Uma Arquitetura para Configuração e Gerenciamento de Recursos em um Middleware para Sistemas de Televisão Digital Interativa

Adilson Barboza Lopes, Frederico Borelli, Glêdson Elias,
Guido Lemos S. Filho, Maurício F. Magalhães

***Resumo*—Sistemas de televisão digital interativa devem incorporar conceitos de middleware para compatibilizar e abstrair especificidades de hardware e sistemas operacionais. Neste contexto, este artigo descreve uma arquitetura extensível para configuração e gerenciamento de recursos e componentes de um middleware proposto para sistemas de televisão digital interativa.**

***Palavras-Chave*—middleware, televisão digital interativa, componentes de software.**

***Abstract*—Interactive digital television systems should provide concepts of middleware for harmonizing and abstracting discrepancies related to hardware and operating systems issues. In such a context, this paper describes an extensible architecture for specification, configuration and management of resources and components of a proposed middleware.**

***Index Terms*—middleware, interactive digital television, software components.**

I. INTRODUÇÃO

Televisão digital interativa (TVDI) tem se tornado uma tecnologia cada vez mais relevante no cenário mundial. Esta tecnologia permite a utilização de aparelhos de televisão como computadores dedicados à tarefa de executar programas interativos, abrindo perspectivas bastante interessantes para o mercado de desenvolvimento de software. Embora TVDI seja uma área que ainda se encontra em fase de maturação, diversas tecnologias de hardware [1, 2] e software [3, 4] já estão disponíveis. No entanto, ainda discute-se padronização dos dispositivos, sistemas operacionais, ambientes de execução, linguagens, serviços e APIs.

Adilson Barboza Lopes, Frederico Borelli e Glêdson Elias, Departamento de Informática e Matemática Aplicada, UFRN, Brasil, E-Mails: adilson@dimap.ufrn.br, fred@natalnet.br, gledson@dimap.ufrn.br. Guido Lemos S. Filho, Departamento de Informática, UFPB, Brasil, E-mail: guido@di.ufpb.br. Maurício Ferreira Magalhães, Departamento de Engenharia da Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Brasil, E-mail: mauricio@dca.fee.unicamp.br .

Uma aplicação para TV interativa pode manipular diversas mídias com requisitos temporais críticos, tais como: áudio e vídeo. Além disso, a complexidade destas aplicações é incrementada, uma vez que poderão ser executadas em diversos dispositivos, com capacidades variadas de processamento e armazenamento, e diferentes interfaces com o usuário.

Neste contexto, este trabalho busca soluções que permitam a construção de plataformas e ambientes flexíveis e abertos. Como resultados iniciais deste esforço destacam-se duas propostas preliminares: o *framework* Cosmos – um *framework* para configuração e gerenciamento de sistemas multimídia distribuídos abertos [5], e o *middleware* AdapTV – um middleware adaptativo, baseado no Cosmos, para Sistemas de Televisão Digital Interativa [6, 7]. O presente artigo apresenta uma visão integrada e estendida destas propostas, detalhando e descrevendo um modelo de especificação para configuração e gerenciamento de recursos, com o objetivo de viabilizar a implementação de um componente configurador para o middleware AdapTV.

Este artigo está organizado da seguinte forma. Na Seção II, alguns conceitos básicos e um modelo de funcionamento para sistemas de TV interativa são introduzidos. A Seção III faz uma análise comparativa dos vários trabalhos relacionados. A seção IV introduz o *Framework* Cosmos, enquanto que a Seção V apresenta o *Middleware* AdapTV. A Seção VI descreve e discute uma aplicação exemplo, ilustrando a especificação de componentes e composição da aplicação. Por fim, a Seção VII apresenta as considerações finais.

II. SISTEMAS DE TELEVISÃO DIGITAL INTERATIVA

Atualmente, a indústria está desenvolvendo diversos tipos de dispositivos domésticos, *set-top box* ou televisão digital integrada, capazes de receber e processar fluxos digitais que multiplexam canais de vídeo, áudio e dados. Por sua vez, o canal de dados pode encapsular software a ser executado nestes dispositivos, provendo assim a possibilidade de interação direta ou indireta com os usuários. Em [6], sistemas de TVDI são representados usando uma arquitetura de quatro camadas: hardware, sistema operacional, middleware e aplicações.

Para o desenvolvimento de sistemas de televisão interativa, deve-se levar em consideração o nível de interatividade que a

infraestrutura de transmissão proverá aos seus usuários. Este nível de interatividade pode ser classificado em duas categorias: sistemas sem canal de retorno e sistemas com canal de retorno [6].

A Figura 1 apresenta o papel do middleware em um sistema de televisão digital com canal de retorno. O cenário descrito é baseado na arquitetura cliente/servidor. No cliente, o middleware lida com a recuperação, especificação, gerenciamento, apresentação e execução de fluxos de dados multimídia e componentes de software. No servidor, fluxos (*streams*) elementares de vídeo, áudio e dados são multiplexados para compor um único fluxo de transporte. Este fluxo é transmitido em um canal de *broadcast* codificado no formato MPEG2-TS [8].

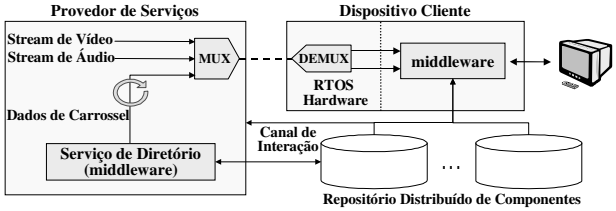

Fig. 1. Estrutura de um sistema de televisão interativa.

III. TRABALHOS RELACIONADOS

O conceito de middleware tem sido explorado recentemente em várias propostas. Nesse contexto, o *framework A/V Streams* da OMG [9] define componentes e serviços com uma abordagem arquitetural para implementação de aplicações envolvendo fluxos multimídia. Nesta mesma linha, a ISO também propõe o *framework* PREMO [10]. Vale a pena destacar a grande importância que o PREMO desempenhou na concepção de algumas propostas mais recentes, como, por exemplo, a *A/V Streams* da OMG [9]. Entretanto, o *framework* PREMO apresenta um alto nível de complexidade devido à abrangência de sua proposição, que tenta considerar um conjunto bastante extensivo de requisitos para aplicações multimídia distribuídas. Provavelmente, em conseqüência dessa complexidade, não é comum encontrar referências para implementações totalmente aderentes ao modelo PREMO. No entanto, o conceito de propriedades proposto no PREMO é muito poderoso, em particular para dar suporte à configuração, reconfiguração, reflexividade e gerenciamento de recursos.

Sob a perspectiva de alocação de recursos e adaptação dinâmica, Cosmos foi influenciado por [11] e [12]. Ambas propostas apresentam visões arquiteturais de *frameworks* para gerenciamento de recursos, com hierarquias distintas especificadas para recursos e tarefas.

Em contraste com [12], Cosmos emprega uma estratégia de gerenciamento de configuração em vários estágios, nos quais fábricas devem observar apenas se a plataforma local tem capacidade para suportar os requisitos mínimos requeridos pelo componente. A alocação de recursos ocorre somente quando todos os componentes da aplicação tiverem sido instanciados. Esta estratégia adota um modelo para alocação baseada em grupos, facilitando a definição de políticas de alocação globais, por exemplo, para encapsular requisitos de QoS (Qualidade de Serviço) no nível arquitetural, ao invés de tratar cada alocação individualmente.

O AdapTV propõe um middleware para Sistemas de TVDI baseado no Cosmos. Como característica importante para dar suporte a configuração e gerenciamento de recursos, destaca-se o uso do conceito de propriedades. Este conceito está sendo explorado no modelo de especificação definido. O estágio atual de desenvolvimento do middleware AdapTV acena para uma perspectiva de viabilidade do mesmo.

IV. O FRAMEWORK COSMOS

O *framework* Cosmos tem como objetivo atuar como um *framework* genérico para projeto e desenvolvimento da camada *middleware* para uma variedade de sistemas distribuídos multimídia. De modo a atender este propósito, o *framework* define componentes abstratos, que fornecem uma visão da arquitetura do sistema, abstraindo os detalhes de implementação desses componentes.

A. Características Básicas

O Cosmos concentra seus esforços na representação e manipulação da diversidade de conceitos, requisitos e componentes relacionados com os sistemas multimídia. Componentes Cosmos provêem APIs (*Application Programming Interfaces*) que abstraem detalhes e características específicas de recursos, sistemas operacionais e de tecnologia de comunicação. Estas APIs dão suporte ao controle, gerenciamento, distribuição e apresentação de dados multimídia.

Para lidar com a diversidade de requisitos, o *framework* incorpora vários conceitos que estão distribuídos em diferentes hierarquias: interfaces, componentes, recursos e serviços. Nos níveis inferiores das hierarquias, encontram-se os componentes básicos. Estes componentes provêem um conjunto de interfaces que são herdadas pelos componentes dos níveis superiores da hierarquia. Componentes básicos foram projetados para dar suporte ao tratamento de eventos, sincronização e propriedades.

As interfaces de propriedades dão suporte às tarefas de configuração, gerenciamento e adaptação. No Cosmos, elas são caracterizadas como meta-interfaces, usadas para descrever os aspectos não-funcionais dos componentes como, por exemplo, QoS, políticas de gerenciamento, critérios de busca e localização de componentes. Gerentes de configuração realizam consultas e definições de valores de propriedades para coordenar as várias fases do ciclo de vida e para estabelecer negociações e acordos envolvendo, por exemplo, a definição de parâmetros de QoS na alocação de recursos.

A seguir, as principais características do *framework* Cosmos são destacadas:

- Definição de componentes para processamento de mídias. Implementam o conceito lógico de recursos virtuais (*VirtualResources*) envolvendo características reais de dispositivos de hardware e software.

- Definição do conceito de portas. Portas são usadas para conectar fluxos de objetos de mídia ou de controle entre componentes, através de componentes denominados *VirtualConnections*. O papel de um componente *VirtualConnection* é abstrair o tratamento de fluxos, isolando-os para a aplicação, ao nível de componente e de porta. Conexões podem ser realizadas entre componentes locais ou remotos, cabendo ao middleware que implementa a conexão definir o tipo de canal mais adequado para realizar a comunicação.
- Definição de mecanismos para tratar fluxos de dados baseado em tratadores de eventos.
- Composição de múltiplos recursos virtuais e conexões em uma única unidade para propósito de gerenciamento de recursos e controle de fluxo através da definição do conceito de grupo (recurso lógico).
- API para controle de progressão de fluxos, sincronização e interação.
- Separação de interfaces para configuração de componentes (meta-interface) e para controle (interface operacional).

B. *Framework Arquitetural de Alto-Nível*

Para dar suporte à configuração e gerenciamento de recursos, Cosmos define um *framework* arquitetural de alto-nível, introduzido na Figura 2.

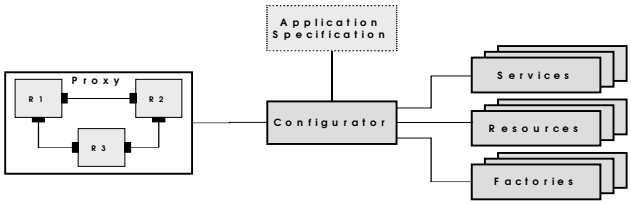

Fig. 2. Framework arquitetural de alto-nível do Cosmos.

Os principais componentes que dão suporte aos serviços de configuração e gerenciamento de ciclo de vida envolvem elementos como:

- Configurator: inicia, configura e gerencia recursos e componentes;
- Proxy: representação interna de uma aplicação em execução;
- Fábricas: responsáveis por criar componentes;
- Serviços: torna possível a incorporação de novas funcionalidades.
- Recursos: mecanismos para representação e controle de dispositivos lógicos e físicos da plataforma.

Como componente central do *framework*, o *configurator* é o componente crítico responsável por iniciar, configurar e gerenciar os recursos e componentes do sistema (camadas de *middleware* e aplicação). Cada aplicação é definida através de uma especificação envolvendo os componentes e os elementos da arquitetura, bem como os vários requisitos de QoS necessários à sua execução.

O *configurator* é também responsável pelo gerenciamento do ciclo de vida dos componentes. Ele realiza operações de inspeção, definições de configuração, negociações e ajustes dinâmicos de propriedades, que estão associados com os componentes e fluxos multimídia envolvidos no sistema.

C. *Configuração e Gerenciamento de Recursos*

O processo de configuração de recursos virtuais é realizado em várias etapas, que são modeladas internamente nos componentes como um conjunto de possíveis estados. A Figura 3 ilustra apenas os estados relacionados com a fase de instanciação de componentes e negociação de propriedades. Estes estados são suportados no atual estágio de desenvolvimento do AdapTV, e tratados pelo modelo de especificação apresentado na seção V.

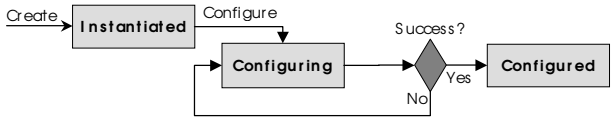

Fig. 3. Processo de Configuração de Recursos Virtuais.

Para cada componente indicado na especificação da aplicação, o *configurator* verifica se existe coerência entre os recursos requisitados, os requisitos de QoS indicados e as características da plataforma, e se, eventualmente, poderão ser atendidos na etapa de alocação. Requisitos de QoS são especificados também através de valores de propriedades. Se a verificação for satisfatória, uma fábrica realiza a instanciação. Porém, neste instante, o recurso ainda não será alocado. Somente quando todos os componentes forem instanciados e as conexões entre portas processadas – observando, naturalmente, se existe coerência entre as características das portas envolvidas - é que o processo de alocação de recursos é iniciado.

Por limitação de espaço, os demais estados, relacionados com as fases de alocação de recursos e execução da aplicação, não são apresentados neste artigo.

V. O MIDDLEWARE ADAPTV

Sistemas de Televisão Digital Interativa (TVDI), de forma semelhante aos sistemas multimídia tradicionais, envolvem uma diversidade de requisitos e características. Em adição, existe uma heterogeneidade de plataformas de suporte, com vários tipos de dispositivos, recursos de hardware e software, bem como usuários com diferentes necessidades. Conseqüentemente, para lidar com esta diversidade de situações, sistemas de TVDI devem prover uma infra-estrutura altamente adaptativa. A arquitetura do *middleware* proposto leva em consideração alguns requisitos que foram enumerados em [6] para middleware de suporte a aplicações TVDI, como por exemplo: confiabilidade, segurança, portabilidade, sincronização, extensibilidade e reflexividade.

A. *Arquitetura em Camadas*

A arquitetura do *middleware* está organizada em duas camadas: camada de configuração e gerenciamento, e camada de manipulação de mídia. Estas camadas provêem APIs

especializadas para a camada de aplicações, conforme observado na Figura 4:

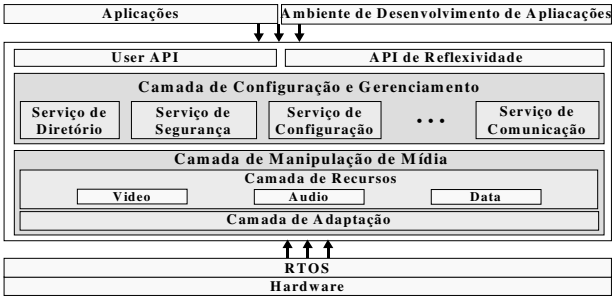

Fig. 4. Arquitetura em camadas do AdapTV

A arquitetura do AdapTV foi definida com base nas propostas apresentadas anteriormente em [6, 7]. Na camada mais alta do *middleware*, encontram-se duas interfaces de programação (APIs): usuário e reflexividade. Estas APIs fornecem a base para a construção de ambientes de desenvolvimento e de aplicações interativas. O uso destas APIs permite acessar os recursos de hardware e do sistema operacional de maneira transparente às suas especificidades.

A API do usuário provê um conjunto de operações básicas que fornecem as funcionalidades tradicionais do *middleware*, como, por exemplo, acesso à rede, sintonia de canais, e localização de componentes na rede. Por outro lado, a API de reflexividade disponibiliza um conjunto de operações para obtenção de informações sobre o ambiente. Através desta API, as aplicações e o próprio *middleware* podem modificar a configuração do ambiente. É importante observar que a reflexividade pode ser explorada de duas formas complementares:

- **Reativa:** aplicações usam a API de reflexividade para obter meta-informações dos componentes e eventualmente para realizar adaptações estruturais.
- **Pró-ativa**: componentes de monitoramento do middleware reconhecem a necessidade de adaptação da aplicação, por exemplo quando um valor limite de QoS for atingido, de forma a requisitar automaticamente ao middleware para realizar a adaptação.

A camada de configuração e gerenciamento engloba várias funcionalidades, incorporadas em módulos típicos, como os indicados a seguir:

- **Serviço de Configuração**: responsável por configurar, negociar, alocar e gerenciar componentes e recursos virtuais;
- **Serviço de Diretório**: recuperação de componentes requeridos de forma transparente;
- **Serviço de Segurança**: mecanismos para autenticação e controle de acesso;
- **Serviço de Transações**: provê suporte transacional para aplicações de comércio eletrônico;
- **Serviço de Comunicação**: suporte ao desenvolvimento de aplicações distribuídas;

Deve-se observar que a camada de configuração e gerenciamento não tem acesso direto aos dispositivos e aos recursos do sistema operacional. Acessos a estes componentes são feitos através da camada de manipulação de mídia. Para isso, esta camada define recursos virtuais especializados para tratar cada tipo básico de fluxo (vídeo, áudio e dados), bem como para os demais aspectos de baixo-nível da plataforma.

Devido à existência de diferentes sistemas operacionais e a variedade de tipos de dispositivos, parte da implementação de um dispositivo virtual torna-se específica para cada plataforma ou dispositivo. Assim, estas partes de implementação apresentam baixo nível de portabilidade. Para amenizar o impacto desta restrição decidiu-se subdividir a camada de manipulação de mídia em duas subcamadas:

- **Camada de Recursos**: consiste dos recursos virtuais que definem APIs uniformes para os componentes de níveis mais altos, ao passo que assume a existência de APIs uniformes para acessar a camada de adaptação;
- **Camada de Adaptação**: composta de componentes que definem uma API uniforme para os componentes específicos da camada de recursos; estes componentes isolam os aspectos de implementação específicos, incorporando a parte de código dependente de cada tipo de plataforma e de dispositivo.

Uma abordagem mais abrangente da arquitetura definida para o *middleware* AdapTV, onde é apresentada uma visão integrada envolvendo os principais componentes das diferentes camadas, pode ser vista na Figura 5. Pode-se observar que a camada de gerenciamento corresponde ao modelo arquitetural apresentado na seção IV e proposto pelo *framework* Cosmos, incluindo módulos como: *configurator*, *proxy*, *resources*, *factories* e *services*. No middleware, a maioria desses módulos é implementada como especializações de *VirtualResources*, *Factories* e *Services*.

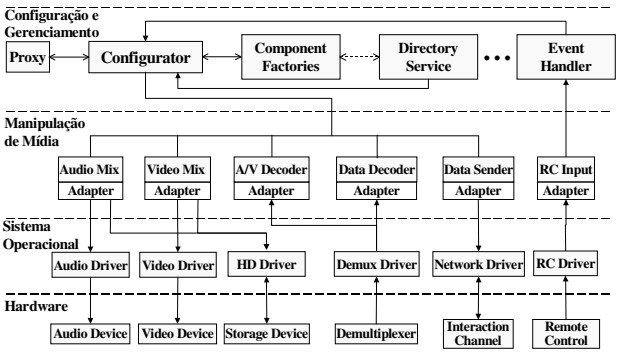

Fig. 5. Arquitetura detalhada do AdapTV

B. Modelo de Especificação de Componentes

No *middleware* AdapTV, uma aplicação é especificada e tratada explorando o conceito de composição. A seguir, apresentamos o modelo de especificação de metadados de componentes e de estratégias de composição suportadas pelo *middleware*. Explorando tais metadados, um ambiente baseado no *middleware* proposto permite a composição das aplicações e componentes.

Para especificar uma aplicação, o projetista precisa descrever os componentes, as propriedades de suas portas e as respectivas conexões. O *middleware* AdapTV adota um

modelo de especificação baseado em XML [13]. Esta decisão deve-se ao fato de existir uma gama de manipuladores e ferramentas XML, além de ser uma tecnologia adotada em importantes padrões para televisão digital como, por exemplo, MHP [14], DASE [15] e ARIB[16]. Esta especificação foi formalizada através de um modelo baseado em XML Schemas. Na Figura 6 a seguir, os blocos principais do esquema proposto são brevemente apresentados, onde são descritos os blocos *components*, *ports* e *connections*.

```
<xs:element name="component">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="attributes"/>
      <xs:element ref="depends"/>
      <xs:element ref="properties"/>
    </xs:sequence>
  </xs:complexType>
  <xs:attribute ref="name"/>
  <xs:attribute ref="description"/>
</xs:element>
```

```
<xs:element ref="port">
  <xs:complexType>
    <xs:element ref="property"
        maxOccurs="unbounded"/>
  </xs:complexType>
  <xs:attribute ref="name"
      use="required"/>
</xs:element>
<xs:element name="ports">
  <xs:complexType>
    <xs:element ref="port"
        maxOccurs="unbounded"/>
  </xs:complexType>
</xs:element>
```

```
<xs:simpleType name="location">
  <xs:restriction base="xs:string">
    <xs:pattern value=
        "((a-z)|(0-9))*:((a-z)|(0-9))*"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="from"
        type="location"/>
<xs:element name="to"
        type="location"/>
<xs:element name="connect">
  <xs:complexType>
    <xs:attribute ref="from"
        use="required"/>
    <xs:attribute ref="to"
        use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="connections">
  <complexType>
    <xs:element ref="connect"
        maxOccurs="unbounded"/>
  </complexType>
</xs:element>
```

Fig. 6. Blocos component, ports e connections

Dessa forma, a definição de um componente (tag *component* root) pode ter os seguintes elementos: *attributes* – indicam características funcionais; *depends* – indicam outros componentes requisitados; *properties* – expressam requisitos não-funcionais; *ports* – definem as portas do componente; e *connections* – estabelecem conexões entre portas. O bloco *ports* define a sintaxe para a descrição de portas. Cada porta (*port*) é caracterizada por um identificador (*name*) e por um conjunto de propriedades (*property*). Estas propriedades identificam as características associadas à porta, como por exemplo: tamanho do buffer, algoritmo de codificação, taxa de transmissão (ver exemplo da Figura 7). A descrição de uma conexão (*connect*) identifica os pontos origem do fluxo (*from*) e destino do fluxo (*to*); cada um desses pontos é indicado através de uma sintaxe envolvendo os pares *component* e *port* separados pelo caracter dois pontos (:).

VI. ESPECIFICAÇÃO DE UMA APLICAÇÃO EXEMPLO

Para exemplificar o processo de desenvolvimento, a Figura 7 mostra um caso de uso proposto e descrito no modelo XML definido neste artigo, tomando como exemplo a aplicação torcida virtual apresentada em [17]. Para cada componente deve-se disponibilizar sua especificação XML. Esta será tratada pelo *configurator* para realizar operações tais como: busca de componentes da hierarquia de dependência estabelecida; instanciação dos mesmos através das respectivas *factories*; e negociação para garantir os requisitos especificados, além de realizar as conexões necessárias entre as portas de fluxo.

```
<COMPONENT name="VirtualCheering"
  description="Serviço virtual cheering.">
  <ATTRIBUTES>
    <ATTRIBUTE name="Chairs" default="50"/>
    <ATTRIBUTE name="GameDescription"/>
  </ATTRIBUTES>
  <DEPENDS>
    <COMPONENT name="VirtualEnquiring"/>
    <COMPONENT name="AudioVideoDecoder"/>
    <COMPONENT name="AudioMixRouter"/>
  </DEPENDS>
  <PROPERTIES>
    <PROPERTY name="Buffer" default="10000"/>
    <PROPERTY name="DiskCache" default="4000"/>
    <PROPERTY name="Jitter" default="2"/>
    <PROPERTY name="VideoEncoding"
        values="MPEG-1, MPEG-2, MPEG-4"
        order="asc"/>
  </PROPERTIES>
  <PORTS>
    <PORT name="AudioInput">
      <PROPERTY name="Gain" values="0..100" order="asc"/>
      <PROPERTY name="Codification" values="PCM"/>
    </PORT>
    <PORT name="VideoOutput">
      <PROPERTY name="FrameRate"
          values="20,25,30"
          order="desc"/>
      <PROPERTY name="ScreenWidth"
          values="250..300"/>
      <PROPERTY name="ScreenHeight"
          values="200..300"
          order="desc"/>
    </PORT>
    <PORT name="AudioOutput">
      <PROPERTY name="Sampling" values="11.4, 22.8,
      44.19"/>
      <PROPERTY name="Volume" values="1..10" default="5"/>
    </PORT>
  </PORTS>
  <CONNECTIONS>
    <CONNECT from="AudioVideoDecoder:MidIn"
        to="VirtualCheering:AudioInput"/>
    <CONNECT from="VirtualCheering:VideoOutput"
        to="VideoMixRouter:VideoOut"/>
    <CONNECT from="VirtualCheering:AudioOutput"
        to="AudioMixRouter:AudioOut"/>
  </CONNECTIONS>
</COMPONENT>
```

Fig. 7. Descrição XML do serviço torcida virtual

A especificação define um componente *VirtualCheering* que realiza um serviço de torcida virtual. Neste serviço, sinais de áudio capturados nos microfones das televisões clientes são enviados a um servidor, mixados e transmitidos aos clientes. O componente *VirtualCheering* depende de três

componentes: *VirtualEnquiring*, que realiza a função de enquete virtual, coletando dados dos espectadores sobre uma determinada enquete e enviando a um servidor; e os dois últimos, *AudioVideoDecoder* e *AudioMixRouter*, para manipulação de fluxos de áudio e vídeo.

Vale ressaltar que não são discutidos todos os aspectos e possibilidades da aplicação. A descrição apresentada tem por objetivo apenas fornecer uma visão da forma como as aplicações são construídas para o *middleware* AdapTV. A seguir, explica-se brevemente algumas *tags* XML adotadas:

- DEPENDS: identifica os componentes dos quais o componente descrito depende;
- ATTRIBUTES: enumera os atributos funcionais do componente descrito;
- PROPERTIES: identifica os atributos não funcionais (ou propriedades);
- PORTS: identifica as portas de entrada e saída do componente descrito;
  - PROPERTY: identifica as propriedades de cada uma das portas especificadas;
- CONNECTIONS: indica as conexões entre as portas dos componentes.

É importante observar que algumas propriedades são descritas através de um conjunto de valores. Para um conjunto de possíveis valores v_1, v_2, v_3, ... , v_n, as formas possíveis para especificá-lo são:

- v_1 .. v_n : conjunto de valores desde v_1 até v_n
- v_1, v_2, v_n : conjunto contendo apenas v_1, v_2 e v_n
- v_1 .. v_2, v_m .. v_n : conjunto com todos os valores contidos nas faixas especificadas

Desta forma, estabelece-se um conjunto finito de valores para os quais aquela propriedade poderá ser instanciada. A ordem de prioridade (nível de QoS que interessa à aplicação) dos elementos deste conjunto de valores é definida de acordo com a tag *order*, que pode assumir os valores a*sc* ou *desc*, indicando ao *configurator* que dê prioridade aos elementos do início ou fim deste conjunto, respectivamente.

VII. CONSIDERAÇÕES FINAIS

A proposta apresentada neste artigo tem como objetivo o desenvolvimento de um *middleware* de suporte ao desenvolvimento de aplicações de TVDI.

Este middleware deve fornecer flexibilidade e abrangência, permitindo através do modelo de recursos adotado, representar uma diversidade de recursos de software, hardware e sistemas operacionais. O objetivo é que componentes possam se adequar ao padrão que venha a ser adotado no Brasil para sistemas de TVDI, assim como viabilizar a sua aplicação nos padrões já definidos para outros países. Uma característica importante do middleware é a utilização de componentes arquiteturais ativos, como por exemplo o *configurator* e monitores de QoS.

A linguagem XML tem sido intensivamente explorada na literatura e apresentada como uma alternativa interessante em termos de linguagem de descrição arquitetural de componentes de software. O exemplo especificado demonstra o potencial do *middleware* proposto em termos de reusabilidade de componentes da aplicação. Este exemplo foi efetivamente desenvolvido em outro contexto e, através da especificação XML, o mesmo foi mapeado para o *middleware*. Desta forma, mostra-se que aplicações já desenvolvidas no contexto Web poderão ser portadas para o sistema de TV interativa.

Atualmente, alguns componentes do middleware encontram-se implementados: o núcleo do framework *Cosmos* e o *parser XML*. Os demais componentes do *AdapTV* estão em fase de implementação. Testes iniciais com o protótipo estão sendo realizados em laboratório, ainda em um contexto centralizado. Estes testes são promissores e mostram a viabilidade da proposta integrada de *Cosmos*/*AdapTV*. O objetivo é fazer uma avaliação e validação por etapas.

Como perspectiva de trabalhos futuros, pretende-se estender o protótipo, de modo a construir um ambiente de desenvolvimento com ferramentas para processamento de especificações de aplicações em XML em um universo ampliado de componentes. A idéia é que o *configurator* possa desempenhar o papel de um negociador de fato, ou idealmente, que realize buscas generalizadas de componentes, a partir de chaves associadas a características desejadas para determinados componentes em um sistema distribuído.

REFERÊNCIAS

[1] *WebTV*, em http://www.webtv.com/company/index.html.

[2] *OpenTV*, em http://www.opentv.com

[3] *Microsoft* TV, em http://www.microsoft.com/tv

[4] *JavaTV*, em http://java.sun.com/products/javatv

[5] Lopes A. B, Elias G. e Magalhães M. F. *COSMOS*: *Um Framework para Configuração e Gerenciamento de Sistemas Distribuídos Abertos*. In: III Workshop de Teses e Dissertações em Multimídia, Hipermídia e Web. Webmidia 2003, Salvador, BA.

[6] Borelli, F., Elias, G. *Uma Arquitetura de Middleware para Sistemas de Televisão Interativa*. In: III Workshop de Desenvolvimento Baseado em Componentes, São Carlos, SP, 2003.

[7] Souza F. B. e Elias G. AdapTV: *Um Middleware Adaptativo para Sistemas de Televisão Digital Interativa.* In: III Workshop de Teses e Dissertações em Multimídia, Hipermídia e Web. Webmidia 2003, Salvador, BA.

[8] International Organization for Standardization. *ISO/IEC 13818 Part 1: Systems*

[9] OMG. *The Audio/Video Streams Specification – Version 1.0*, Object Management Group, 2000.

[10] Duke D.J., Herman I. *A Standard for Multimedia Middleware*. In: ACM Multimedia, 1998

[11] Blair, G.S., Coulson G., et all. *The Design and Implementation of Open ORB Version 2*. In: IEEE Distributed Journal, 2001, vol 2, no. 6

[12] Duran-Limon, H., Blair, G.S. *Reconfiguration of Resources in Middleware*. In: IEEE International Workshop on Object Real-Time Dependable System (WRDS 2002) San Diego, 2002

[13] W3C. *The Extensible Markup Language*. Em: www.w3.org/XML

[14] *MHP Specification*, em http://www.mhp.org/technical_essen/specification.html

[15] *Nist Dase Development Environment*, em http://www.itl.nist.gov/div895/cmr/dase/

[16] *ARIB STD-24, Data Coding And Transmission Specification For Digital Broadcasting* , em http://www.arib.or.jp/english/index.html

[17] Tavares T., et al. *Sharing Virtual Acoustic Spaces over Interactive TV Programs – Presenting "Virtual Cheering" Application*. Aceito para apresentação em: The 2004 IEEE Intternational Conference on Multimedia and Expo-ICME´2004.