica, isto é, uma arquitetura reconfigurável em tempo de execução, na qual um servidor *Principal* será responsável pela captura e digitalização do sinal da TVU e, principalmente, pelo gerenciamento dos servidores *Secundário.* Estes, por sua vez, são diretamente responsáveis pela transmissão do fluxo para os clientes, bem como, pela gerência das conexões com os mesmos. Um servidor *Secundário* pode ser visto como um refletor do servidor *Principal* colocado de modo estratégico na rede para atender da

melhor maneira possível aos usuários. São agentes inteligentes que definem a localização dos servidores *Secundário* na rede. Esta localização é dinâmica, ou seja, de acordo com a necessidade, os servidores podem ser ativados ou desativados, garantindo assim, um melhor oferecimento do serviço.

Um modelo básico do funcionamento deste sistema está descrito nas figuras seguintes através de uma simbologia livre. Os principais módulos apresentados pelo sistema são:

- Software Cliente;
- Servidor *Principal* (composto dos módulos: Gerente de Conexões de Servidores e Captura do Fluxo);
- Servidor *Secundário* (composto dos módulos: Gerente de Conexões de Clientes e Transmissor de Fluxo).

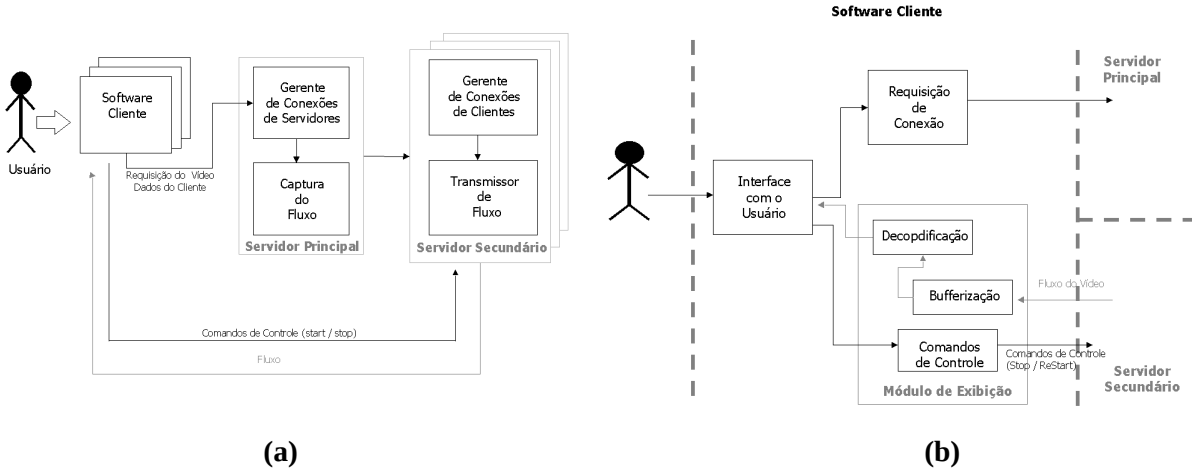

Figura 2 – (a) Modelo Básico do Sistema. (b) Detalhamento do Módulo Software Cliente.

Na Figura 2-a podemos observar o Modelo Básico do Sistema, onde é possível verificar a interconexão dos três principais módulos do sistema. O detalhamento maior do módulo *Software Cliente* pode ser visto na Figura 2-b. O Módulo *Interface com o Usuário* é , responsável pela interação do usuário com o sistema, um dos requisistos para este módulo é que ele se faz implementado através de uma *applet* Java. Deste modo, qualquer usuário da rede poderá utilizar o sistema sem ter a preocupação de instalar um software especial para isso. O processo de *Requisição de Conexão* é feito diretamente para o Servidor *Principal* que encaminha o pedido para o servidor *Secundário* que melhor atende a requisição. A partir deste momento, quem assume a conexão é o servidor *Secundário* que envia o fluxo para o usuário, bem como, responde aos comandos que possam ser a ele enviados (parar ou reiniciar a exibição).

Os módulos de gerenciamento dos servidores *Principal* e *Secundário*, são responsáveis pela administração das conexões com servidores e clientes, respectivamente. No primeiro caso, é utilizada uma estrutura de dados que reflete o estado atual dos servidores *Secundário.* Enquanto no segundo caso, a tabela é um espelho das conexões com os clientes, isto é, uma estrutura de dados altamente dinâmica que contém a identificação dos clientes conectados ao servidor em questão. Inicialmente, esta

identificação será implementada unicamente pelo endereço IP das máquinas cliente, porém, podem ser agregadas outras informações visando o tratamento de questões como confidencialidade e integridade na transmissão do fluxo.

A digitalização do sinal da TVU é uma atividade exclusiva do módulo *Captura de Vídeo,* que nada mais é do que uma placa de digitalização acoplada a uma máquina servidora.

3.1 Descrição da Arquitetura de Software do Sistema *Internet-DTV* em *Acme*

Para especificar o problema em *Acme* utilizamos além, da própria linguagem, a ferramenta *AcmeStudio* [03].

A primeira etapa da descrição arquitetural proposta é definir um conjunto de tipos especiais, pré-definidos para os elementos básicos *Acme*. A definição de tipos nos permite atualizar e alterar características do sistema sem alterar internamente os módulos do sistema, simplesmente modificando suas propriedades, pois é possível alterar, incluir e excluir tipos em *Acme* mesmo após o término da especificação da arquitetura.

Nossa proposta descreve os seguintes tipos para os **componentes:**

InterfaceUIComp – Define um componente responsável pela interface com o usuário.

AgentComp – Define objetos capazes de mover-se de um *host* para outro gerenciando mensagens de acordo com as características do meio.

ProccessComp – Define funções ou procedimentos de software responsáveis pela execução de processos.

ConTableClientsComp – Define uma estrutura de dados dinâmica que armazena e gerencia as conexões de clientes com o servidor de TV.

ConTableServersComp – Define uma estrutura de dados dinâmica responsável pelo mapeamento e gerenciamento dos servidores de TV secundários (*split*).

BufferComp– Define uma estrutura que armazena uma quantidade limitada de dados referentes a segmentos do fluxo de vídeo no cliente.

MasterServerComp – Define o servidor principal de TV, que é responsável pela captura do fluxo gerado ao vivo pela TVU e pelo gerenciamento dos servidores secundários.

SecServerComp – Define um módulo remoto do servidor de TV, que é um repetidor do servidor principal responsável pela transmissão do fluxo gerado no servidor principal para os clientes. Este módulo realiza dois processos principais: o gerenciamento das conexões com os clientes através do armazenamento do endereço IP dos mesmos (Módulo Gerente de Conexões Clientes) e a transmissão do fluxo gerado pelo servidor principal (Módulo Transmissão do Fluxo). Desta forma, é possível ativar várias réplicas do servidor principal na rede de acordo com a demanda de usuários.

Já os **conectores**, podem ser do tipo:

CommunicationConn - Estabelece um elo de comunicação entre cliente e servidor (podendo ser um protocolo de comunicação intranet ou internet).

ActiveSecServerConn – Análogo ao tipo definido anteriormente (*CommunicationConn*), apresentando o diferencial de dinamismo, isto é, o atributo ***optional*** (detalhado posteriormente)**.**

DecodeMPEGConn – Descreve um decodificador MPEG que possibilita a decodificação do fluxo de vídeo transmitido para o modo descomprimido que será exibido para o cliente, este decodificador depende da compressão de vídeo adotada. Neste caso a compressão adotada será MPEG, mas a troca deste conector possibilita a troca no padrão de decodificação.

Contudo, a definição de tipos não é suficiente para especificar o dinamismo desta arquitetura. Desta forma, utilizamos o mecanismo de famílias *Acme* para modelar um tipo especial de servidores. A *SecServerFam* define uma família de componentes do tipo *SecServerComp*. Através da definição da *SecServerFam* é possível especificar um número indefinido de componentes tipo *SecServerComp*, os quais serão ativados ou desativados dinamicamente pelos agentes móveis da arquitetura.

Porém, a ativação ou desativação dinâmica destes componentes implica na definição de duas outras famílias destinadas a ligação dos servidores secundários (*SecServerComp*) com o servidor principal (*MasterServerComp*) que são uma família de portas: *ActiveSecServerFam* e uma de conectores *ActiveSecServerFam*. A primeira descreve um conjunto de portas que habilitam o funcionamento de um servidor secundário, isto é, uma família de portas do tipo *ActiveSecServerPort.* No segundo caso, é uma família de conectores do tipo *ActiveSecServerConn* que é um tipo análogo ao tipo *CommunicationConn,* com um fator diferencial dinâmico. Este fator é definido através do atributo especial ***optional*** definido pelo *Acme*, o qual permite que elementos estejam ora presentes e ora ausentes na arquitetura. Assim, para a especificação de uma *ActiveSecServerPort* teremos:

```
Component MasterServer = {
...
        port ReceiveFromClient
        port Send2Sec
        port AgentPort
        optional port ActiveSecServer
...
            };
```

A utilização do *AcmeStudio* impõe uma abordagem de mais alto nível na especificação da arquitetura, já que a linguagem *Acme* define inicialmente uma configuração *toplevel* do sistema, isto é, a visão global do mesmo e, posteriormente, são detalhados os demais componentes compondo a estrutura hierárquica do sistema. Nesta descrição arquitetural, utilizamos um "supercomponente" *DTV_System,* o qual define além da configuração *toplevel* do sistema, um componente do tipo *AgentComp*. Desta forma, é possível especificar, ainda em nível arquitetural, estruturas como agentes. Estas estruturas, apresentam propriedades específicas que definem métodos para criação e gerenciamento de mensagens capazes, por exemplo, de ativar ou desativar, módulos internos do sistema (como um servidor secundário).

A Figura 3, denota o mais alto nível da aplicação onde está especificada a modelagem genérica do sistema de TV Digital. O sistema é representado a partir de quatro componentes (módulos) principais: MasterServer, SecServerComp, ClientSoftComp e o DTV_System. Este último, representa estruturalmente os agentes que envolvem a aplicação. Os componentes encontram-se interligados através de conectores específicos que estabelecem os protocolos necessários para a comunicação. Além disso, são definidas portas e papéis (*roles*) que descrevem a comunicação com o conector e estabelecem regras para a comunicação. Na Figura 3, ainda é possível visualizar a representação gráfica da estrutura em *Acme,* desta forma os componentes são representados pelas caixas retangulares, os conectores pelos círculos maiores e os papéis pelos círculos menores, mostrados na junção entre conectores e componentes e as portas pelo quadrados.

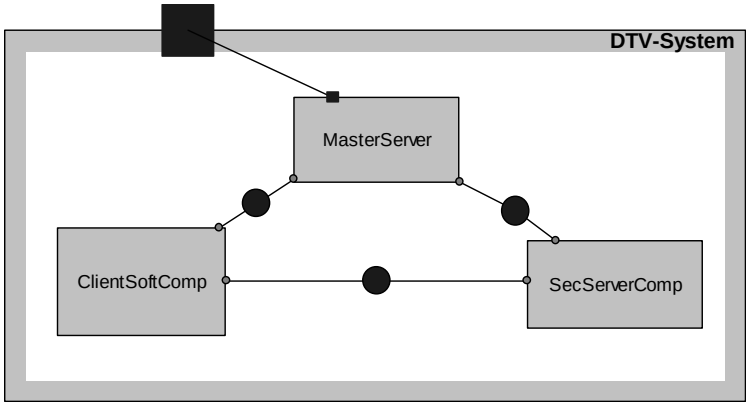

Figura 3 - Diagrama *Acme* para Visão Geral da Aplicação.

O componente ClientSoftComp (Figura 4-a) é responsável pela interação com o usuário, isto é, constitui o meio pelo qual o usuário tem acesso às funções do sistema. Para tanto, este componente é refinado em outros dois componentes: UserInterface e Buffering; e em um conector que é o DecodeConn. Nesta figura, são visíveis graficamente as portas que são apresentadas como quadrados pequenos ligados aos componentes e as pontes, ou *Bindings*, que são os quadrados maiores, as quais estabelecem a ligação entre os diferentes níveis de representação do sistema em *Acme*.

O servidor principal de TV é detalhado na Figura 4-b, onde podemos observar os componentes: *StreamingCapComp, ConnectionTableComp* e *ConnectionManagerComp,* bem como suas interligações. Nota-se que o componente *StreamingCapComp* comunica-se, unicamente, com elementos de níveis estruturais externos. Isto se deve ao fato de que no servidor principal acontecerá apenas a captura do fluxo gerado pela TVU e a transmissão ficará a cargo dos servidores secundários. Em nível de implementação, os servidores secundários farão o acesso remoto (RMI – ***R****emote* ***M****ethod* ***I****nvocation*) à classe responsável pela captura do fluxo do servidor principal.

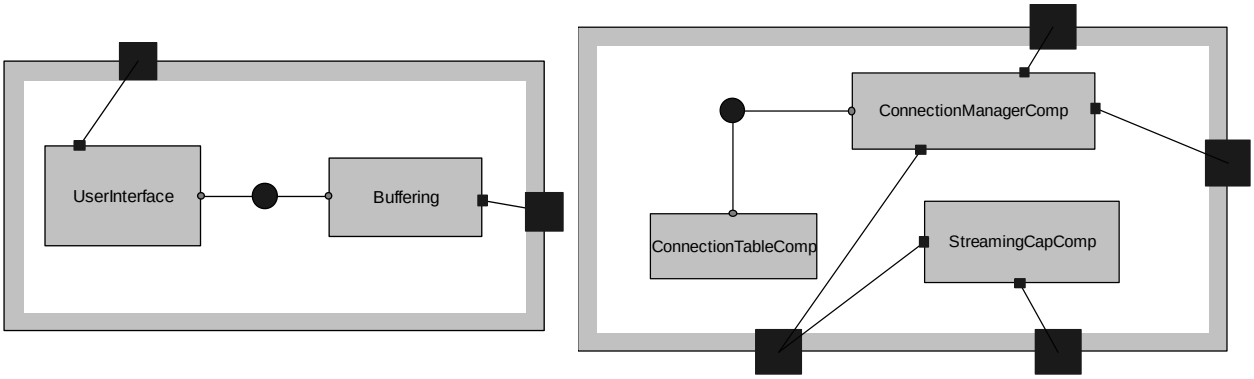

Figura 4 – (a) Detalhamento do Módulo ClientSoftComp. (b) Detalhamento do Módulo MasterServer.

O componente *ConnectionManagerComp* pode ser visto como o gerente deste módulo da aplicação, pois ele recebe as solicitações do cliente e as encaminha para o servidor secundário mais conveniente no momento do pedido. Para tomar tal decisão, este componente permanece em contato constante com o sistema de agentes.

Outro ponto importante é a definição do componente *ConnectionTableComp,* o qual descreve, como visto anteriormente, uma estrutura de dados dinâmica, que contém os dados referentes as conexões com os servidores secundários. Para definição dessa estrutura é necessário, essencialmente, a localização e o número máximo de clientes suportados pelo servidor remoto e o status das condições atualizadas do mesmo. Analogamente, cada servidor secundário, também possui uma estrutura similar que gerencia as conexões com os clientes. Para tanto, é preciso, essencialmente, do endereço IP do cliente, porém, para dar suporte a aspectos como segurança e autoria podem ser agregados a essa estrutura outros dados como: máquina login e senha do usuário, no caso de autenticação e marca d'água de proprietário, para garantir direitos autorais.

Outro aspecto interessante é tratado no componente que controla os comandos enviados pelos usuários. Este componente interpreta e executa os controles sobre o fluxo de vídeo durante a apresentação para o usuário. Tratando-se de DTV, estes comandos são basicamente: *start* e *stop.* Todavia, se considerarmos a crescente tendência da TV interativa que aumenta as possibilidades de controle do usuário, o acréscimo de comandos a este componente significará agregar interatividade à transmissão de fluxo. Essa interação representa muito mais do que uma simples mudança de canal, ajuste de volume, ou até mesmo a escolha de um filme ou do seu desfecho, ela deve acrescentar mais controle, mais conveniência e mais diversão para os usuários.

4. Implementação do Sistema *Internet-DTV*

O componente *InterfaceUIComp* definido na especificação, é implementado através de um *applet* Java, o qual é responsável por receber e apresentar para o usuário o fluxo transmitido pelo servidor. Para tanto, o *applet* segue sua seqüência de métodos: *init( )*, *start( )*, *stop( )* e *destroy( ).* A classe *player* é mais importante da interface cliente, pois tem por função a decodificação e exibição do

fluxo transmitido pelo servidor. A Figura 6, mostra o *layout* do software cliente deste sistema de TV Digital.

Figura 6 - *Layout* do Cliente.

O servidor principal (componente ***MasterServer)*** se comunica com o dispositivo de captura do sinal (placa de captura de vídeo) para receber o fluxo dos dados e processá-los através da classe *DataSource*. Deste modo, é possível disponibilizar o fluxo para os servidores secundários através da invocação remota da classe *DataSource*, e estes, por sua vez, transmitem o fluxo para os clientes.

Para o servidor secundário (componente *SecServerComp*) aceitar múltiplos clientes é necessário que ele possua um *DataSource* para cada cliente, e ainda, *DataSources* distintos para transmissão de áudio e vídeo. Para solucionar esse problema, o JMF oferece o *CloneableDataSource*, que permite criar clones de *DataSources*, podendo-se então ter uma grande quantidade de clientes por máquina. Existem vários tipos de *DataSources,* em nosso servidor utilizamos o *PushDataSource*, utilizado quando se usa o protocolo RTP para transmissão do fluxo. Abaixo segue o trecho de código implementado correspondente.

```
DataSource origem = null;   // Cria variável origem do tipo Datasource
            try {   origem = processor.getDataOutput();  coloca o conteúdo no
Datasource
            } catch (NotRealizedError e)  // tratamento de erros
               System.exit(-1);
            }
            DataSource cloneableDataSource = null;    //inicialização dos datasources
clonáveis
            DataSource clonedDataSource = null;
            cloneableDataSource
              = Manager.createCloneableDataSource(origDataSource); // criação dos
clones
               clonedDataSource
```

5. Conclusões e Perspectivas Futuras

Este artigo apresentou a descrição da arquitetura e de implementação do sistema *Internet-DTV*.

Uma importante contribuição do artigo é a arquitetura descrita formalmente em *Acme* que serve como referência para outras implementações que venham a ser feitas desta classe de sistemas, cabendo destacar o aspecto dinâmico definido na arquitetura no que tange à ativação e desativação de servidores secundários e redistribuição dos clientes entre os servidores ativos.

Outra contribuição é a implementação do *Internet-DTV* que nos permitiu validar a arquitetura proposta. O grande diferencial do *Internet-DTV* em relação a sistema correlatos [15,16,17,18,19], é ter sido concebido com base em uma arquitetura dinâmica capaz de se ajustar rapidamente à variação na carga do sistema, tanto em termos de quantidade quanto de localização dos clientes. O aspecto dinâmico também está presente nos clientes que são capazes de receber, decodificar e apresentar fluxos codificados com formatos diferentes. Na versão atual do cliente ele manipula simultaneamente, MPEG-2, MPEG-1 e RA. Isto é muito importante em redes onde a QoS (Qualidade de Serviço) varia muito, como o caso da Internet, pois permite ao cliente oferecer ao usuário o vídeo com a melhor qualidade possível de ser transportado na rede. Por exemplo, em um dado instante, quando a banda passante é pequena o cliente apresenta vídeo RA (160Kbps) passando para MPEG-1 (1,5Mbps) e MPEG-2 (6Mbps) quando a banda disponível permitir.

Na versão corrente do *Internet-DTV,* o cliente e o servidor principal estão implementados. Estamos trabalhando na implementação dos servidores secundários e em técnicas para estabelecer a configuração ótima da aplicação utilizando, para tanto, algoritmos genéticos.

6. Referências Bibliográficas

[01] GARLAN, David; MONROE, Robert & WILE, David. **Acme: an Architecture Interchange Language**. Janeiro-1997.

[02] GARLAN, David, MONROE, Robert & WILE, David. **Acme Overview Version 0.1.1**. Novembro-1997. http://www.cs.cmu.edu/~acme

[03] KOMPANEK, Andrew. **AcmeStudio - User's Manual Copyright** ©1998. School of Computer Science. Carnegie Mellon University - 1998. http://www.cs.cmu.edu/~acme

[04] MEDVIDOVIC, Neno. **A classification and Comparison Framework for Software Architecture Description Language.** Departament of Information and Computer Science, University of California, irvine, USA. VCI-ICS-97-02. Feb-1997.

[05] COSTA, Marcello Thiry Comicholi. **Uma Arquitetura Baseada em Agentes para Suporte ao Ensino à Distância.** Florianópolis. Universidade Federal de Santa Catarina, abril de 1999. (Tese de Doutorado).

[06] MILENKOVIC, Milan. **Delivering Interactive Services via a Digital TV Infrastructure.** IEEE Multimedia 1998. 34-43p.

[07] de CARMO, Linden. **Core Java Framework**.

[08] ARIDOR, Yariv & LANGE, Danny B. **Agent Design Patterns: Elements of Agents Application Design.** Proceedings of 2nd International Conference on Autonomous Agents. (1998)

[09] CECILIO, Edmundo L. & RODRIGUES, Rogério F**. Televisão de Alta Definição (HDTV).** Relatório Técnico, Laboratório Telemídia, Depto. de Informática, PUC-Rio. 1996.

[10] BIRKMAIER, Craig. **The future of digital television**. Setembro-1998 http://www.digitaltelevision.com/future.shtml

[11] HENDLER, J. **Intelligent Agents: Where AI Meets Information Technology**. IEEE Expert Systems, Dezembro-1996, 20-23 p.

[12] SPECTOR, L. **Automatic Generation of Intelligent Agent Programs**. IEEE Expert Intelligent Systems, Fevereiro-1997, 3-4 p.

[13] KNAPIK, M. e JOHNSON, J. **Developing Inteligent Agents for Distributed Systems**. Computing McGraw-Hill, NY:McGraw-Hill, 1998.

[14] SHAW, Mary e GARLAN, David. **Software Architecture: Perspectives on an Emerging Discipline**. Prentice Hall. 1996.

[15] **Weckauf DTV GmbH** - http://www.weckauf.de/

[16] **WebTV** - http://www.webtv.com

[17] **RealServer** – http://www.real.com

[18] **IP/TV** – http://www.cisco.com/iptv

[19] **Burstware** – http://www.burst.com