Universidade Federal de Pernambuco
Graduação em Engenharia da Computação
Centro de Informática

2010.2

eColect - Desenvolvimento de uma aplicação responsável por coletar dados Sensíveis a Contexto no ambiente de TV Digital

Trabalho de Graduação

Aluno – Luiz Augusto Zelaquett de Souza (lazs@cin.ufpe.br)
Orientador – Carlos André Guimarães Ferraz (cagf@cin.ufpe.br)

Recife, 14 de Dezembro de 2010.

Luiz Augusto Zelaquett de Souza

eColect - Desenvolvimento de uma aplicação responsável por coletar dados Sensíveis a Contexto no ambiente de TV Digital

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: Profº Dr. Carlos Ferraz

Recife

2010

Assinaturas

Dezembro de 2010

Carlos André Guimarães Ferraz
(orientador)

Luiz Augusto Zelaquett de Souza
(graduando)

Agradecimentos

Agradeço primeiramente aos meus pais Edwaldo Gomes e Gislane Zelaquett, que sempre me apoiaram e também as minhas irmãs Elaine e Cristine, e minha vó Yolanda.

Agradeço ao meu orientador Carlos Ferraz pela confiança dada em mim na realização deste trabalho e pelo apoio dado durante todos os anos no CESAR.

Agradeço também a Mário Fried, então gerente da equipe de TV Digital do CESAR, na qual eu participei por 4 anos, por ser um ótimo chefe e permitir que eu tivesse acesso a toda tecnologia necessária para construção de meus conhecimentos.

Agradeço também todos que participaram do grupo de TV Digital do CESAR, por serem ótimos companheiros, Paulyne Jucá, Andrino Coêlho e Jayro Santos, em especial a Diogo Pedrosa e Artur Tenório.

E a todos que contribuíram, direta ou indiretamente, para a conclusão deste trabalho, meus sinceros agradecimentos.

Resumo

A TV Digital começa a se consolidar como produto acessível ao grande público. Fala-se muito em termos como "alta definição" e "HD", no entanto, a digitalização do sinal de TV possibilita também a interatividade, é a chamada TVDi.

Assumindo que a interatividade seja o próximo grande passo da TV e que se torne um produto de massa, medidas devem ser tomadas com a finalidade de potencializar as funcionalidades que passarão a integrar a TV.

A utilização da computação sensível ao contexto pode ser usada para proporcionar uma interatividade mais eficaz, conferindo a ela informações relativas ao telespectador, como por exemplo, o posicionamento, região geográfica que vive, interesses pessoais, etc.

Este trabalho de graduação propõe uma aplicação de TV Digital, em Opentv, que permite a coleta de dados do usuário, como o tempo de permanência em um canal, para que esses sejam utilizados na melhoria dos serviços providos aos telespectadores.

Palavras Chaves: TV Digital, Interatividade, Sensibilidade ao Contexto, Aplicações Interativas.

Abstract

Digital TV is beginning to consolidate as the product accessible to the general public. There is much talk in terms like "high definition" and "HD", however, the digitization of the TV signal also enables interactivity, is called TVDi.

Assuming that interactivity is the next big thing for TV and it becomes a product of mass, measures must be taken in order to enhance the features that will join the TV.

The use of context-sensitive computing can be used to provide a interaction more effective, giving her information on the viewer, for example, positioning, geographic region, living, personal interests, etc.

This paper proposes an application for graduation Digital TV in OpenTV, which allows the collection of user data, including time spent in one channel, so these are used for the improvement of services provided to viewers.
Este trabalho de graduação propõe uma aplicação que permita a coleta de dados do usuário, para que esses sejam utilizados na melhoria dos serviços providos aos telespectadores.

Key-words: Digital TV, Interactive, Context-Aware, Interactive Application.

Sumário

Índice de Figuras

Índice de Tabelas

Índice de Siglas

API	*Abstract Programming Interface*
ARIB	*Association of Radio Industries and Businesses*
ATSC	*Advanced Television System Committee*
BD	*Banco de Dados*
DVB	*Dgital Video Broadcast*
EPG	*Eletronic Program Guide*
ES	*Elementary Stream*
HD	*High Definition*
HTML	*Hyper Text Markup Language*
IDE	*Integrated Development Environment*
ISDB	*Integrated Services Digital Broadcasting*
ISDTV	*International System for Digital Tlevision*
IP	*Ìnternet Protocol*
MHP	*Multimedia Home Platform*
MPEG	*Moving Picture Experts Group*
NCL	*Nested Context Language*
TS	*Transport Stream*
TVD	*Televisão Digital*
TVDi	*Televisão Digital Interativa*
VoD	*Video on Demand*
SO	*Sistema Operacional*
STB	*Set-to- Box*
XML	*Extensible Markup Language*

1. Capítulo

Introdução

1.1. Motivação

A massificação do sinal digital de TV, o grande interesse dos usuários de TV por novas tecnologias, vídeo sobre demanda (VOD), *gadgets* que possibilitam ver a previsão do tempo e acesso a WEB através da televisão são exemplos de novas características agregadas a TV que cada vez mais farão parte do cotidiano dos usuários de TV.

Sabendo-se que o sistema de transmissão atual, seja ele aberto ou por assinatura, se baseia no ambiente broadcast, no qual uma programação única é enviada, pelos provedores de conteúdo, para todos seus usuários, e que, por exemplo, os usuários de sistemas por assinatura, só podem especificar os conteúdos recebidos, através de uma assinatura de determinado pacote ou canal.

Como entender os usuários e atender suas demandas?
Como definir se um canal não tem aceitação do público?
Como definir melhor um conjunto de vídeos sobre demanda que serão disponibilizados ao cliente?
Como traçar campanhas de marketing televisivo o mais eficiente possível?

São essas entre outras tantas perguntas que poderiam ser respondidas e solucionadas se houvesse acesso a informações precisas e em tempo real sobre telespectadores.

Pretendendo participar do processo de solução a essas perguntas é que nasceu a motivação para o desenvolvimento dessa proposta.

As construções de uma ferramenta que obtivesse informações precisas e as enviassem para análise, no ambiente da TV Digital.

1.2. Objetivos do trabalho

O objetivo desse trabalho é mostrar que existem formas simples de coleta de dados dos usuários que fazem uso da TV Digital.

Não é foco deste trabalho definir o funcionamento, nem a implementação da ferramenta responsável por tratar ou interpretar as informações coletadas.

Entretanto são propostas algumas formas de utilização desses dados, que podem parecer triviais quando vistos de forma pontual, mas que podem, quando coletados durante um período de tempo, formar um padrão e podem ser utilizados para definir melhores programações, horários de programas e que vídeos sob demanda devem ser disponibilizados.

1.3. Estrutura do Documento

O restante deste documento está organizado em 4 capítulos, divididos da seguinte forma, No Capítulo 2 estão descritas algumas características da TV Digital, das aplicações interativas e da ciência de contexto.

No Capítulo 3 são apresentados os relacionamentos entre a ciência de contexto e a aplicação, uma introdução do funcionamento da aplicação, bem como a definição do cenário de sua execução.

No Capítulo 4 encontram-se descritas as tecnologias que foram utilizadas na implementação de uma prova de conceito do que foi proposto no Capítulo 3.

Por fim, no Capítulo 5 as conclusões e trabalhos futuros são expostos.

2. Capítulo

Fundamentação teórica

2.1. A TV Digital

Melhorias na qualidade de áudio e vídeo são latentes ao sistema digital de TV, mas uma outra grande vantagem do sistema digital é que, como toda informação é digitalizada antes da transmissão, qualquer outro tipo de dado digital além do vídeo e áudio também pode ser difundido. Dessa forma, é possível a difusão de aplicativos a serem executados no receptor de TV digital. É baseado nesse outro benefício do sistema digital que as portas para a interatividade na televisão digital são abertas.

Já a partir do padrão MPEG-2, e posteriormente no MPEG-4, é possível multiplexar junto com o fluxo de áudio e vídeo, um fluxo de dados. Esse fluxo é o canal de transmissão das aplicações interativas.

O *middleware* desmembra todo o sinal enviado por um determinado canal de transmissão, separando-o em 3 tipos de fluxos elementares (ES): áudio, vídeo e dados. É feita a decodificação e interpretação de cada um desses fluxos. A parte de dados é enviada para o *middleware* que interpreta esses dados e carrega a aplicação.

Na Figura 1 é mostrada uma arquitetura genérica proposta para um sistema de TV Digital, muito semelhante em todos os sistemas hoje utilizados.

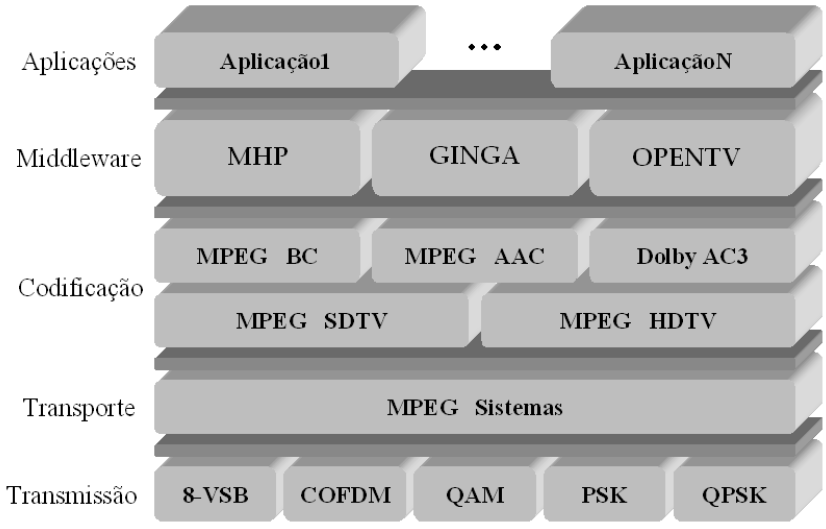

Figura 1 - Arquitetura genérica de um sistema TV Digital

Na arquitetura mostrada na Figura 1 o padrão de TV Digital, tomado como exemplo, encontra-se subdividido em 5 camadas, são elas:

1- Aplicação;
2- Middleware;
3- Codificação;
4- Transporte;
5- Transmissão.

Cada uma dessas camadas desempenha papeis importantíssimos na materialização de um sistema digital de TV.

A camada de Aplicação, contendo as aplicações, quem vêm em conjunto com os fluxos de áudio e vídeo.

A camada do Middleware, contendo todo o mecanismo de controle de execução de aplicações.

Já a camada de Codificação, desempenhando o papel de tratar como os fluxos de áudio e vídeo são encapsulados e desencapsulados.

E as camadas de Transporte e Transmissão responsáveis por papéis de baixo nível, como desempenhar que forma as informações serão enviadas ao mundo exterior ao da TV.

Reafirmando que a arquitetura proposta acima é bastante semelhante aos sistemas digitais de TV, que tanto arquiteturas como a do SBTVD ou ISDB-Tb[13], sistema utilizado no Brasil, como em sistema fechado, utilizado em diversas TVs a Cabo no mundo, fazem uso de um sistema bastante semelhante ao descrito acima, como podemos analisar abaixo.

Pode-se verificar que a arquitetura proposta na Figura 2 e utilizada pelo SBTVD se assemelha bastante com a arquitetura exposta na Figura 1.

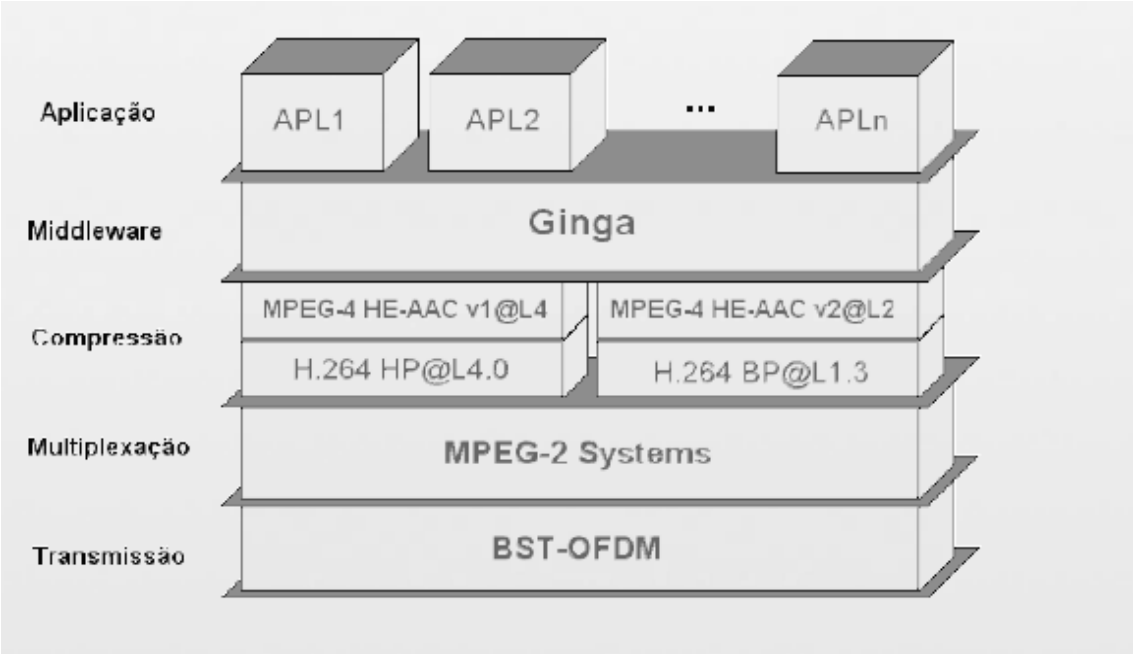

Figura 2 – Arquitetura do SBTVD[13]

Já em arquiteturas proprietárias, com as das TVs por assinatura, podemos analisar a Figura 3, que mostra claramente as camadas de Aplicação e a camada de Middleware, a camada a baixo, descreve a existência de um sistema operacional no qual o *middleware* é executado. Essa camada também existe na arquitetura mostrada na Figura 2, mas foi omitida, pois a referida figura mostra uma visão mais macro do sistema utilizado no SBTVD.

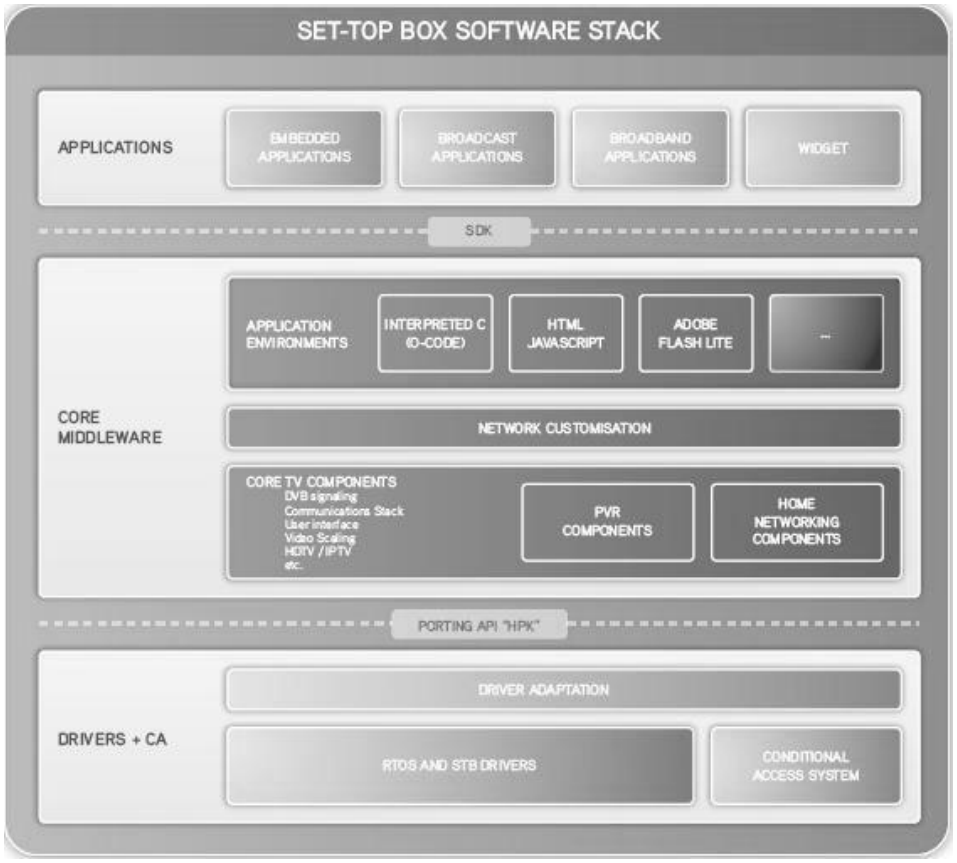

Figura 3 – Arquitetura do OPENTV[9]

A omissão de camadas na Figura 3, não invalida a afirmação de que a arquitetura proposta pela Figura 1 é genérica o bastante, isso porque, as camadas logo a baixo da do Middleware, no caso as camadas de Codificação, Transporte e Transmissão têm que existir para o sistema digital de TV existir, talvez como alguns componentes a mais ou a menos, mas em sua essência as arquiteturas se parecem.

2.2. Componentes importantes para Interatividade

Focando nos componentes que fazem com que a aplicação se materialize, ou seja, os componentes do sistema digital de TV que participam de forma mais ativa, desde a recepção dos fluxos de áudio, vídeo e dados, até a apresentação da aplicação para o usuário.

As camadas do sistema digital que mais participam desse processo de transformação de bits em algo visual a ser mostrado na tela da TV, se destacam as camadas de Aplicação e de Middleware, mas as camadas de mais baixo nível possuem componentes chave para apresentação das aplicações interativas, são eles:

-
-

- *Low Level Stream Acess* - Responsável por acessar os fluxos de transportes (TS), processá-los e demultiplexá-los nos vários fluxos elementares que os constituem, ou seja, num conjunto de fluxos de áudio, vídeo e dados;

- *ES Information* - Responsável por identificar e informar, aos elementos das camadas superiores quais fluxos elementares, no caso, dizer se o fluxo é de áudio, vídeo ou dados, que estão presentes em um determinado TS sintonizado;

- *Elementaty Stream Selection* - Responsável por disponibilizar determinado fluxo elementar às camadas superiores;

- *Elementary Stream Processing* - São os elementos responsáveis por processar os fluxos elementares, tais quais, os decodificadores de áudio e vídeo;

- *Data Processing* - São os itens responsáveis por processar e disponibilizar às camadas superiores os fluxo elementares de dados;

- *Application Management* - Responsável por carregar, configurar, gerenciar e executar as aplicações que irão funcionar no *middleware*.

No conjunto de itens relativos à comunicação com o ambiente exterior ao do STB temos o item:

- *Interactive Channel* - Elemento responsável por disponibilizar interfaces que possibilitam o acesso ao canal de interação, canal esse bidirecional, que permite a comunicação entre todo o *middleware* e o exterior.

Além do mais temos serviços destinados a permitir que objetos mesmo após terem sido utilizados, possam voltar a existir, sendo isso feito através da utilização de meios de armazenamento voláteis, como memórias flash, a esses itens conferimos a funcionalidade de persistência.

A existência desses componentes pode ser vista nas Figura 2 e Figura 3, no entanto, é na Figura 4 que são mostrados esses itens mais claramente.

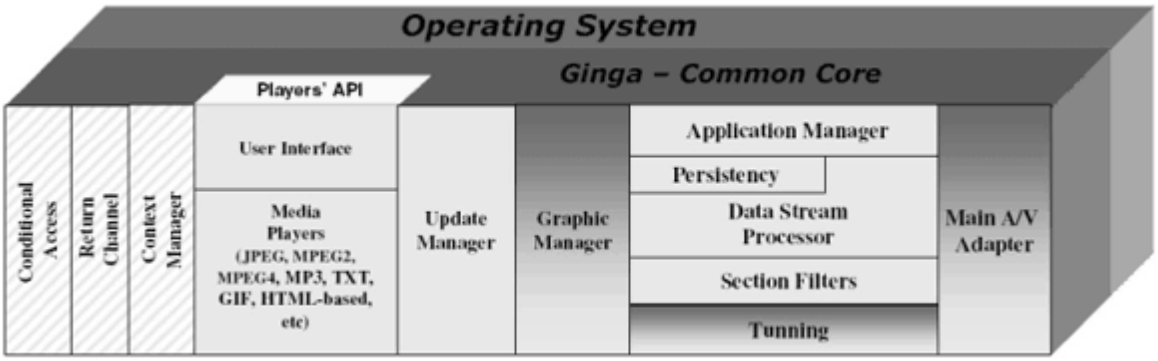

Figura 4 – Arquitetura do GINGA[12]

2.3. A INTERATIVIDADE

A interatividade que para Steuer [14] é definida como:

"a extensão em que os usuários podem participar modificando a forma e o conteúdo do ambiente mediado em tempo real"

Segundo ele, existem fatores que contribuem para potencializar a interatividade, são eles:

- A velocidade da interatividade, que corresponde à taxa em que a entrada de dados pode ser assimilada pelo ambiente mediado;
- A amplitude da interatividade, correspondente ao número de possibilidades de ação a cada momento;
- E o mapeamento, que vem a ser a habilidade de o sistema mapear seus controles em face das modificações no ambiente mediado de forma natural e previsível [14].

Segundo KOOGAN & HOUAISS[15]:

"A interatividade é a troca entre o usuário e um sistema computacional por meio de um terminal dotado de tela de visualização".

Assim sendo, o usuário pode interferir alterando a forma e o conteúdo do ambiente interagido em tempo real.

Existem algumas classificações para a interatividade, e a que a agrupada em três grupos, definindo que nível de interatividade existe num determinado sistema ou aplicação é bastante interessante, são eles:

2.3.1. Interatividade local (Local Interactive)

Interatividade que acontece apenas a nível local, nenhuma informação é enviada para o exterior, não há, portanto utilização do canal de interatividade.

São aplicações que podem ser residentes ou não residentes. O EPG é um típico exemplo de aplicação de interatividade local, residente, já um exemplo de uma não residente, um bom exemplo seria uma aplicação que é um jogo, localizado em um canal, sintonizado pelo usuário.

2.3.2. Interatividade Udirecinal (One-Way Interactive)

Aplicações desse tipo são normalmente não residentes e se caracterizam pela interatividade ser propagada em apenas um sentido. Uma aplicação meteorológica que sempre é atualizada pelo provedor de conteúdo e o usuário acessa ela através de uma canal, por exemplo, é um tipo de aplicação de interatividade que se propaga no sentido provedor de conteúdo - usuário, e não residente.

2.3.3. Interatividade Bidirecinal (Two-Way Interactive)

Nesse tipo de aplicações a interação ocorre nos dois sentidos. As interações do usuário na aplicação serão propagadas ao provedor de conteúdo, e esse por sua vez, enviará uma resposta à interação prévia.

Hoje em dia aplicações desse tipo não são comuns, uma vez necessitaria de uma aplicação mais elaborada, que enviasse e estivesse preparada para interpretar as informações recebidas, além do provedor de conteúdo que deveria está preparado para enviar diferentes informações para diferentes usuários que estejam executando a aplicação.

2.4. As Aplicações Interativas

Além da classificação da interatividade existem também algumas classificações para os tipos de aplicações interativas.

A classificação de aplicações interativas pode ser dada quanto a linguagem ou paradigma em que elas estão escritas, e também podem ser agrupadas dependendo de em que parte do *middleware* ela funcionará.

Quanto aos paradigmas em que estão escritas, as aplicações interativas podem ser agrupadas, atualmente, em dois grupos, as escritas no paradigma declarativo ou no paradigma imperativo. O detalhamento de cada um desses dois grupos é feito nas posteriores seções.

2.4.1. Tipos de Aplicações Interativas

2.4.1.1. Aplicações Declarativas

São aplicações construídas em linguagens mais intuitivas, de mais alto nível e por isso, mais fáceis de serem desenvolvidas. Para serem implementadas é preciso que o programador forneça apenas o conjunto de ações que devem ser realizadas, e não se preocupe como as ações devem ser feitas.

A linguagem enfatiza a declaração descritiva de um problema ao invés de sua decomposição em implementações algorítmicas, por isso, normalmente não necessita de tantas linhas de código para implementar certa tarefa e por isso são normalmente aplicações de um tempo de desenvolvimento menor que uma aplicação imperativa, talvez por apresentar uma complexidade menor, ou por ser mais entendível a olhos menos treinados.

Como exemplo de linguagens utilizadas em aplicações desse tipo, temos: o XML e o HTML ou variações dessas duas linguagens, como o NCL.

2.4.1.2. Aplicações Imperativas

São aplicações em que o programador possui um maior controle do código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa, em decorrência da existência de fluxos condicionais de controle e laços, por exemplo.

São aplicações desenvolvidas normalmente em linguagens Java, C, C++ ou que se baseiam em alguma dessas linguagens através de frameworks.

Analisando alguns *middlewares* existentes no mercado, chega-se ao cenário de utilização dos paradigmas de linguagem, demonstrados na Tabela 1.

Mercado	Modelo	Middleware	Linguagem	Paradigma
Aberto	ATSC	DASE	Java	Procedural
	DVB	MHP	Java	Procedural
	ISDB	ARIB	BML	Declarativo
	ISDTV/ SBTVD	GINGA	NCL/LUA + Java	Declarativo + Procedural
Fechado	OpenTV	OpenTV	C	Procedural
	Thomson	SN	Flash	Procedural
	MHP	MHP	HTML + Java	Declarativo + Procedural

Tabela 1 – Comparativo entre os Sistemas de TV Digital

Na Tabela 1são descritos se o sistema de TV é fechado ou aberto, qual o modelo do sistema digital, qual o *middleware* que faz uso, bem como as linguagens que podem ser utilizadas para implementar aplicações para esses sistemas e que paradigmas fazem uso.

Já quanto à localização na arquitetura do *middleware*, ou seja, onde a aplicação irá executar, as aplicações podem ser agrupadas em:

2.4.1.3. Aplicações Nativas

São aplicações tipicamente procedurais, localizadas no *middleware*, possuindo total integração com todos os componentes existentes no sistema, e possibilidade de acesso a componentes apenas disponíveis para aplicações localizadas neste nível estrutural.

São aplicações que persistem durante todo o tempo de vida do STB.

2.4.1.4. Aplicações Não-Nativas

2.4.1.4.1. As Aplicações Residentes

São aplicações que tanto podem ser desenvolvidas sobre a interface do SO, quanto sobre o *middleware*. São enviadas para o STB e armazenadas permanentemente ou por um espaço de tempo determinado.

2.4.1.4.2. Aplicações Não-Residentes

São aplicações que podem ser procedurais ou declarativas, enviadas por um TS. Possuem acesso limitado aos componentes do *middleware* e limitações na manipulação de dados são constantes nessa estrutura pela limitação no acesso a mídias de armazenamento.

Estruturalmente está localizada um nível acima do que se encontra o *middleware* e tem, via de regra, seu período de vida determinado pelo tempo de permanência em um canal.

A Figura 5, contida na referência[10], com modificações, mostra com clareza a localização e com quem se comunicam cada um dos três conjuntos de aplicações, no *middleware*.

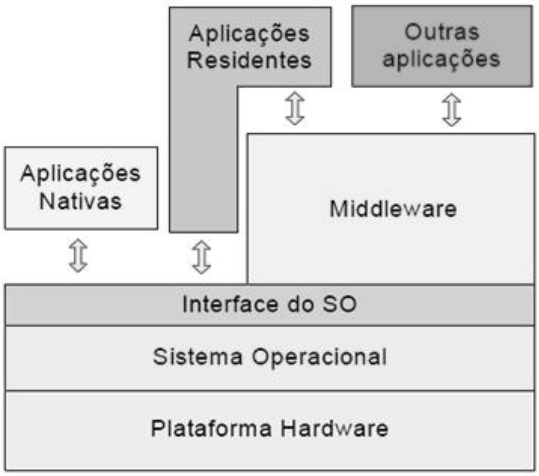

Figura 5 – Localização das Aplicações na Arquitetura de TV Digital

2.4.2. Evolução das Aplicações Interativas

A evolução das aplicações interativas certamente se dará quando primeiramente os STBs forem mais robustos e preparados para executar melhor as aplicações interativas, quando for pensada uma forma de interagir melhor do que os controles remotos utilizados hoje em dia, e quando os provedores de conteúdo estiverem dispostos a disponibilizar ao usuário a funcionalidade de interação total com o conteúdo disponibilizado.

Sendo esses itens solucionados provavelmente haverá a massificação dos VODs e a disponibilização de conteúdos, não apenas vídeo, através do canal de interatividade, a alavancada da venda de produtos através do *t-commerce* e a integração entre outras mídias e a TV.

Falar em evolução das aplicações em TV Digital passa-se pela personalização do conteúdo disponibilizado ao usuário.
Esse momento será atingido quando houver uma integração completa entre as aplicações e a ciência de contexto.

2.5. A Ciência de Contexto

A ciência de contexto é motivada da seguinte maneira por Weiser [8]:
"(...) para que se consiga obter uma computação pervasiva, é preciso aplicações que adaptem seu comportamento baseando-se na informação adquirida do ambiente físico e computacional."

Já segundo Dey [7]:
"Contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar, ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o usuário e a aplicação em si".

Sendo umas das definições mais utilizadas, essa última, torna mais fácil a caracterização de um contexto em aplicação. Uma vez que um pedaço de informação pode ser utilizado para definir ou caracterizar uma interação do usuário com a aplicação, então essa informação é contexto.

Abowd & Mynatt sugerem cinco dimensões para especificação e modelagem de informações de contexto [11]:

Dimensão	Descrição
Who (identificação)	Quem realiza uma determinada atividade, quem pode alterar o contexto ou que pode ser notificado caso o contexto seja alterado.
Where (localização)	Onde o contexto está. Esta e uma das dimensões mais usadas devido ao grande interesse de sistemas baseados em localização.
When (tempo)	A informação temporal para determinar quanto tempo uma entidade está dentro de um contexto.
What (atividade)	O que o usuário está fazendo neste momento.
Why (intenção)	Determinar o porquê o usuário está realizando determinada atividade.

Tabela 2 – Variáveis de Contexto

As classificações e a definição as dimensões de funcionamento da ciência de contexto expostas, bem como, as definições sobre a TV digital, sobre aplicações interativas e a sobre interatividade servem para margear a construção da aplicação proposta por esse trabalho.

As definições dos padrões utilizados na aplicação, além da justificativa da utilização de determinado conceito serão expostos no próximo capítulo.

3. Capítulo

O eColect

3.1. Introdução

Das diversas classificações e explicações sobre ciência de contexto e computação pervasiva expostas, e também existentes na literatura, a aplicação desenvolvida pode ser agrupada seguindo as seguintes categorias e subcategorias, descritas na Tabela 3 [2].

Categoria	Subcategoria	O porquê desta categorização?
Persistência	Temporal	Pois as informações contextuais mudam com o tempo.
Evolução	Dinâmico	O contexto muda muito rápido, como um canal sintonizado no momento.
Meio	Físico	Contextos com informações tangíveis.
Relevância para a aplicação ou serviço	Necessária	Para que uma aplicação ou serviço possa ser executado o conjunto de informações contextuais a serem obtidas é obrigatório.
Situação Temporal	Passado e Presente	As informações de contexto do passado, que têm como finalidade formar uma história de contextos de um usuário e o contexto neste exato instante.
Interação (Entre uma aplicação ciente de contexto e a parte do sistema responsável para obter o contexto)	Contexto *Push* e Contexto *Pull*	O sistema responsável por fornecer contexto envia periodicamente contexto para a aplicação e a aplicação tem de explicitamente requisitar ao sistema as informações de contexto.

Tabela 3 – Classificação do eColect quanto à Ciência de Contexto

O desenvolvimento de aplicações que apresentem comportamentos diferentes para diferentes situações ou entradas do usuário, confere a ela uma característica de ser mais realista e fiel ao usuário. Determinações se um determinado menu estará disponível ou quais informações serão mostradas na aplicação podem ser algumas das aplicações de ciência de contexto a aplicações interativas.

3.2. ARQUITETURA ADOTADA

Existem algumas formas de realizar a coleta de informações de um usuário.

A adotada foi ser desenvolvida apenas tendo acesso às informações de uma interface disponibilizada pelo *middleware*. Nesta proposta a aplicação está, levando-se em conta a arquitetura do sistema de software do STB, em um nível acima ao do *middleware*. Abordagem mais comumente utilizada hoje em dias por operadoras de TV a cabo, para utilização de aplicações interativas, por ser um modelo mais barato, uma vez que não depende do modelo do fabricante do STB para vir a ser produzida e integrada ao *middleware*.

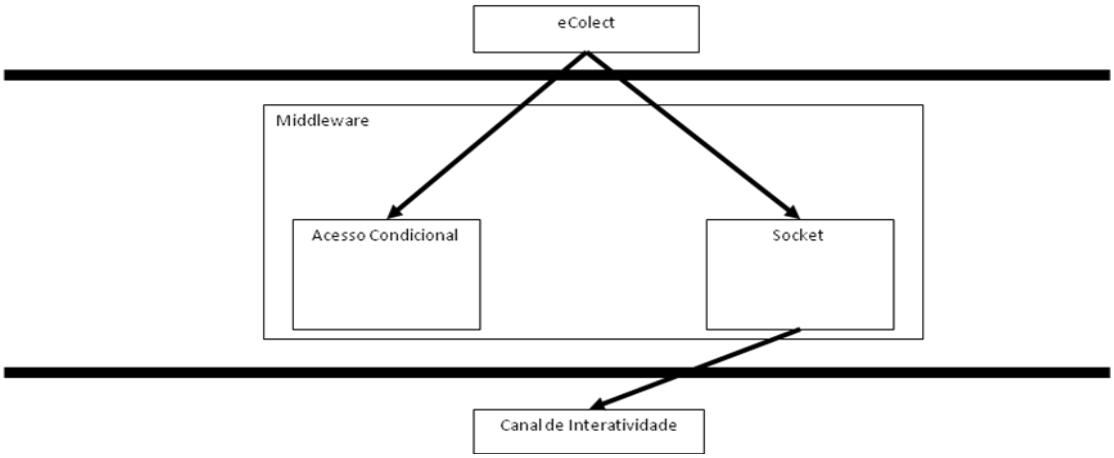

Figura 6 – Arquitetura proposta para localização do eColect no sistema de TV Digital.

3.3. O PADRÃO DE MIDDLEWARE ADOTADO

O padrão de *middleware* adotado para a construção da aplicação foi o Opentv[9].

A escolha foi baseada nos seguintes aspectos:

1. Por mais de 4 anos trabalhei no CESAR, desenvolvimento de aplicações para TV Digital, grande parte desses anos foi utilizando o padrão Opentv;
2. Por possuir grande familiaridade, acumulada ao longe de minha vida acadêmica e profissional, com a linguagem C;
3. Por acreditar que o ambiente, levando-se em conta a arquitetura e o modelo de negócio das TVs a cabo é o ideal para o funcionamento do eColetc, pois é um ambiente controlado, tendo a operadora de TV a cabo de agregador e controlador dos conteúdos disponibilizados aos usuários;
4. A abrangência mundial da utilização do middleware escolhido, uma vez que ele encontra-se implantando em mais de 100 milhões *set-top-boxes* e televisões[6] no mundo.

Caracterizada por uma linguagem baseada em ANSI C, o Opentv é um framework que redefine diversas funções definidas pela linguagem C, e que tem como controle de fluxos o "levantamento" de callbacks sua maior peculiaridade.

A aplicação Opentv[3] se caracteriza pela existência de 3 módulos obrigatórios e 1 módulo opcional, que juntos, foram uma aplicação propriamente dita, são eles:

1. Módulo de Código;
   Também chamado de módulo core ou módulo OTV, é o módulo que contém todos os arquivos ".c" da aplicação.

2. Módulo de Dados.
   Chamado também de módulo de recursos estáticos ou módulo RC, é o módulo que contém os arquivos de áudio ou vídeo que poderem está presentes na aplicação.

3. Módulo de Diretório;
   Contém um mapeamento entre todos os módulos existentes na aplicação, bem como as configurações de cada um desses módulos, propriedades como a prioridade de envio dos módulo é definida aqui.

Já o módulo opcional:

4. Módulo de Recursos;
   Chamado também de módulo de recursos dinâmicos ou módulo RES, é responsável por contém os dados necessários para atualização dinâmica.

O primeiro passo para construção de uma aplicação OPENTV, tendo configurado os módulos que serão necessários para construção da aplicação, é a definição da função *main*:

```
1 void main() {
2         icon * logo;
3         icon_resource  logo_contents;
4         logo_contents.ppixmap = &otvlogo;
5         logo_contents.x = LOGO_XPOS;
6         logo_contents.y = LOGO_YPOS;
7         logo_contents.selectable = FALSE;
8
9         init_icon_class(icon_resource_ID);
10        logo = (icon *) O_gadget_new(icon_resource_ID ,otvlogo_icon_ID, &logo_contents);
11
12        O_palette_set(closed_screen_colors, PALETTE_RGB_MODEL, 0, 16);
13        O_background_fill(0);
14
15        O_palette_set_transparency(MAX_TRANSPARENCY);
16        O_ui_set_root(logo);
17        O_gadget_activate(logo);
18        O_debug ("Entering main loop\n");
19
20        for (;;) {
21                o_message msg;
22                O_ui_get_event_wait (&msg);
23                O_debug ("Got an event: class %d, type %d.\n", O_msg_class(&msg), O_msg_type(&msg));
24                if (( O_msg_class(&msg) == MSG_CLASS_CONTROL) && (O_msg_type(&msg) == MSG_TYPE_QUIT )) {
25                        O_debug ("Got a quit event.\n");
26                        O_exit ();
27                }
28        }
29 }
30
```

Figura 7 – Exemplo da função main()

Nesse exemplo, retirado do conjunto de códigos exemplos disponibilizado pela OPENTV, vemos uma das características mais importante na programação utilizando esse padrão, que é a existência de uma estrutura de controle de mensagens, simbolizada pelo lanço infinito *for(;;)*. Essas mensagens abrangem tanto o tratamento de eventos do usuário, como a comunicação entre componentes do *middleware*.

Outro item também importante é a necessidade da criação de *gadgets*, que são os componentes básicos da programação em OPENTV, defino no exemplo acima pela função *O_gadget_new(...),* que cria um *gadget*, que representa um ícone na aplicação.

3.4. Funcionamento Básico

Ela funciona, basicamente, coletando algumas informações definidas, armazenando-as no *smartcard* do usuário e posteriormente enviando, no momento agendado, a um servidor de contexto.

3.4.1. Alguns itens que deverão ser atacados pela aplicação

Nesta seção são descrito uma lista de dificuldades e problemas técnicos, que são normalmente encontrados em aplicações cientes de contexto, e como na aplicação eColect se propõe a solucioná-la.

3.4.1.1. Separação de Responsabilidades

A obtenção dos dados será através uma aplicação que coletará informações definidas e as enviará para um servidor de contexto, que fará a manipulação dessas informações, para melhorar o serviço que é oferecido ao usuário, por exemplo.

3.4.1.2. Interpretação de Contexto

Os dados obtidos pela aplicação serão enviados ao servidor de contexto e lá serão interpretados ou combinados com informações coletadas de outras fontes com o intuito de converter os dados básicos em informações de alto nível, relevantes e importantes.

3.4.1.3. Disponibilidade do Tempo de Aquisição de Contexto

Admitindo-se que a aplicação irá rodar em todos canais disponíveis para o usuário, a aplicação está em execução sempre que o STB estiver ligado e sintonizado.

3.4.1.4. Armazenamento e História de Contextos

O armazenamento vai se dá em duas etapas:

A primeira, localmente, gravada no *smartcard* do usuário e irá conter uma história recente das informações coletadas;

A segunda etapa se dará remotamente, gravada no servidor de contexto. Nesse servidor serão gravadas uma grande quantidade de dados, de todos os usuários que fizerem uso da aplicação.

Em relação ao problema de armazenamento que pode existir nessas duas etapas, a solução proposta é apresentada nos itens abaixo.

3.4.1.5. Armazenamento Etapa 1

Na primeira etapa as informações são enviadas sempre que for atingido 80% do percentual de armazenamento do *smartcard* do cliente ou quando um determinado espaço de tempo for atingido, ou seja, ao se passar uma semana os dados são obrigatoriamente enviados ao servidor, mesmo que os 80% do percentual de armazenamento não tenham sido atingido.

3.4.1.6. Armazenamento Etapa 2

Sobre o armazenamento na segunda etapa, uma abordagem semelhante, da proposta por Andrino, em sua tese de mestrado [5], na qual fala da simplicidade dos dados armazenados e do somatório dos bytes armazenados por informações semelhantes às coletadas pela aplicação proposta por este trabalho e chega ao valor de aproximadamente 35.000KB de necessidade de evolução de armazenamento ao ano, para que todas informações coletadas sejam persistidas de forma satisfatória.

3.4.2. Protocolo de Interação entre Usuário-Aplicação

O funcionamento *front-end* da aplicação pode de forma visual ser mostrado pela Figura 8, nela é mostrado que a aplicação eColect aplicação tem como alvo as seguintes informações, canais sintonizados, quem é a pessoa que está manipulando a aplicação, o tempo em que a interação está sendo feita e em que lugar se dá essa interação.

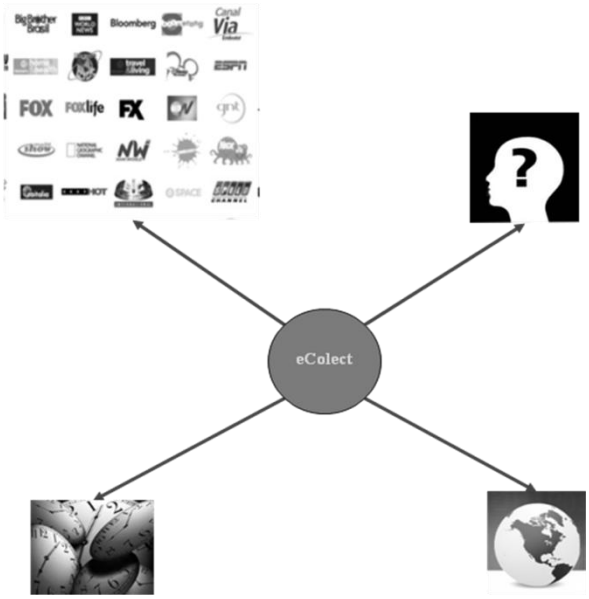

Figura 8 – Variáveis coletáveis pelo eColect

A lista de informações que são os alvos da aplicação:

- O canal de origem e destino;
- Data e hora que o canal foi sintonizado;
- Tempo de permanência no canal sintonizado;
- Quem foi o usuário;
- Onde esse usuário está localizado.

Sendo algumas dessas informações obtidas através da associação de outras, como é o caso da localização do usuário que é obtida através da determinação de quem é o usuário, através de seu número de *smartcard*.

3.4.3. Protocolo de Interação entre Aplicação-Servidor

O funcionamento *back-end* da aplicação, ou seja, como ela funciona quando coleta os dados do usuário, é descrito pela Figura 9.

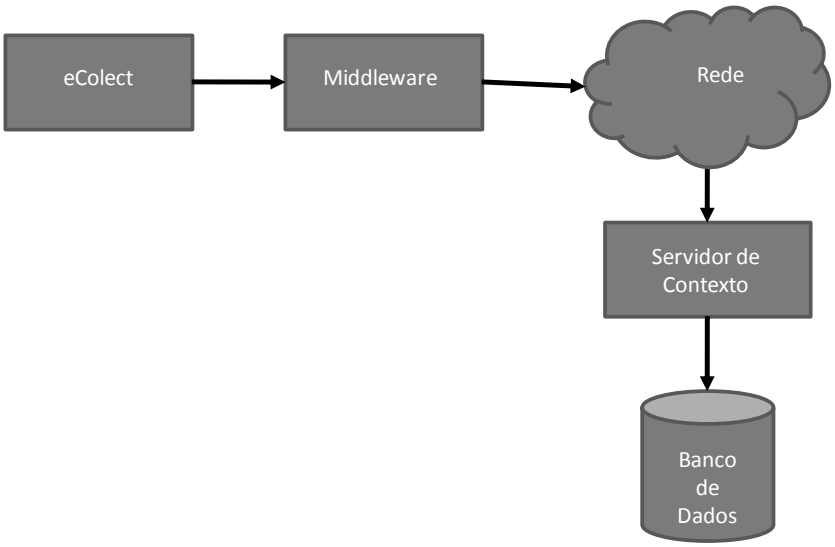

Figura 9 – Arquitetura back-end do eColect

O caminho dos dados coletados até serem armazenados no Banco de Dados do Servidor de Contexto, exposto na Figura 9, pode ser textualizado da seguinte maneira:

1. Assim que são coletados, os dados são armazenados em variáveis globais, definidas na própria aplicação;

2. As variáveis são agrupadas em uma string e preparadas para serem adicionadas ao arquivo localizado no *smarcard* do usuário, nesse momento há a camada de funções de leitura e escrita do *smartcard*, ao *middleware*;

3. Uma vez que as informações estão gravadas no *smartcard*, elas serão enviadas ao Servidor de Contexto, nesse momento, a aplicação faz chamada a funções de conexão, como criação de um *socket*, dar-se feito o envio após essa etapa;

4. Uma vez que os dados foram enviados pelo *middleware*, eles trafegam pela rede até encontrar a porta e o endereço ip descritos pelo *socket*;

5. Tendo os dados chegado ao Servidor de Contexto esse irá receber o arquivo formatado pela aplicação eColect;

6. Após realizar uma retirada das informações contidas no arquivo enviado, o Servidor de Contexto irá persistir em seu Banco de Dados as informações.

3.5. O cenário de execução

1. O usuário sintoniza um canal desejado, um canal de notícias, por exemplo;

2. A aplicação é automaticamente carregada, pois ela está atrelada ao TS do canal sintonizado, e configurada para ser inicializada automaticamente, assim que o canal é sintonizado;

3. Nesse momento a aplicação começa a coletar as informações desejadas, como por exemplo, o canal de origem, o canal que se está no momento, o tempo de permanência;

4. O usuário pressiona o botão para mudança de canal, pois o jogo de futebol de seu time está por começar;

5. A aplicação armazena as informações no *smartcard* do usuário;

6. A aplicação é descarregada pelo gerenciador de aplicações do *middleware*;

7. O *middleware* sintoniza, com ajuda do *Tunner*, o novo canal;

8. Uma nova instância da aplicação, atrelada ao novo canal, é carregada;

9. As novas informações do usuário são coletadas pela aplicação;

10. O usuário permanece durante um bom tempo assistindo a partida de futebol do seu time;

11. A aplicação faz a análise e percebe que o tamanho das informações chegou ao limite previsto para envio ao servidor;

12. Após o tempo de vida do arquivo, podendo assim chamar, as informações serão enviadas ao servidor de contexto. Esse tempo de vida determinará a quantidade de informações coletadas ou o número de dias, uma semana, por exemplo, que essas informações foram capturadas.

3.6. Possíveis Utilizações dos Dados Coeletados

Uma das possíveis arquiteturas, que demonstra a utilização dos dados coletados pode ser vista na Figura 10.

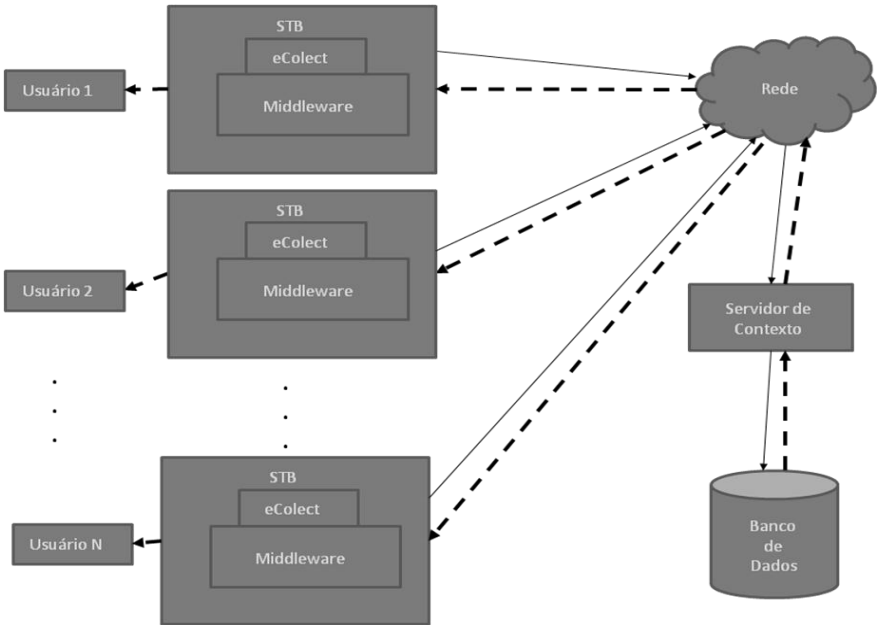

Figura 10 – Arquitetura back-end, visão bidirecional

Na Figura 10 são mostrados, além do fluxo de dados através da interação do usuário(mostrado pelas linhas cheias), um possível fluxo de dados (mostrado pelas linhas tracejadas) desde o banco de dados contendo as informações coletadas dos mais diversos usuários e "voltando" para o usuário, como por exemplo, com a distribuição de um novo canal de esportes.

Mesmo não sendo o foco deste trabalho, o tratamento dos dados coletados ou a definição de que ações tomar com a utilização deles pode ser descrito imaginando os seguintes fatos:

- Os usuários de determinada região Sul gostam mais do canal X;
- O canal Y não está tendo grande aceitação pelos usuários da região Nordeste;
- Diversos canais PPV estão sendo comprados, mas não estão sendo assistidos pelos clientes.

E com esses padrões procurar soluções para melhorar a interação do usuário com a TV, como nos exemplos citados:

- Focar ações comerciais para a região Sul, no canal X;
- Substituir o canal Y por um novo canal para usuários da região Nordeste;

- Avisar por meio de e-mails ou por aviso em outros canais através de uma aplicação interativa que determinado canal ou evento PPV, comprado, irá começar.

As ações e aplicações que podem ser feitas com os dados coletados são infinitas, o valor agregado na utilização dessas informações, para o provedor de conteúdo é uma determinação fiel do que os seus clientes estão achando da programação disponibilizada a eles. E para os clientes o benefício é que haverá uma melhoria nos serviços prestados a ele, como novos canais e novas aplicações interativas que facilitarão sua vida, enquanto assiste TV.

Tendo a Ciência de Contexto em sua essência, melhorar as informações que são disponibilizadas a quem faz uso dela. A adequação a variáveis do tipo: "a quem"," onde", "o que", "quando" e "por que" é inerente a aplicações que visão ser cientes de contexto.

A aplicação aqui proposta irá tentar atacar essas variáveis. Coletar dados que possam servir como ferramenta na determinação de melhores conteúdos disponibilizados aos usuários da TV Digital.

4. Capítulo

Implementação do sistema

4.1. Módulos da Aplicação

A aplicação constitui por 2 módulos, um de código ou OTV e outro de diretório ou DIR.

O módulo de código foi dividido em 3 módulos:

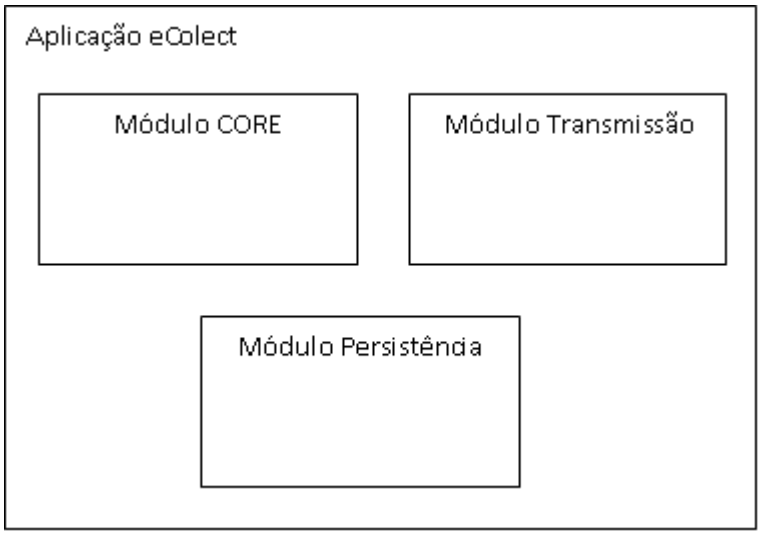

Figura 11 – Módulos do eColect

1. Módulo Principal ou Core:

-Constitui o core da aplicação, contendo as funções de captura de informações e de data e hora.

2. Módulo Persistência:

-Agrupa o grupo de funções relacionadas à gravação das informações no *smartcard* do cliente.

-A verificação da necessidade de envio ou não ao servidor, do arquivo gravado no *smartcard*.

3. Módulo Transmissão:

-Contém as funções necessárias para o estabelecimento da conexão com o servidor e posterior envio das informações coletadas.

Como definido, no Capítulo 3, deste trabalho, os seguintes itens a serem coletados:

- Canal de origem;
- Canal que a aplicação carregar, ou seja, o canal onde está;
- Hora que o canal foi sintonizado;
- Tempo de permanência no canal.

Além, de informações que são obtidas através do número de smartcard do cliente, como:

- Localização do cliente;
- Pacotes assinados pelo cliente.

A implementação, além seguir do documento de requisitos descrito no Apêndice A, seguiu o seguinte roteiro de soluções de perguntas abaixo:

1. Onde e Como guardar as informações coletadas?

   As informações são coletadas e armazenadas no smartcard. E o formato de armazenamento será o seguinte:
   "numeroSmartcard-eConlect.txt"

   O conteúdo do arquivo será composto de entradas, correspondentes a linhas que seguirão o seguinte formato:
   [aaaammddhhmmss].[canal de origem].[canal que está a aplicação carregou].[tempo de permanência no canal]

2. Com que frequência as informações serão coletadas?

   Sempre que o usuário fizer algumas das ações descritas na lista de ações coletáveis, a aplicação irá coletar informações. Não havendo tempo determinado para a finalização da coleta.

3. Com que frequência as informações serão enviadas ao servidor?

   Fazendo uma pré-análise do tamanho disponível para armazenamento no *smartcard*, tamanho máximo permitido, pelo fabricante do *smartcard*, a ser utilizado por uma aplicação, a quantidade de informações pode crescer de forma vertiginosa, com "zapping", por exemplo.

   Levando-se em conta isso quando o tamanho do arquivo atingir o 80% do máximo permitido, vai ser dá início ao processo de envio do arquivo ao servidor de contexto.

Esse 80% foi definido levando-se conta o conjunto de informações que ainda pode está sendo inserido no arquivo e o tempo para se estabelecer uma conexão.

Em caso de insucesso de envio ou o tempo de envio demore mais que o tempo para atingir os 100% do tamanho definido para o arquivo, as informações mais antigas dentro do arquivo serão sobrescritas.

Baseando-se em Andrino[5], que fala sobre o tamanho das informações armazenadas e analisa o crescimento do tamanho dos dados armazenados e chega ao valor de aproximadamente de 1KB para as *features* armazenadas, e levando-se em conta que a primeira o primeiro passo de armazenamento se dará no *smartcard* do cliente.

O tamanho necessário para armazenar os dados localmente gira em torno de 1KB.

4. Como vai ser feito o envio?

   O envio é feito por uma abertura de *socket* e posterior conexão com o servidor.

5. O que acontecerá depois do envio?

   Após o envio, o arquivo será recriado e as informações já enviadas sobrescritas.

Dos três módulos descritos na Figura 11, os mais importantes para definir o funcionamento da aplicação é o Módulo Core e o Módulo Persistência.

4.2. O Módulo Core

No Módulo Core, a função *main*, a primeira função a ser executada de toda a aplicação e todo o fluxo de controle de mensagens de callbacks, entre os diversos componentes do Opentv, está presente no módulo core.

Também nesse módulo se dá o tratamento dos eventos do usuário, através do mapeamento das teclas do controle remoto que importam para a aplicação.

Como já foi definido anteriormente, no Capítulo 3, que informaçôes devem ser capturadas, então para conseguir esses dados é necessário mapear as seguintes teclas no controle remoto:

1) Canal para cima;
2) Canal para baixo;
3) Canal anterior;
4) Todas as teclas numéricas.

O mapeamento dessas teclas no código do eColect, bem como, a chamada da função responsável por chamar o módulo de persistência é mostrado na Figura 12

```
168
169         for (;;) {
170                 o_message msg;
171                 O_ui_get_event_wait (&msg);
172                 switch (O_msg_class(&msg)) {
173                 case MSG_TYPE_KEY_UP:
174                 case MSG_TYPE_KEY_REPEAT: {
175                         O_debug("MSG_TYPE_KEY_UP ou MSG_TYPE_KEY_REPEAT\n");
176                         switch (msg->INFO_KEY) {
177                                 /*Teclas Movimentação entre canais*/
178                                   /*Teclas Numéricas*/
179                                 case 0x0030:
180                                 case 0x0031:
181                                 case 0x0032:
182                                 case 0x0033:
183                                 case 0x0034:
184                                 case 0x0035:
185                                 case 0x0036:
186                                 case 0x0037:
187                                 case 0x0038:
188                                 case 0x0039: {
189                                      acumularNumerosDigitados(msg->INFO_KEY);
190                                      getNextChannelNumber();
191                                      preparaDadosColetados();
192                                      dadosGravados = 1;
193                                      sendOrNotFile(persistence)
194                                      break;
195                                 }
196                                 case KEY_CHANNEL_DOWN_CODE:
197                                 case KEY_CHANNEL_UP_CODE:
198                                 case KEY_PREVIOUS_CODE: {
199                                      getNextChannelNumber();
200                                      preparaDadosColetados();
201                                      dadosGravados = 1;
202                                      sendOrNotFile(persistence)
203                                      break;
204                                 }
```

Figura 12 – Fluxo de Controle de mensagens

As funções responsáveis por coletar a data, hora, minutos e segundos, bem como, o timer que contabiliza o tempo de permanência no canal, podem ser vistas na Figura 13.

```
86 /*Pegar a DDMMHHSS em que o canal foi sintonizado*/
87 void  getDHMSinicial() {
88         O_debug("getDHMSinicial\n");
89         o_tm local_time;
90         o_time hora;
91
92         O_debug("getDHMSinicial\n");
93
94
95         O_system_time_get(&hora);
96         O_time_to_local_tm(&hora, &local_time);
97         O_debug("Peguei a hora e a data do STB\n");
98
99         O_sprintf(os_data_inicio, "%02d%02d%d%02d%02d%02d", local_time.tm_mday,
100                         local_time.tm_mon + 1, local_time.tm_year + 1900,
101                         local_time.tm_hour, local_time.tm_min, local_time.tm_sec);
102
103
104 }
105
106 /*Acumalar o tempo de permanencia em um determinado canal*/
107 void verificarTimer(thread* timer) {
108         O_debug("verificarTimer\n");
109
110        if (dadosGravados == 1) {
111            thread_stop(threadP);
112
113        } else {
114               tempoPemanencia++;
115               thread_start(threadP);
116        }
```

Figura 13 – Funções de coleta de tempo

A coleta dos canais anterior e atual é feita na *main()*, através de chamadas a funções internas do *middleware*. Ver Figura 14.

```
154 void main() {
155         O_debug("\n\n ---Início da aplicação eColect--- \n\n");
156
157         setModoGrafico();
158         detectarModeloSTB();
159
160         getDHMSinicial();
161         threadP = thread_new(1000);
162         thread_add_runnable(threadP, verificarTimer);
163         thread_start(threadP);
164         canalAnterior = getLastChannelNumber();
165         canalAtual = getCurrentChannelNumber();
166
167         Persistence* persistence = persistenceGetInstance();
168
```

Figura 14 – Coleta dos Canais Anterior e Atual

Também o número do smartcard do usuário é obtido utilizando chamadas internas do middleware, como visto na Figura 15.

```
80 void getSmartCardNumber() {
81         nSmartcard = getFromAcesseControlID();
82         O_sprintf(fileName, "%s-eColect.txt", nSmartcard);
83 }
```

Figura 15 – Acesso ao número do smartcard

No Módulo Core, também há a concentração das chamadas às funções que constrõem a conexão com o servidor e as funções que verifiricam o status dos dados armazenados, como pode-se ver na Figura 16

```
120 void preparaDadosColetados() {
121
122         char* strEnvio = O_calloc(400, sizeof(char));
123
124         StringBuilder* sb = stringBuilderGetInstance();
125         Persistence* persistence = persistenceGetInstance();
126
127         O_debug("preparaDadosColetados\n");
128
129         if (sb == NULL) {
130                 O_debug("StringBuilder == NULL\n");
131                 return;
132         }
133
134         if (pacote == NULL) {
135                 O_debug("persistence == NULL\n");
136                 return;
137         }
138
139         stringBuilderGetStrEnvio(sb, strEnvio);
140         /*-------------------------------------------------------*/
141
142         persistenceWriteFile(persistencia);
143         /*-------------------------------------------------------*/
144
145         O_free(sb);
146         O_free(persistence);
147
148         O_free(strEnvio);
149
150         O_debug("Fim de envioDadosColetados\n");
151 }
152
```

Figura 16 – Chamada de funções do Módulo Persistence

4.3. O Módulo Persistence

Já no Módulo Persistence, existem as funções que determinam a criação, o uso e a necessidade de envio do arquivo de dados gravados.

Nas Figura 17 e Figura 18, é demonstrada a chamada de funções de criação do arquivo que irá persistir os dados e a avaliação da necessidade de envio ou não dos dados, respectivamente.

```
37 void persistenceWriteFile(Persistence* this) {
38         PathType pathType = PRODUCER;
39         o_date expirationDate = UNDEFINED_DATE;
40         o_mode permission = OTV_WORLD_READ_WRITE_PERMISSION;
41         File* file = fileNew(pathType, fileName, expirationDate, permission);
42 
43         O_debug("persistenceWriteFile\n");
44         if (this == NULL)  {
45                 O_debug("persistence == NULL\n");
46         } else {
47                 O_debug("strEnvio = %s\n", this->strEnvio);
48                  /*porta e servidor definidos no mod transmition*/
49 
50                 O_sprintf(dadosColetados, "[%s].[%d].[%d].[%d]", os_data_inicio, canalAnterior, canalAtual, tempoPeman
51                 O_debug("****sizeof(Persistence) = %d\n", sizeof(Persistence));
52                 if (file != NULL) {
53                         if (fileWrite(file, this, sizeof(Persistence))) {
54                                 O_debug("persistenceWriteFile: Escreveu no arquivo.\n");
55                         } else {
56                                 O_debug("persistenceWriteFile: Não Escreveu no arquivo.\n");
57                         }
58                 } else {
59                         O_debug("persistenceWriteFile: Não criou o arquivo.\n");
60                 }
61         }
62 }
63 
```

Figura 17 – Função de escrita de dados

```
64 bool sendOrNotFile(Persistence* this) {
65     char* fileName = "eColect.txt";
66     int tArquivo;
67     tArquivo =  getFileSize(fileName);
68     ret = FALSE;
69     tArquivo = tArquivo /100;
70 
71     if (tArquivo > TAMANHO_MAX_ARQUIVO)
72        ret = TRUE;
73 
74     return ret;
75 }
```

Figura 18 – Função de verificação de envio dos dados

4.4. A Execução do eColect

A execução da aplicação é demonstrada no diagrama de estados, Figura 19.

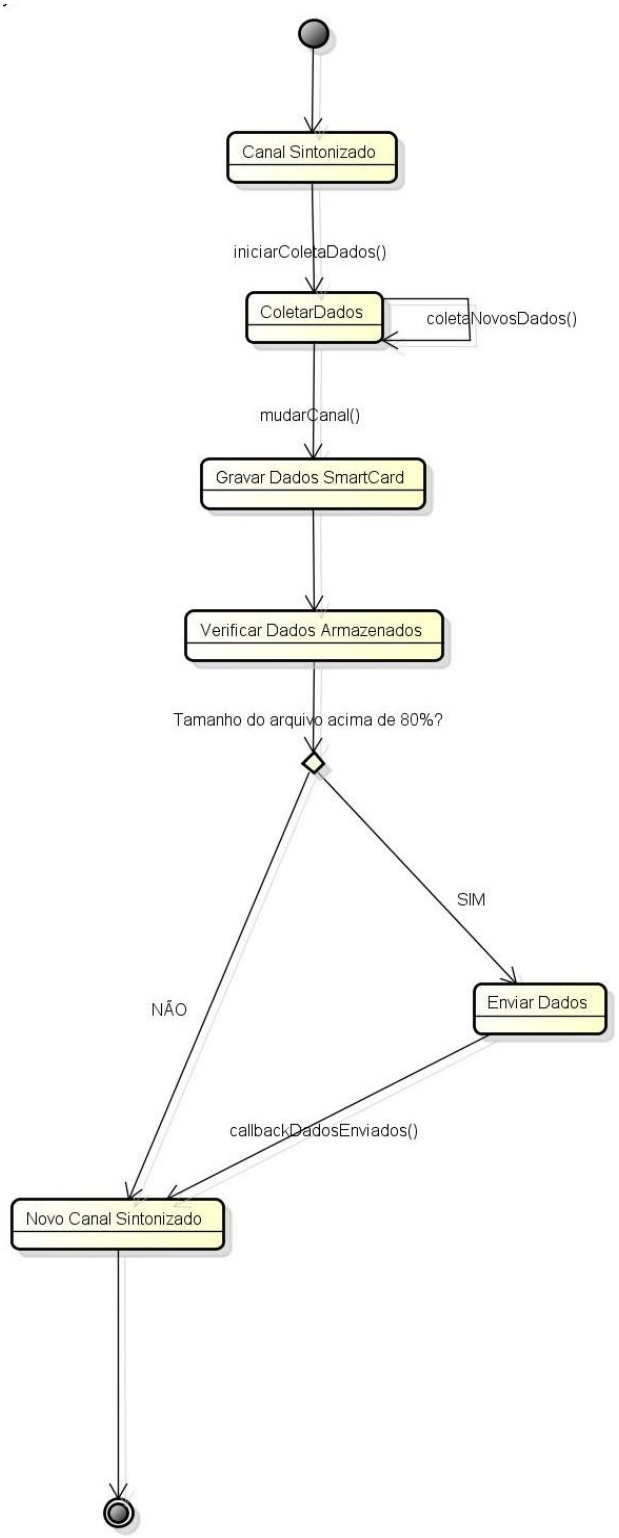

Figura 19 – Diagrama de Estados

Durante a interação usário com a aplicação podem ocorrer dois cenários, definidos pela quantidade de informações coleadas e exportos na Tabela 4.

Quantidade máxima de informações coletadas foi atingida?		Envio dos dados ao Servidor de Contexto?
SIM	➔	NÃO
NÃO	➔	SIM

Tabela 4 – Relação envio x quantidade de dados armazenados

O diagrama de sequência, na Figura 20 descreve como é o funcionamento da aplicação, quando é feita uma coleta de dados e não é feito o envio ao servidor.

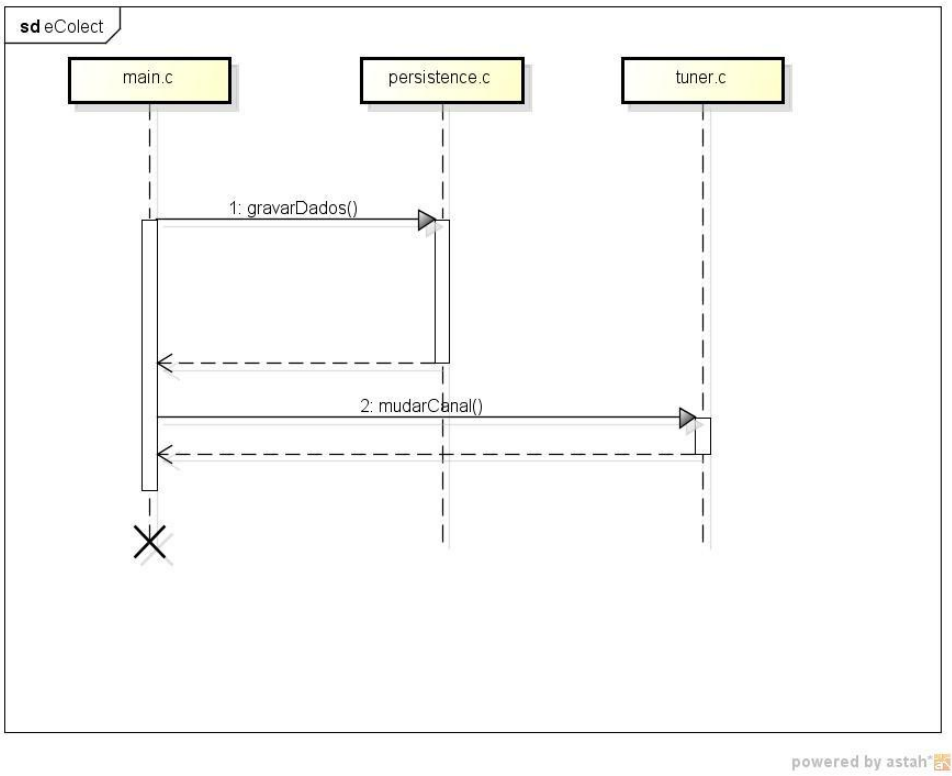

Figura 20 – Diagrama de Sequência, envio não realizado

Já o diagrama descrito na Figura 21 descreve o caso em que o tamanho limite, do arquivo que contém as informações coletadas atingiu seu limite e porntanto será enviado ao servidor.

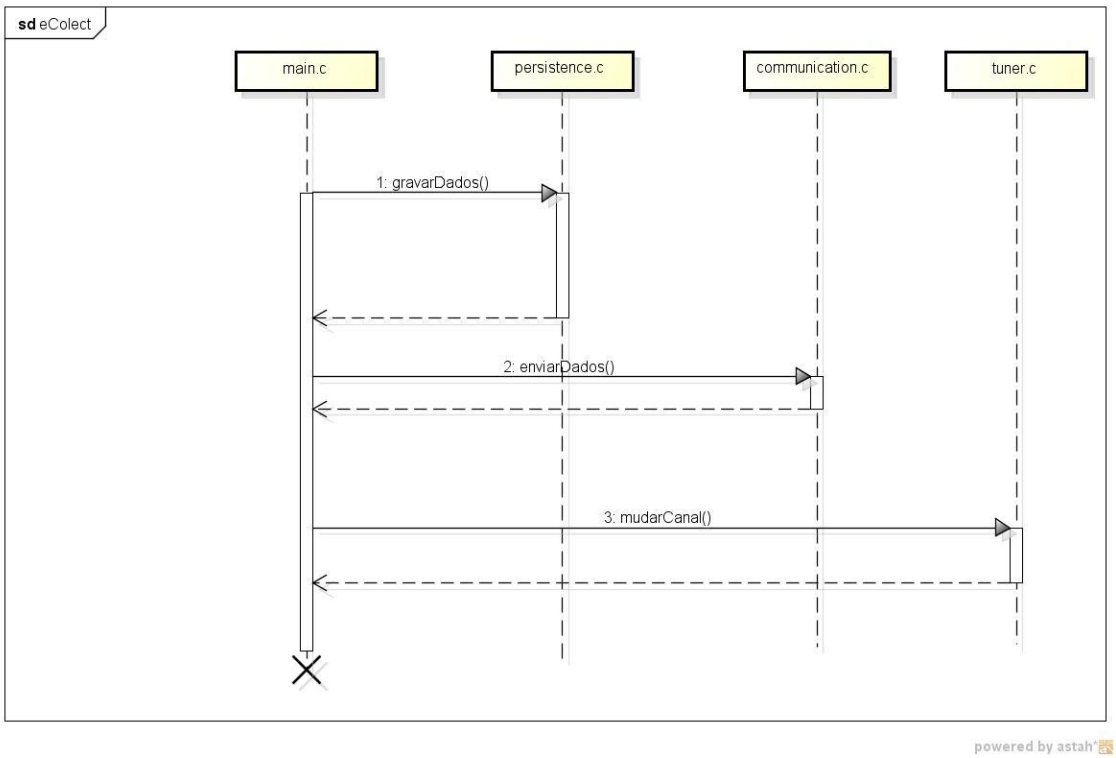

Figura 21 – Diagrama de Sequência, envio realizado

5. Capítulo

Conclusão

Este trabalho não tem como finalidade avaliar se informações pessoais devem ou não ser utilizadas para cunho comercial. A existência da prerrogativa do controle de privacidade, tão em volga atualmente, é válida. No entanto, a constante melhoria do conteúdo disponibilizado ao usuário deve ser tida como o objetivo maior.

Fato que deve ser lembrado é o que existe na internet onde grande maioria das informações dos sites navegados é capturada, processada e informações personalizadas são enviadas ao usuário que vistiou esses sites.

A ciência de contexto foi desenvolvida com a finalidade básica de conferir personalização ao conteúdo que se tem acesso, e tendo essa descrição como linha principal, é que o eColect foi concebido.

Não pretendendo em momento algum solucionar todos os problemas que ainda existem quanto à contextualização das informações, mas vislumbrando ajudar na solução desses problemas, e expor que é possível obter informações confiáveis e de forma fácil no ambiente da TV Digital.

Trabalhos Futuros

O desenvolver uma solução parecida ao eColect, para o ambiente de TV Digital aberta, como, por exemplo, a utilização do GINGA, seria um grande desafio, uma vez que não é um ambiente de apenas um concentrador e distribuidor de conteúdo.

Outro trabalho de grande importância seria o desenvolvimento de um servidor de contexto, que sem dúvida alguma seria o mais trabalhoso, mas o importante em toda solução.

Um trabalho, também bastante interessante seria determinar formas de utilizar os dados coletados para:

i) Melhoria na programação oferecida aos usuários;
ii) Potencializar a utlização da interatividade na TV através:
    (a) Da melhoria da usabilidade das aplicações interativas desenvolvidas;
    (b) Desenvolvimento de aplicações que atendam às necessidades do usuário.

Com certeza essa determinação das formas de utilização dos dados coletados serião trabalhos bastante produtivos, comercialmente falando.

6. Referências

[1] Valdecir Becker, Carlos Piccioni, Carlos Montez, Günter H. Herweg Filho.
Datacasting e Desenvolvimento de Serviços e Aplicações para TV Digital Interativa

[2] Paulo Martinelli Hemmlepp.
Aplicações de TV Digital sensíveis a contexto para dispositivos móveis.
Trabalho de Graduação

[3] *OpenTV Software Developer's Kit API Q2-2003.* August 2003

[4] *Interactive TV Web*
Disponível em http:// www.interactivetvweb.org,
Acessado em 10 de novembro de 2010.

[5] Andrino Soares de Souza Coêlho.
ProfileTV: Um Sistema de Gerenciamento de Perfis em TVDi,
Dissertação de Mestrado,Cin-UFPE 2008

[6] Research and Market. *OpenTV – Market Research Reports*. Disponível em http://www.broadbandtvnews.com/2008/02/20/opentv-found-in-100-million-enabled-devices/, acessado em 23 setembro de 2010.

[7] Dey, Anind K.; Abowd , Gregory D.
Towards a Better Understanding of Context and Context-Awareness.
1st International Symposium on Handheld and Ubiquitous Computing, 1999.

[8] Weiser, Mark.
The Computer for the Twenty-First Century. Scientific
American, 1991.

[9] *OpenTV*
Disponível em *www.opentv.com*
Acessado em 20 novembro 2010

[10] EduardoRodrigues de Carvalho
Uma plataforma modular para testes com interatividade na TV Digital Brasileira
Dissertação de Mestrado, USP 2008

[11] Abowd, G.D.; Mynatt, E.D.
Charting Past, Present and Future Research in Ubiquitous Computing.
ACM Transactions on Computer-Human Interaction, Vol 7,
No. 1, Março de 2000.

[12] GINGA,
Disponível em *www.ginga.org.br/*
Acessado em 10 Dezembro 2010

[13] SBTVD,
Disponível em *www.forumsbtvd.org.br/*
Acessado em 10 Dezembro 2010
[14] Steuer, J.
Defining virtual reality: dimensions determining telepresence.
Journal of Communication: Autumn 1992
[15] KOOGAN & HOUAISS, 1999
Dicionário

Apêndice A:

Documento de Requisitos

Restrições e Premissas

[RP001] Disponibilidade de banda

O eColetc necessitará de uma largura de banda adequada para que possa executar suas funcionalidades de forma satisfatória. Esta largura de banda deverá ser determinada por testes feitos pelo provedor de conteúdo ou operadora de TV a cabo.

[RP002] Protocolo de comunicação entre STB e Server

O protocolo de comunicação entre o eColect e o servidor de aplicações é de responsabilidade do setor de TI da operadora de TV. O mesmo tem de ser definido e disponibilizado previamente, antes da fase de implementação do eColect.

Nesta definição do protocolo deverão ser acordadas:

A formatação da string de envio de dados, da aplicação eColect com o Servidor de Contexto;

[RP003] Definição do IP e da porta de comunicação com servidor

A definição do IP bem como da porta de comunicação com o servidor devem ser definidos pela operadora de TV e disponibilizados à equipe de desenvolvimento.

[RP004] Definição do conjunto de dados enviados pela aplicação eColect ao servidor de Contexto

Foi definido que o conjunto de dados enviados ao Servidor de Contexto é composto pelos:

Dados coletados do usuário da aplicação:

Data e Hora da coleta – Deverá ser coletado seguindo o seguinte formato: "AAAAMMDDHHMMSS";

Canal Origem – Deverá ser coletado o canal anterior ao que está sintonizado;

Canal Atual – Deverá ser coletado o canal atualmente sintonizado;

Tempo de permanência no canal – Deverá ser coletado o tempo de permanência no canal sintonizado.

E pelos dados que a aplicação recolherá do STB do cliente:

Número do smartcard.

O formato final dos dados coletados, ou seja, cada linha dentro do arquivo que será enviado ao servidor seguirá o seguinte padrão:

[aaaammddhhmmss].[canal de origem].[canal que está a aplicação carregou].[tempo de permanência no canal]

E o formato do arquivo enviado ao Servidor de Contexto, seguirá o seguinte padrão: "numeroSmartcard-eColect.txt"

[RP005] Definição das Teclas Capturadas pela aplicação eColect

A definição das teclas que a aplicação eColect irá mapear foi:

As teclas:

Canal para cima;
Canal para baixo;
Canal anterior;
Números.

Requisitos Funcionais

[RF001] Capturar a data, hora, minutos e segundos que um canal é sintonizado (Essencial)

A aplicação deve ser capaz de capturar a data, hora, minutos e segundos.

[RF002] Capturar o canal anterior ao sintonizado pelo usuário (Essencial)

A aplicação deve ser capaz de capturar o canal de anterior ao atualmente sintonizado pelo usuário.

[RF003] Capturar o canal atual que o usuário está sintonizado (Essencial)

A aplicação deve ser capaz de capturar o canal atual que o usuário encontra-se.

[RF004] Leitura do número do SmartCard (Essencial)

A aplicação deve ser capaz de ler o número do smartcard do cliente.

[RF005] Criar um arquivo no SmartCard (Essencial)

A aplicação deve ser capaz de criar um arquivo no formato definido na seção Restrições e Premissas e salvá-lo no smartcard do cliente.

[RF006] Conectar-se ao Servidor de Contexto (Essencial)

A aplicação deve ser capaz de estabelecer uma conexão com o Servidor de Contexto através de um socket, utilizando o endereço IP e a porta, definidos pela operadora de TV por assinatura.

[RF007] Envio dos dados ao servidor (Essencial)

A aplicação deve ser capaz de enviar ao Servidor de Contexto, os dados coletados pela aplicação.

[RF008] Tentar reenvio dos dados (Essencial)

A aplicação deve possuir a função de reenviar os dados coletados, caso haja erro na conexão com o servidor.

[RF009] Sobrescrever um arquivo no SmartCard (Essencial)

A aplicação deve ser capaz de sobrescrever um arquivo no smartcard do cliente, quando esse já tiver sido enviado ao Servidor de Contexto.

Requisitos não funcionais

[RNF001] Tempo de carregamento da aplicação

O tempo de carregamento da aplicação será afetado pela disponibilidade da largura de banda necessária.

O tempo de carregamento dessa aplicação não deve superar 3 segundos.
A análise do tempo de carregamento de cada conteúdo assim como alocação de banda será feita a partir da necessidade da precisão dos dados coletados.

[RNF002] Performance de execução

Após ser carregada, a aplicação não terá apresentação visual nenhuma para o usuário.

O tempo de troca de, de canais não será impactado uma vez que a aplicação é pequena, ou seja, de rápido carregamento e descarregamento, esse requisito deve ser considerado por corresponder a um fator de qualidade de software.

Escopo Não Contemplado

A aplicação não possibilita atualização automática de conteúdo.
Sua finalidade é enviar os dados elecitados nos requisitos funcionais e apenas eles, ao servidor de Contexto.
O desenvolvimento do Servidor de Contexto, não faz parte da aplicação eColect.