Especi£cação de Canal de Retorno em Aplicações para TV Digital Interativa

Cidcley T. de Souza e Carina T. de Oliveira

***Resumo*— Uma das principais características das aplicações de TV Interativa é a presença de recursos de interatividade em aplicações junto ao conteúdo televisivo. Para a construção dessas aplicações, é necessária a adoção de novas técnicas de engenharia de software que considerem suas características peculiares. Nesse sentido, apresentamos nesse artigo um estilo arquitetural que permite capturar os conceitos por trás dos níveis de interatividade das aplicações de TV Interativa de forma a facilitar o projeto dessas aplicações.**

***Palavras-Chave*— TV Digital, Canal de Retorno, Arquitetura de Software.**

***Abstract*— One of the main characteristics of Interactive TV applications is the presence of interactivity resources in applications along with the TV contents. In order to build such applications, it is necessary to employ new software engineering techniques that take into consideration their particular characteristics. With that in mind, we present in this paper an architectural style that allows the gathering of the concepts behind the levels of interactivity of the Interactive TV applications in a way to facilitate the project of such applications.**

***Keywords*— Digital TV, Return Channel, Software Architecture.**

I. Introdução

A partir dos anos 30, a televisão instalou-se nos lares de todo o mundo e, desde então, tem se consolidado como um importante veículo de comunicação, causando grande impacto em diversas áreas do conhecimento: educação, cultura, comércio, política, etc. Nos últimos anos, além da televisão, o computador também vem se destacando como um instrumento uni£cador dos meios de acesso à informação, porém é uma tecnologia que não bene£cia a todos. A di£culdade de acesso a um computador conectado à Internet priva a maior parte da sociedade de informação e, conseqüentemente, conhecimento.

Baseando-se nesse cenário, o Brasil está desenvolvendo um novo padrão de televisão, aberta e gratuita, com tecnologia superior à atual e que atenda às necessidades da sociedade brasileira. O modelo de referência para o Sistema Brasileiro de Televisão Digital (SBTVD) [4] segue a tendência mundial da convergência digital para que, em um futuro próximo, a televisão ofereça os recursos que hoje só são acessíveis aos que possuem um computador conectado à Internet ou àqueles que pagam para obter novos recursos na televisão. Porém, apesar da importância da transição tecnológica, o principal objetivo do SBTVD é a inclusão digital [3]. Como a televisão atinge praticamente toda a população do país, torna-se assim uma poderosa ferramenta de integração nacional [6].

O SBTVD oferecerá aos brasileiros diversas vantagens com relação à televisão analógica, como uma melhor qualidade de som e imagem, otimização de cobertura, maior diversidade de programação devido ao maior número de canais disponíveis, acesso à Internet e, principalmente, uma nova gama de aplicações interativas englobando texto, vídeo, áudio e os mais variados tipos de elementos grá£cos. Contudo, a construção dessas aplicações irá exigir da indústria de software nacional uma adaptação das técnicas de engenharia de software de forma a contemplar as novas características dessa mídia.

Argumentamos, nesse trabalho, que a utilização de uma estratégia de projeto arquitetural que considere às características peculiares das aplicações de TV Digital Interativa (TVDI) possibilitará uma melhor produtividade no desenvolvimento dessas aplicações. Dessa forma, propomos um novo estilo arquitetural, denominado iTVX, que permite a especi£cação arquitetural de aplicações para TVDI de forma a tratar explicitamente aspectos de interatividade dessas aplicações.

Além dessa seção, onde apresentamos as principais motivações desse trabalho, apresentamos na seção 2 uma descrição geral sobre TVDI, ressaltando os níveis de interatividade das aplicações. Na seção 3, apresentamos uma visão geral sobre o processo de especi£cação de arquiteturas distribuídas. Na seção 4, apresentamos o framework DraX+, um conjunto de ferramentas para a realização de projeto arquitetural de aplicações distribuídas que estenderemos para contemplar as características exigidas das aplicações de TVDI. Além disso, apresentamos nessa mesma seção, o estilo arquitetural iTVX. Um exemplo da aplicação de iTVX é mostrado na seção 5. Concluímos esse trabalho na seção 6 com uma discussão sobre a utilização de iTVX e alguns direcionamentos futuros.

II. Televisão Digital Interativa

A arquitetura apresentada no sistema de TV analógica difere em importantes aspectos quando comparada com a arquitetura do sistema de TV digital. A arquitetura básica de um sistema de TV analógica é formada por três componentes: um emissor, um meio de difusão e uma recepção doméstica [6]. O emissor tem como função gerar o sinal dos programas de televisão produzidos em estúdio para que, posteriormente, o meio de difusão (terrestre, satélite ou cabo) possa transmiti-lo unidirecionalmente. A recepção domestica é composta, basicamente, por uma antena que recebe o sinal do meio e um receptor analógico, geralmente embutido na televisão, que envia o sinal para o monitor do telespectador.

Centro Federal de Educação Tecnológica do Ceará (CEFET-CE), Gerência de Telemática, Fortaleza, Ceará, Brasil, E-mails: cidcley@cefetce.br,carinatoliv@gmail.com. NASH (Núcleo Avançado em arquitetura de Software distribuído e sistemas Hipermídia.)

Existem sistemas internacionais reconhecidos de TV digital: o europeu *Digital Video Broadcasting* (DVB) [5], o norte-americano *Advanced Television System Committee* (ATSC) [2] e o japonês *Integrated Services Digital Broadcasting* (ISDB) [8]. Estes sistemas podem ser representados através de uma arquitetura em camadas, onde cada camada oferece serviços para uma camada superior e utiliza os serviços que são oferecidos por uma camada inferior. Apesar da representação das arquiteturas ser a mesma, estas diferem quanto às aplicações, aos processos de modulação, codi£cação, compressão e transmissão, e quanto ao middleware adotado.

A interatividade é um recurso essencial para uma TV digital de qualidade, criando-se, assim, através da união destas, uma TV digital interativa denominada Interactive Digital Television Systems (IDTV). Através da IDTV, o telespectador abandona o papel meramente passivo para se tornar ativo frente à programação, com o poder de interagir com a televisão e elaborar o seu próprio conteúdo televisivo. Do ponto de vista técnico, o grau de interação do usuário pode ser classi£cado em três categorias: local, intermitente e permanente [6].

A interatividade local pode ser considerada como a mais básica das categorias. A antena doméstica continua recebendo os ¤uxos de áudio e vídeo, enquanto o receptor digital, denominado Set Top Box (STB), é o responsável pelo armazenamento do ¤uxo de dados, ou seja, pelas aplicações que estão sendo executadas e que permitem ao telespectador a interação propriamente dita. Porém, o telespectador não consegue realizar o envio de dados em direção ao emissor, pois não possui um canal de retorno ou canal de interação. Como exemplos de aplicações para este nível de interatividade pode-se citar a con£guração de legendas, jogos residentes, guias de programação *Electronic Program Guides* (EPGs), etc.

Na interatividade intermitente, a antena doméstica continua exercendo a função de receptora de ¤uxos de áudio e vídeo. Já o STB apresenta uma mudança signi£cativa para o aumento da interatividade: um canal de retorno. Este canal permite que o telespectador transmita ¤uxos de dados ao servidor, mas este não pode enviar respostas ao telespectador. Por isso, o canal de retorno é considerado não-dedicado. Sendo uma comunicação de dados unidirecional, essa categoria de interatividade é muito utilizada em aplicações como votações, pesquisas de opinião, entre outras.

A interatividade permanente é uma evolução da interatividade intermitente, na qual a comunicação dos dados deixa de ser unidirecional para se tornar bidirecional, existindo para isso um canal de retorno dedicado no STB. Através deste nível de interatividade, é possível ter o acesso às funções básicas de um computador conectado à Internet e usufruir aplicações como navegação, e-mail, chat, competições interativas (e.g. jogos multi-usuários em tempo real), compras, homebanking, educação à distância, entre outras.

III. Especificação de Arquiteturas Distribuídas

Arquitetura de software [9] tem se mostrado uma disciplina realmente útil por desempenhar um papel importante no processo de desenvolvimento do software, separando os elementos de computação (componentes) dos mecanismos de comunicação (conectores). Esta separação permite e facilita a promoção do reuso em níveis abstratos, um dos principais benefícios fornecidos pelo desenvolvimento arquitetural. Entretanto, para que esses benefícios da arquitetura de software sejam realmente alcançados, a mesma deve ser tratada explicitamente servindo como base para análise, projeto e implementação.

A adoção de estilos arquiteturais também contribui para a elaboração de especi£cações reusáveis em contextos especí£cos. Um estilo arquitetural de£ne um vocabulário de elementos de projeto e um conjunto de restrições sobre como esses elementos podem ser combinados, permitindo o reuso de organizações arquiteturais estabelecidas para resolver um determinado problema recorrente. Existem numerosos estilos arquiteturais, como por exemplo: cliente-servidor, pipeline, entre outros.

Estilos arquiteturais são importantes na etapa de especi£cação da arquitetura, mas, para que a arquitetura seja a base do projeto, ela deve ser comunicada claramente para todos os participantes do processo de desenvolvimento. Nesse contexto, destacamos as ADLs (Architecture Description Languages) para a realização dessa tarefa.

Em [10], apresentamos um framework, denominado DraX (*DistRibuted Architecture based on XML*). DraX é formado por um conjunto de ferramentas baseadas em tecnologias de larga aceitação no mercado e na academia para especi£cação, veri£cação, análise e implementação de arquiteturas de software e de estilos arquiteturais distribuídos.

A. O Framework DraX

O framework DraX é formado por um conjunto de linguagens e ferramentas que dão suporte a todo o processo de desenvolvimento baseado em arquitetura, que vai desde a especi£cação da arquitetura e do estilo arquitetural, passando pela fase de veri£cação de especi£cações até a chegar à geração de código. Desse modo, como o ferramental de DraX é relativamente extenso, resolvemos criar uma metodologia que possa guiar os passos de desenvolvimento de arquiteturas com o framework.

Podemos visualizar duas etapas de desenvolvimento distintas e paralelas que interagem entre si. Na primeira etapa, a arquitetura da aplicação é descrita; na segunda etapa o estilo arquitetural é descrito, caso esse ainda não tenha sido especi£cado. O resultado do desenvolvimento de um estilo é sua especi£cação em Xtyle devidamente validada. Essa especi£cação poderá ser referenciada na etapa de descrição de uma arquitetura ArchML. A tarefa de validação dessa arquitetura, entre outras funções, veri£cará a aderência da mesma ao estilo referenciado.

A metodologia geral que propomos para a utilização de DraX para especi£cação de arquiteturas distribuídas é composta pelas seguintes etapas:

- Especi£cação: Nessa fase, são especi£cadas as arquiteturas e os estilos arquiteturais, utilizando as linguagens ArchML [11] e Xtyle [13] respectivamente;
- Validação: A veri£cação sintática, estrutural e comportamental das arquiteturas e estilos é realizada nessa fase,

sendo utilizados os esquemas e scripts de validação e veri£cação de DraX;
- Geração de Código: Os templates de implementação são gerados a partir das especi£cações arquiteturais previamente validadas e das informações dos estilos que essas arquiteturas utilizam.

Nesse trabalho, por limitação de espaço, apresentaremos apenas um breve resumo sobre a especi£cação sintática de arquiteturas e estilos arquiteturais.

1) Especi£cação Sintática de Componentes e Arquiteturas: A fase inicial do desenvolvimento com DraX é realizada através da especi£cação das arquiteturas e estilos arquiteturais. Cada arquitetura é especi£cada através da referência a componentes que são especi£cados externamente. Nesse caso, instâncias de cada tipo de componente são de£nidas para formar uma arquitetura. Em DraX, a linguagem ArchML foi de£nida para realizar essa tarefa.

Além da de£nição da arquitetura em si, essa pode seguir um estilo arquitetural. A obediência a um estilo em DraX é de£nida através da aderência dos componentes a um vocabulário especí£co de£nido no estilo. Nesse caso, cada componente da arquitetura deve ser de um ou mais tipos que são de£nidos no estilo (ou estilos) que a arquitetura segue. Esses estilos, por sua vez, tanto podem já existir previamente, como podem ser criados através da linguagem Xtyle.

Por ser uma aplicação de XML, ArchML especi£ca componentes através de marcações especí£cas. A Figura 1 apresenta a estrutura geral de um documento de descrição de um componente em ArchML. Podemos observar os elementos básicos para a de£nição de documentação, propriedades, interfaces e comportamento. Em ArchML, um componente funciona como um tipo que deve ser utilizado na descrição de instâncias especí£cas no momento da criação de arquiteturas.

Também deve ser observada a possibilidade de um componente possuir mais de uma interface. Essa faceta de ArchML permite uma maior versatilidade na utilização de um componente, possibilitando a utilização desses componentes em diferentes tipos de interações.

```
<?xml version="1.0"?>
<component name="RemoveVowels"/>
<document>
<interfaces>
<interface role="Filter">
<port name="getPhrase"direction="in">
<param name="phrase"type="xsd:string"/>
</port>
</interface>
</interfaces>
<behavior = "getPhrase(data2).t.
RemoveVowels(outPhrase)"/>
</component>
```

Fig. 1. Trecho de Especi£cação Sintática de um Componente em ArchML.

O elemento behavior tem como função referenciar a especi£cação comportamental do componente. Como DraX tem como um de seus objetivos permitir a especi£cação e validação de arquiteturas e de estilos arquiteturais distribuídas e como na fase de validação apenas os dados sintáticos não são su£cientes, é necessária a utilização de características comportamentais. Assim, em ArchML é possível se descrever o comportamento dos componentes. Essa descrição diz respeito ao comportamento observável com relação às portas de£nidas nas interfaces do componente, a qual é realizada utilizando a álgebra de processos $\mathcal{R}\pi$ [12]. Entretanto, por limitação de espaço, não entraremos em maiores detalhes sobre tal aspecto nesse trabalho.

Um Sistema, nome dado a essa estrutura em ArchML, é a descrição estrutural de uma arquitetura distribuída. Eventualmente, referenciaremos por arquitetura os sistemas ArchML, visto que esses termos são sinônimos. A Figura 2 apresenta uma especi£cação de sistema em ArchML.

```
<?xml version="1.0"?>
<document/>
<style href="Pipeline.xty"/>
<types>
<type name="CompCatFile"href="CatFile.aml"/>
...
</types>
<instances>
<instance name="catfile"typeRef="CompCatFile"/>
...
</instances>
<links>
<link name="conn1»
<point instRef="catfile"portRef="output"/>
<point instRef="removevowels"portRef="getPhrase"/>
</link>
...
</links>
</system>
```

Fig. 2. Trecho de Especi£cação Sintática de um Sistema em ArchML.

2) Especi£cação Sintática de Estilos: O termo estilo arquitetural se refere a "um conjunto de regras de projeto que identi£ca os tipos de componentes e conectores que devem ser utilizados para compor um sistema ou subsistema, juntamente com um conjunto de restrições globais e locais na forma como essa composição deve ser realizada"[1]. Garlan, em [7], apresenta de forma mais detalhada um conjunto de aspectos de estilos arquiteturais que devem ser utilizados por qualquer modelo que trabalhe com esse conceito.

Na Figura 3, é mostrado um exemplo de estilo descrito por Xtyle. Podemos observar os elementos básicos de uma especi£cação de estilos referentes a: documentação, tipos e topologia. O elemento xtyle, a raiz da especi£cação, possui um atributo name, que serve para identi£car o estilo que está sendo descrito. O elemento document, que também é usado em ArchML, serve para de£nir informações sobre a realização da especi£cação. topológicos.

O vocabulário do estilo é de£nido no elemento types,além das informações sobre possíveis restrições nos componentes que utilizam esses tipos. Esse elemento é formado por um conjunto de elementos type. Cada type identi£ca um tipo diferente de componente, permitido no estilo.

O elemento topology de£ne informações sobre as interações entre os tipos de£nidos para o estilo. Esse elemento é formado

```
<?xml version="1.0"?>
<xtyle name="Pipeline»
<types>
<type name="Filter">
<ports>
<in mode="async"/>
<out mode="async"/>
</ports>
<behavior href="Filter.xmi"/>
</type>
...
</types>
<topology>
<link name="SourceFilter"start="Source"
      end="Filter"type="push"/>
...
</topology>
</xtyle>
```

Fig. 3. Trecho de uma Especi£cação Xtyle.

por um conjunto de elementos do tipo link. Cada elemento link representa uma conexão permitida entre os tipos de£nidos no estilo. Os atributos start e end de link de£nem os nomes dos tipos que podem ser conectados.

IV. A Extensão DraX+ e o Estilo Arquitetural iTVX

De forma a representar arquiteturalmente as aplicações para TVDI, o framework DraX deve ser estendido de modo a permitir se explicitar as formas de interação entre os componentes das aplicações. Assim, aspectos de comunicação, que em DraX eram deixados para o middleware onde as aplicações deveriam executar, agora devem ser explicitamente modelados.

Nesse sentido, submetemos uma extensão de DraX, que denominaremos de DraX+, na qual incluímos a especi£cação de conectores. Esses conectores permitirão a de£nição das interações das aplicações de TVDI considerando os níveis de interatividade, apresentados na seção 2 desse trabalho.

A. A Extensão DraX+

Para estender DraX de forma a suportar a de£nição de conectores, devemos, inicialmente, incluir uma gramática para permitir a especi£cação dessa nova abstração. Desse modo, seguiremos a seguinte estrutura básica na de£nição de conectores:

```
<?xml version="1.0"?>
<connector name="xsd:string"
<interfaces>
<interface role="xsd:string»+
<port name="xsd:string"direction="xsd:string»+
<param name="xsd:string"type="xsd:string"/>*
</port>
<behavior/>
</interface>
</interfaces>
</connector>
```

Essa especi£cação simpli£cada de uma gramática XML permite visualizar os principais elementos e estruturas da especi£cação de um conector de DraX+. Podemos observar que elementos como document, interfaces e behavior, devem aparecer obrigatoriamente, e uma única vez, na especi£cação de um conector. Esses elementos são utilizados, respectivamente, para a descrição de informações gerais sobre a especi£cação, para a de£nição de interfaces de um conector e para a de£nição do comportamento observável esperado para cada tipo de interface do conector.

Adicionalmente, devemos permitir a introdução de conectores da descrição de arquiteturas. Essa característica pode ser facilmente conseguida sem a necessidade de nenhuma modi£cação na gramática de descrição de sistemas de DraX (ArchML), visto que os conectores podem ser instanciados como componentes e na criação de links, necessitando apenas mais um elemento link, que deverá conectar uma porta de um componente com uma porta de um conector.

Para que possamos especi£car estilos em DraX+, realizamos uma extensão na linguagem Xtyle, que denominamos de Xtyle+, de forma a incorporar a abstração de conectores nessa linguagem. Contudo, essa extensão também é trivial, visto que precisamos apenas de£nir novos tipos (elemento type) relativos aos conectores dentro da especi£cação Xtyle.

V. O Estilo Arquitetural iTVX

De forma simpli£cada, podemos partir do princípio de que aplicações de TVDI são basicamente aplicações do tipo cliente-servidor que utilizam canais de comunicação com restrições relativas aos requisitos de interatividade. Dessa forma, propomos, nessa seção, um novo estilo arquitetural, que denominamos de iTVX, o qual é derivado do estilo Cliente-Servidor, e que inclui restrições topológicas e comportamentais que consideram os diferentes aspectos de interatividade das aplicações de TVDI. De fato, iTVX é uma família de estilos arquiteturais (três, no total) que permitem a descrição arquitetural de aplicações para TVDI.

1) Descrição Informal: Como em todo estilo arquitetural, na de£nição dos estilos de iTVX, necessitamos especi£car os tipos de componentes, os tipos de conectores e as restrições arquiteturais para esses estilos. Realizamos essa especi£cação baseada nas seguintes características:

Componentes: Se considerarmos que as aplicações de TVDI possuem características próprias das aplicações que seguem o estilo arquitetural Cliente-Servidor, os mesmos componentes utilizados na especi£cação arquitetural desse estilo serão utilizados, que são:

- Cliente: componente que inicia uma solicitação ao Servidor, aguardando o resultado da computação do Servidor ou não;
- Servidor: recebe solicitação de Clientes, executando algum tipo de computação interna, retornando, em seguida, o resultado para o(s) Cliente(s) ou não;

Conectores: Os conectores devem permitir a abstração dos diferentes tipos de interatividade relacionados ao canal de retorno. Dessa forma, de£nimos os seguintes tipos de conectores:

- iTVXConnL: conector que representa uma comunicação com nível de interatividade local;

- iTVXConnI: conector que representa uma comunicação com nível de interatividade com canal de retorno intermitente;
- iTVXConnP: conector que representa uma comunicação com nível de interatividade com canal de retorno permanente.

Restrições Arquiteturais: Como relação ao ¤uxo de dados e controle, estes seguem o padrão adotado pelo estilo Cliente-Servidor, ou seja, a comunicação sempre é iniciada pelo Cliente e retornada pelo Servidor (se necessário). Quanto às restrições topológicas, sempre devemos ligar um ou mais componentes do tipo Cliente a um componente do tipo Servidor.

2) Especi£cação Xtyle+: A especi£cação Xtyle+ do estilo iTVXL é representada da seguinte forma:

```
<?xml version="1.0"?> <xtyle name="iTVXL»
<types>
<type name="Client">
<ports>
<in minOccurs="0"/>
<out/>
</ports>
<behavior href="t.out<data>.(in(data2).0 + 0)"/>
</type>
<type name="Server">
<ports>
<in/>
<out minOccurs="0"/>
</ports>
<behavior href="in(data).t.(out<data2>.0 + 0)"/>
</type>
<type name="iTVXConnL">
<ports>
<in/>
<out/>
</ports>
<behavior href="in(data).out<data>"/>
</type>
</types>
<topology>
<link name="ClientConnL"start="Client"
      end="iTVXConnL"controlType="push"/>
<link name="ConnLServer"start="iTVXConnL"
      end="Server"controlType="push"/>
</topology>
</xtyle>
```

A descrição do comportamento do conector é o que pode diferir nas especi£cações dos estilos de iTVX. Dessa forma, por questão de simpli£cação, mostramos a seguir apenas as especi£cações dos conectores em cada novo estilo especi£cado. É a seguinte a especi£cação do estilo iTVXI:

```
<type name="iTVXConnI">
<ports>
<in/>
<out/>
</ports>
<behavior href="in(data).waitchannel(out).
      connect(out).out<data>.disconnect(out)"/>
</type>
```

Já a especi£cação do estilo iTVXP é a seguinte:

```
<type name="iTVXConnI">
<ports>
<in/>
<out/>
</ports>
<behavior href="in(data).connect(out).
      out<data>.disconnect(out)"/>
</type>
```

Observemos que na especi£cação do conector para simular a interatividade com canal de retorno intermitente, utilizamos o processo (canal) waitchannel, que deverá aguardar a liberação de um canal de comunicação para posterior conexão. Por sua vez, na especi£cação do conector para simulação do canal de retorno permanente, já realizamos a conexão diretamente. Observemos que mesmo sendo uma diferença sutil em nível de especi£cação formal, os resultados simulam exatamente o comportamento das aplicações para TVDI com relação à variação dos tipos de canais de retorno, permitindo uma avaliação formal do comportamento das aplicações.

Por não ser o foco do presente trabalho, não de£nimos aqui as restrições arquiteturais para esses estilos, mas elas podem ser facilmente especi£cadas utilizando as informações da seção 4.2.1.

VI. EXEMPLO

Para apresentar a aplicação dos estilos iTVX no suporte à descrição arquitetural de aplicações para TVDI, utilizaremos um exemplo simples de uma aplicação de e-shop (Figura 4). Essa aplicação permite que usuários selecionem produtos e posteriormente realizem a compra diretamente pela TV. A aplicação será carregada no STB e o usuário poderá realizar as interações através do controle remoto. Inicialmente, consideraremos que o canal de retorno será intermitente utilizando uma linha discada com a operadora de TV.

Essa aplicação, por sua vez, é composta por duas outras aplicações: uma aplicação que estará executando no STB e uma aplicação que estará executando na operadora. A primeira aplicação, que está no STB, que denominaremos de iTVShop-Client, possui uma porta de entrada (getProdList) para receber as informações gerais sobre os produtos e duas portas de saída, uma para receber detalhes sobre um produto (reqProdInfo), quando solicitado, e outra para realizar a compra (buyProd), onde serão enviados o número do usuário, um número de segurança (pin code) e outras informações sobre o pedido.

A aplicação que £ca na operadora, denominada iTVShopServer, tem uma porta de saída (sendProdList) para enviar as informações gerais sobre produtos e duas portas de entrada por onde receberá pedidos de informações detalhadas sobre um produto especí£co (sendProdInfo) e outra para receber os dados da compra (buyProd).

A especi£cação dos componentes dessa aplicação é trivial pelo fato de termos claramente um componente que está atuando com papel (role) de Cliente (iTVClient) e outro que atua com papel de Servidor (iTVServer). A especi£cação simpli£cada de arquitetura dessa aplicação utilizando os estilos iTVX com o canal de retorno intermitente, pode ser realizada da seguinte forma:

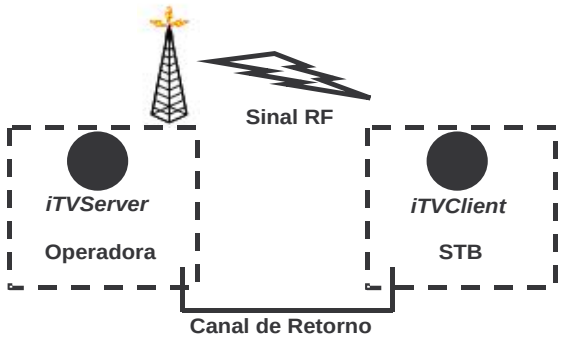

Fig. 4. Estrutura da Aplicação iTV e-shop

```
<?xml version="1.0"?>
<system>
<style href="iTVXI.xty"/>
<types>
<type name="iTVClient"href="iTVClient.aml"/>
<type name="iTVServer"href="iTVServer.aml"/>
<type name="iTVConnector"href="iTVXI.aml "/>
</types>
<instances>
<instance name="client"typeRef="iTVClient "/>
<instance name="server"typeRef="iTVServer "/>
<instance name="connI"typeRef="iTVConnector "/>
</instances>
<links>
<link>
<point instRef="client"portRef="reqProdList"/>
<point instRef="connI"portRef="in"/>
</link>
<link>
<point instRef="connI"portRef="out"/>
<point instRef="server"portRef="sendProdList"/>
</link>
<link>
<point instRef="client"portRef="buyProd"/>
<point instRef="connI"portRef="out"/>
</link>
<link>
<point instRef="connI"portRef="in"/>
<point instRef="server"portRef="buyProd"/>
</link>
</links>
</system>
```

Podemos modi£car a forma de interação do canal de retorno simplesmente modi£cando a informação sobre o estilo que está sendo utilizado pela aplicação. Assim, basta trocar, na primeira linha da especi£cação, o valor iTVXI.xtp (que identi£ca uma especi£cação Xtyle+) por iTVXP.xtp

Essa especi£cação é bem simpli£cada. Poderíamos utilizar um estilo mais complexo se considerarmos que as informações dos produtos £cam armazenadas localmente no STB e que a aplicação iTVClient deve acessar essas informações. Desse modo, teríamos um outro link, utilizando agora um conector com interatividade local.

VII. Conclusão

O Brasil vem atualmente trabalhando na proposta de um novo modelo de TV digital a ser adotado no país. O projeto SBTVD (Sistema Brasileiro de Televisão Digital) oferecerá aos brasileiros diversas vantagens com relação à televisão analógica, como uma melhor qualidade de som e imagem, otimização de cobertura, maior diversidade de programação devido ao maior número de canais disponíveis, acesso à Internet e, principalmente, uma nova gama de aplicações interativas englobando texto, vídeo, áudio e os mais variados tipos de elementos gra£cos.

Contudo, o desenvolvimento de aplicação para essa mídia requer a adoção e adaptação de ferramentas e tecnologias atualmente utilizadas no desenvolvimento de software tradicional, visto que existem diversas características diferentes a serem consideradas.

No presente trabalho, propomos a utilização de projeto arquitetural na construção de aplicações para TV Digital Interativa (TVDI). Nesse sentido, apresentamos um estilo arquitetural, o iTVX, que permite a especi£cação de arquiteturas de aplicações para TVDI considerando aspectos de comunicação relativos ao canal de retorno para essas aplicações.

Através de iTVX podemos especi£car e validar formalmente as aplicações de TVDI utilizando ferramentas tradicionais. Atualmente, estamos trabalhando na geração automática de código a partir de especi£cações detalhadas em iTVX.

Referências

[1] Abowd, G. and Allen, R. and Garlan, D. Formalizing Styles to Understand Descriptions of Software Architecture. *ACM Transactions on Software Engineering and Methodology*, 1995.

[2] ATSC. Advanced Television Systems Committee. www.atsc.org, Último acesso: Maio de 2005.

[3] V. Becker and C Montez. *TV Digital Interativa: Conceitos, desa£os e perspectivas para o Brasil*. 2004.

[4] Ministério das Comunicações. Sistema de TV Digital. Disponível em <http://sbtvd.cpqd.com.br/historico_sbtvd.php>, Acesso em: 01 Março 2005.

[5] DVB. Digital Video Broadcasting Project. www.dvb.org, Último acesso: Maio de 2005.

[6] Lemos G. Fernandes, J. and G Silveira. *Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação (JAI-SBC2004)*, chapter Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas. 2004.

[7] Garlan, D. Style-Based Re£nement for Software Architecture. In *Joint Proceedings of the Second International Software Architecture Workshop (ISAW2) and the International Workshop on Multiple Perspectives in Software Development (Viewpoints '96)*. ACM Press, 1996.

[8] ISDB. ISDB-T - Terrestrial Integrated Services Digital Broadcasting (ISDB-T): Speci£cation of Channel Coding, Framing Structure and Modulation. Technical report, 1999.

[9] Shaw, M. and Garlan, D. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[10] Souza, Cidcley T. de. Arquiteturas de Software e Estilos Arquiteturais Distribuídos - Especi£cação, Validação, Análise e Implementação. *Tese de Doutorado. Universidade Federal de Pernambuco*, 2003.

[11] Souza, Cidcley T. de and Cunha, Paulo R. F. Especi£cando Arquiteturas de Software em XML. In *XXVII Conferêcia Latino-Americana de Informáica*, Mérida, Venezuela, 2001.

[12] Souza, Cidcley T. de and Cunha, Paulo R. F. Reusing Formal Speci£cation of Components. In *Proc. 6th IASTED International Conference on Software Engineering and Applications (SEA 2002)*, MIT, Cambridge, USA, 2002.

[13] Souza, Cidcley T. de and Cunha, Paulo. R. F. Especi£cação de Estilos em Arquiteturas de Software. In *XXIX Conferência Latino-Americana de Informática.*, La Paz, Bolivia, 2003.