FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Novos serviços interativos para televisão digital

Diogo Alves de Sousa Castro

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Maria Teresa Andrade

23 de junho de 2014

Novos serviços interativos para televisão digital

Diogo Alves de Sousa Castro

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Rui Carlos Camacho de Sousa Ferreira da Silva (Prof. Associado)
Arguente: Jose Manuel de Castro Torres (Prof. Associado)
Orientadora: Maria Teresa Magalhaes da Silva Pinto de Andrade (Prof. Auxiliar)

---

23 de junho de 2014

Resumo

Esta dissertação decorre da conclusão do Mestrado Integrado em Engenharia Informática e Computaçãoe foi proposta pela empresa MOG Solutions. Pretende-se com este trabalho desenvolver formas que possibilitem produtores de conteúdo audiovisual introduzirem conteúdos interativos complementares, recorrendo a uma camada de anotação de dados e a *software* para *Smart TVs*.

A televisão tem vindo a perder importância relativa devido, principalmente, a dispositivos alternativos de consumo de conteúdos com acesso à Internet, como computadores, *tablets* ou *smartphones*. Estes dispositivos possibilitam formas de interação com os conteúdos que não são possíveis com o modelo atual de televisão, levando a uma perda de espetadores, que se traduz em perda de receita e em decréscimo da qualidade do produto final. Embora se tenha tentado desenvolver serviços complementares, estas soluções necessitavam de equipamento adicional e eram de difícil implementação e sincronização com o sinal linear.

Pretende-se com esta dissertação conceber e desenvolver uma solução de interatividade, no contexto do consumo de conteúdo televisivo linear em *Smart TVs*, que evite a complexidade e o problema de sincronismo referido. Para atingir este objetivo, um número de desafios necessita de ser considerado, nomeadamente:

- Identificação do tipo de interatividade que deve ser oferecido ao utilizador;
- Identificação do tipo de metadados e como representá-los;
- Definição de como adicionar estes metadados adicionais ao sinal linear de TV;
- Especificação de uma camada de software para tratar os metadados do lado do cliente e da forma de envio e execução nesse mesmo lado;
- Criação de um protótipo que demonstre as possibilidades das soluções propostas, simplificando e permitindo a interação dos consumidores com o conteúdo adicional.

Como propostas de implementação da solução, sugere-se a utilização de uma extensão do MPEG-7 para anotação de conteúdos e a utilização da norma HbbTV®para envio de uma aplicação genérica para a *Smart TV* do cliente. Esta aplicação desenvolvida em HTML terá a função de receber as anotações, processá-las e convertê-las numa interface usável pelo utilizador. Os elementos adicionais, tais como imagens, tanto deverão poder ser descarregados da Internet, como virem embebidos na aplicação.

Como caso de uso demonstrativo são fornecidas informações complementares a um anúncio publicitário e é introduzida essa mesma publicidade num momento temporal específico de um programa de televisão. Para avaliar o cumprimento dos objetivos é avaliado se a informação complementar chega ao cliente, se é apresentada de acordo com o especificado nos metadados e se está sincronizada com a programação linear.

Abstract

This dissertation is associated with the conclusion of the MSc in Informatics and Computing Engineering and it was proposed by the company MOG Solutions. The aim of this work is to develop new tools to allow producers of audiovisual content to insert additional interactive content in Smart TVs, using a layer of metadata and software.

The television has been losing relative importance, mainly due to alternative content consumption devices with Internet access, like computers, tablets or smartphones. These devices allow new ways of interaction with content that are not possible with the current television model, leading to a loss of viewers that translates into loss of revenue and decrease in the quality of the final product. The development of complementary services has been tried, but these solutions required additional equipment and revealed to be difficult to implement and synchronize with the linear signal.

The aim of this dissertation is to design and develop a solution of interactivity in the context of linear television consumption in Smart TVs. The solution must also avoid complexity and synchronization problem referred. To achieve this goal, a number of challenges need to be addressed notably:

- Identification of the type of interactivity that should be offered to the user;
- The type of metadata required and how to represent it;
- How to add this additional metadata to the linear TV signal;
- Specification of a software layer to handle the metadata at the client side and how to deliver it to and run it in the client;
- Creation of a prototype to demonstrate the possibilities of the proposed solutions, streamlining the TV enhanced content to the consumers and allowing them to interact with it.

We suggest the use of an extension to MPEG-7 to annotate the content and the HbbTV® standard to send a generic HTML application to client's Smart TV. This application will receive, process and convert the metadata to an interface usable by the user. The additional assets, like images, can be downloaded from the Internet or can be embedded with the application.

As a demonstrative use case, complementary information to an ad is showed, and the same ad is associated with a specific temporal moment of a television program. To assess the fulfilment of the objectives, it is verified if the complementary information reaches the client, if it is presented according the specified in the metadata and if it is synchronized with the linear programming.

Agradecimentos

Não poderia concluir esta dissertação sem lembrar e agradecer a todos aqueles que me ajudaram a tornar possível a conclusão da mesma.

Desta forma, devo começar por prestar o meu profundo agradecimento à Prof.ª Maria Teresa Andrade, minha orientadora. Além de ter sido uma guia e de ter sido incansável no apoio prestado, foi alguém com quem pude debater abertamente ideias, levando-me, assim, a atingir todos os meus objetivos.

Gostaria de agradecer à MOG, nas pessoas do Eng.º Pedro Ferreira e Eng.º Alexandre Ulisses, pela oportunidade que me proporcionaram. A experiência de trabalho num ambiente empresarial, bem como todo o apoio e orientação foram essenciais para o meu crescimento e para que me mantivesse no caminho certo.

Não posso, jamais, esquecer os colegas que me acompanharam durante os 5 anos deste curso e que, com grande paciência me foram aturando. Aos meus amigos próximos, que desde sempre me acompanham e que se mantiveram presentes e disponíveis, particularmente neste último e complicado ano, o meu muito obrigado. A sua amizade nesta fase da minha vida em muito contribuiu para que hoje aqui esteja.

Aos meus pais e a toda a minha família, por todas as palavras de apoio, oportunidades que me criaram e orientação para procurar outras mais. Sei bem o valor das lições que me dão todos os dias. Sem eles nunca seria a pessoa que hoje sou.

Por fim, ao meu irmão. Apesar da nem sempre fácil relação entre nós, gostaria de lhe dedicar um agradecimento especial. Sempre presente e disponível, a sua influência e ajuda têm marcado toda a minha vida positivamente, mesmo que só agora o reconheça e agradeça. Obrigado, Ricardo.

Bem hajam.

Diogo Castro

Conteúdo

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

Abreviaturas e Símbolos

2D	Duas dimensões
A/V	Audiovisual
AIT	Application Information Table
API	Application Programming Interface
BBC	British Broadcasting Corporation
D	Descriptor
DDl	Data definition language
DI	Digital Item
DID	Digital Item Declaration
DS	Description Scheme
DSM-CC	Digital Storage Media Command and Control
DTV	Digital Television
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
egta	European Group of Television Advertising
EIT	Event Information Table
EPG	Electronic Program Guide
FIFO	First In First Out
HbbTV®	Hybrid Broadcast Broadband TV
HTML	HyperText Markup Language
IP	Internet Protocol
IPTC	International Press Telecommunications Council
JSON	JavaScript Object Notation
MPEG	Moving Picture Experts Group
MXF	Material eXchange Format
NIT	Network Information Table
PAT	Program Association Table
PCR	Program Clock Reference
PMT	Program Map Table
PSI	Program-Specific Information
PVR	Personal Video Recorder
SDT	Service Description Table
SMTPE	Society of Motion Picture and Television
STC	System Time Clock
TAL	TV Application Layer
TV	Televisão
UDP	User Datagram Protocol
VTR	Video tape recorder
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Capítulo 1

Introdução

A interatividade, no contexto do consumo de televisão radiodifundida, começou a ser introduzida ainda na época analógica. Estas experiências implicavam, tipicamente, a utilização de meios complementares como o telefone. Com a chegada das tecnologias digitais começou-se a tentar fundir o consumo de conteúdos audiovisuais radiodifundidos (TV), com o acesso a recursos associados, por via da Internet, recorrendo, por exemplo, à partilha de vídeo, áudio e texto em tempo real, sincronizados com a emissão, ou à disponibilização de informação contextual. No entanto, as soluções desenvolvidas careciam de simplicidade e implicavam o emprego de *hardware* complementar específico, geralmente caro, como *set-top boxes*. Em termos técnicos, estas soluções eram de difícil implementação e manutenção, devido à necessidade de sincronismo entre todos os elementos, além de requererem infraestruturas próprias para armazenamento/transmissão dos diferentes tipos de conteúdo.

1.1 Contexto/Enquadramento

Esta Dissertação surge no âmbito da conclusão do Mestrado Integrado em Engenharia Informática e Computação, da Faculdade de Engenharia da Universidade do Porto. O corrente problema foi proposto pela empresa MOG Solutions, especializada em produtos e soluções de produção e transmissão em ambientes baseados em ficheiros. Esta acompanhará o desenvolvimento da dissertação, com orientação da Professora Maria Teresa Andrade, da Faculdade de Engenharia da Universidade do Porto.

1.2 Motivação e Objetivos

A televisão, enquanto dispositivo, e meio, de acesso a informação e entretenimento, tem vindo a perder importância relativa devido, principalmente, ao aparecimento de dispositivos e formas

alternativas de acesso a esses bens e serviços. Os novos dispositivos de acesso, que surgem tipicamente associados ao contexto Internet, são os computadores, *tablets* e *smartphones*. Estes permitem uma interação com os conteúdos, ou mesmo com os seus contextos de produção, que não é possível obter com o modelo atual de televisão. Esta perda de importância afeta assim os produtores, que assistem a uma fuga de espetadores, o que se traduz numa perda de receita publicitária e, como tal, numa perda de capacidade, quantitativa e qualitativa, de produção de conteúdos. Por seu lado, o decréscimo de qualidade da programação oferecida significa que também os espetadores acabam prejudicados.

Além disto, e como veremos a seguir, os serviços interativos disponibilizados são geralmente fornecidos pelos operadores de cabo ou, mais recentemente, pelas estações emissoras, deixando de fora a possibilidade de produtores de conteúdos fornecerem os seus serviços complementares diretamente ao consumidor final.

Nesta dissertação pretende-se conceber e desenvolver uma solução de interatividade, no contexto do consumo de conteúdo televisivo síncrono[1] em dispositivos *Smart TV*, que evite a complexidade (necessidade de emprego de *hardware* dedicado, por exemplo), e problemas de sincronismo das soluções anteriores. Adicionalmente, deverão ser os produtores de conteúdos a especificar a informação complementar. Para atingir este objetivo geral será necessário:

1. Identificar o tipo de interatividade a disponibilizar;
2. Definir o formato de uma camada de metadados;
3. Definir o método de introdução dos metadados no sinal difundido;
4. Introduzir uma camada de software em *Smart TVs* que trate os metadados;
5. Facilitar a produção dos metadados.

Ou seja, o primeiro passo será identificar o tipo específico de interatividade que irá ser disponibilizada aos utilizadores, e definir um caso de uso da mesma. Esta clarificação permitirá delimitar o perímetro da solução a desenvolver e implementar.

Em termos técnicos, a solução a desenvolver deverá comportar a definição do formato e do método de introdução de uma camada de metadados no sinal difundido. Esta camada destina-se a ser empregue em *Smart TVs*, para permitir a almejada interatividade. Este será mesmo o principal objetivo técnico desta dissertação.

Os dispositivos *Smart TV* permitem o acesso à Internet e possuem sistemas operativos avançados passíveis de serem programados. A solução a desenvolver deverá assim comportar a introdução de uma camada de software nestes dispositivos que explore os metadados, acima referidos, para a adição de interatividade à experiência de consumo dos espetadores. Essa interatividade implicará, tipicamente, a possibilidade do utilizador interagir com dados ou conteúdos adicionais, de alguma forma relacionados com o conteúdo síncrono principal, que são obtidos de fontes *on-line* ou do sinal radiodifundido.

[1] Conteúdo audiovisual transmitido em tempo real por uma estação de televisão.

Por fim, para facilitar a produção de conteúdos passíveis de serem consumidos interativamente, pretende-se que a aplicação desenvolvida forneça ferramentas que facilitem a criação de conteúdos.

Esta dissertação pode ser enquadrada cientificamente nas áreas de anotação de conteúdos, de interação do utilizador e visualização avançada ou na distribuição de conteúdo. Estes assuntos estão geralmente associados às publicações IEEE Transactions on Broadcasting, Springer Multimedia Tools and Applications e no SMPTE Journal. Também é relevante a conferência TVX - ACM International Conference on Interactive Experiences for TV and Online Video.

1.3 Estrutura da Dissertação

Para além desta introdução, este relatório contém mais 3 capítulos. O capítulo 2, inicia com um enquadramento da televisão digital e dos serviços interativos disponibilizados atualmente, seguindo-se a descrição do estado da arte acerca de anotação de conteúdos e tecnologias para *Smart TVs*. No capítulo 3 é documentada a implementação realizada. É definido o modelo de interação a disponibilizar ao utilizador, a organização dos metadados e como funcionam e interagem os componentes da solução criada. É exemplificada a utilização da aplicação através de um caso de uso e são avaliados/validados os resultados obtidos. Este capítulo termina com uma descrição sobre a forma de integrar a solução em ambiente real. Por fim, no capítulo 4 é feita uma conclusão do relatório e é aferida a satisfação dos objetivos propostos.

Capítulo 2

Estado da Arte

Para uma melhor compreensão de alguns dos conceitos relevantes para esta dissertação, o presente capítulo irá analisar o estado da arte atual. As seguintes secções descrevem as áreas temáticas que compõem o trabalho: a televisão digital, os tipos de interatividade existentes, a anotação de conteúdos, as capacidades interativas das *Smart TVs* e as ferramentas/tecnologias relevantes para o desenvolvimento da solução e integração no mundo real. Pretende-se, com este levantamento, encontrar formas de reutilizar as ferramentas existentes, para que se possa criar um sistema que seja interoperável com aquilo que existe atualmente no mercado, seja do lado do consumo ou do lado da transmissão.

2.1 Televisão digital

Desde meados do século XIX que a televisão se tem afirmado como um dos mais importantes meios de comunicação. A palavra televisão deriva da combinação da palavra grega "*tele*"com a palavra latina "*visio*", significando visão distante. A sua definição inclui não só os equipamentos, mas os programas de televisão, os seus processos de produção e, na área das telecomunicações, a transmissão televisiva, também denominada de *broadcasting*. Nas últimas décadas a evolução tecnológica forçou a redefinição do conceito de televisão. De transmissões analógicas a digitais, da baixa à alta definição, muitos têm sido os avanços que abrem a porta a inovações.

A DTV, televisão digital, é hoje em dia a forma usada para transmissão do sinal televisivo, recorrendo à transmissão de áudio e vídeo num sinal processado digitalmente. Existem quatro normas a nível mundial [Sta14]: ISDB-T, disponível no Japão, SBTVD-T, disponível na América do Sul, DTMB na China, ATSC, na América do Norte, e DVB, disponível na Europa, e em diversos países da África, Ásia e Austrália. A figura 2.1 ilustra a distribuição destas normas a nível mundial.

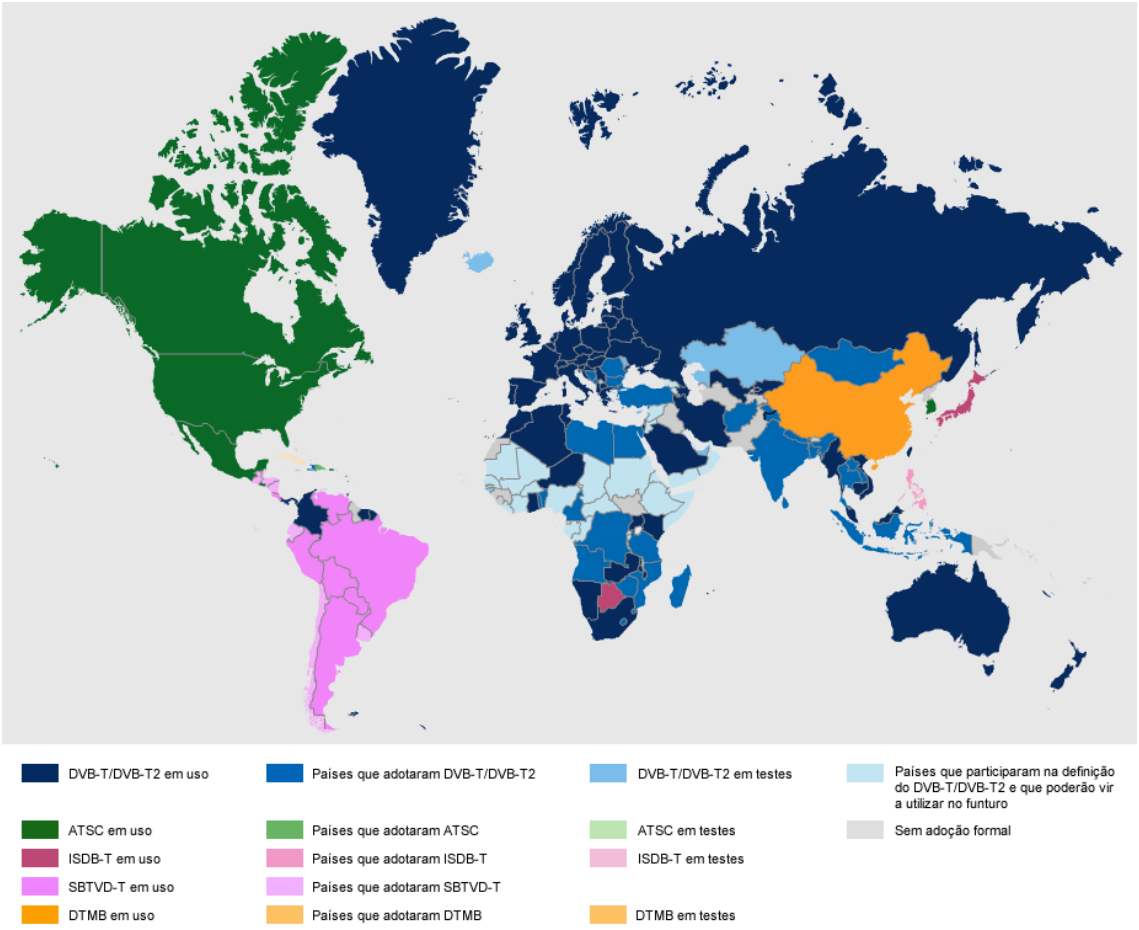

Figura 2.1: Estado de implementação da DTV a nível mundial em 2014 [Sta14, DVB14]

2.1.1 DVB

O projeto DVB nasceu de um consórcio de instituições ligadas à área de *broadcasting*, definindo várias normas para transmissão de DTV [Kra]. Uma das filosofias iniciais, que se concretizou, consistia em enviar múltiplos programas[1] (canais de televisão, por ex.) num único canal de transmissão. Entre outros, definiu três normas para transmissão: o DVB-T, para transmissão por via terrestre, o DVB-C, para transmissão por cabo, e o DVB-S, para transmissões por satélite. O envio dos dados é feito através de MPEG *Transport Streams*. Mais recentemente, desenvolveu normas para emissão em redes móveis e fixas de comunicação, como por exemplo, o DVB-IPTV.

A transmissão dos programas é acompanhada de tabelas que informam o equipamento recetor dos elementos que a compõem e como estes se relacionam. As informações contidas nestas tabelas chamam-se Informação Específica de Programa, ou PSI. Além das cinco tabelas especificadas no PSI, podem ser declaradas outras. No total, podem destacar-se as seguintes [ETS98]:

[1] Conjunto de informação (áudio, vídeo, dados, entre outros) sincronizada, ou seja, com uma referência temporal comum. Corresponde ao conceito de canal de TV usado pelas estações emissoras.

1. Tabela de Associação de Programas (PAT), a tabela fundamental do PSI. Contém os identificadores da Tabela Mapa de Programas (PMT) e da Tabela de Informação da Rede (NIT), para que o recetor as possa identificar;

2. Tabela Mapa de Programas (PMT), a tabela que descreve como um programa está organizado, identificando e indicando a localização dos *streams* elementares existentes no serviço, nomeadamente o *stream* de vídeo e áudio, a Tabela de Informação da Aplicação (AIT) e o *stream* de eventos. Indica ainda qual dos *streams* contém a referência de relógio do programa (MPEG PCR). Cada programa contém a sua própria PMT;

3. Tabela de Informação da Rede (NIT), que fornece os parâmetros da rede física;

4. Tabela de Descrição do Serviço (SDT), que contém informação descritiva dos programas, voltada para o utilizador. Por exemplo, o nome do programa (canal de TV);

5. Tabela de Informação de Eventos (EIT), que contém informação acerca dos eventos ou programas de TV da emissão, nomeadamente o nome, tempo de início e duração. Esta informação é usada para gerar o EPG;

6. Tabela de Informação da Aplicação (AIT), que contém informação acerca das aplicações existentes no sinal difundido.

2.1.2 MPEG *Transport Stream*

A norma MPEG-2 é direcionada para a transmissão de imagem e áudio de elevada qualidade via satélite, cabo ou terrestre. Cada um destes meios, devido às especificidades do seu meio de propagação, está sujeito a perturbações que podem ser causadoras de erros. O sinal de TV é transmitido como um *stream* de dados MPEG-2, denominado de *Transport Stream* (TS). Este é constituído por sub-*streams*, denominados *streams* elementares, em que cada um pode conter áudio, vídeo ou dados. Cada um destes *streams* elementares é identificado por um identificador (PID), único no contexto desse TS. O *Transport Stream* define um protocolo baseado em pacotes para transmitir os programas codificados em MPEG através das redes digitais referidas. Inclui ainda mecanismos de sincronização e correção que permitem a correta receção dos conteúdos.

Um pacote do TS (figura 2.2) tem tamanho fixo de 188 *bytes*, contendo um cabeçalho (*header*), com tamanho fixo de 4 *bytes*, uma secção de adaptação (*adaptation-field*), que é uma extensão de tamanho variável do cabeçalho, e/ou uma secção de dados (*payload*) [Yos02]. Para distinguir um pacote com secção de adaptação, de um outro com dados, ou de outro com ambos, existe um campo no cabeçalho (uma *flag* de dois *bits*), que indica qual das três hipóteses está presente. O cabeçalho possui ainda um identificador do pacote (PID) e informação temporal para sincronização. Esta última informação é inserida nos *streams* elementares que contêm os Programas, e ainda num campo do *header*, denominado Relógio de Referência do Programa (PCR) [Gro05, Yos02]. Estas informações temporais são comparadas com o relógio do descodificador, podendo assim

fazer as correções necessárias para que este fique sincronizado com o relógio do codificador, nomeadamente através de alterações à frequência de descodificação do sinal. A partir do relógio do descodificador, é efetuada a sincronização dos diferentes elementos (áudio, vídeo, ...) que compõem os programas [Gro05].

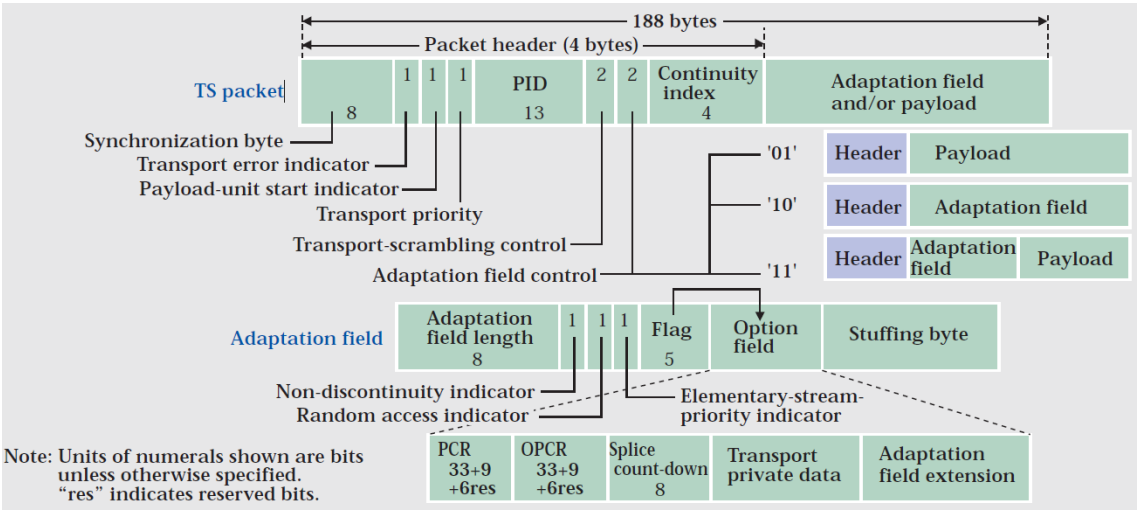

Figura 2.2: Estrutura de um pacote de TS [Yos02]

2.1.3 DSM-CC

O DSM-CC foi desenvolvido originalmente para permitir controlar máquinas presentes na industria audiovisual, como VTRs, mas foi ganhando funcionalidades ao longo do tempo [Mor11, ISO98]. Entre elas incluem-se o controlo de servidores de vídeo MPEG através de uma rede, suporte para transmissão de dados através de MPEG-2, informação temporal para vídeo MPEG-2, transmissão simples de dados e transmissão de sistemas de ficheiros. É através desta tecnologia que é possível enviar dados no DVB, além do áudio e vídeo. Estes podem ser enviados a par da emissão de um programa, ou num programa independente, usando um *stream* elementar como Objeto Carrossel. Objeto Carrossel é uma técnica de envio sucessivo, em ciclo contínuo, de uma informação para o cliente. Como não existe canal de resposta para indicar a perda de informação, evita-se essa situação voltando a enviar tudo a partir do início. Dessa forma também se evita que um cliente perca informação por não recebê-la desde o início da sua emissão. Esta forma de envio pode não ser a mais eficiente quando a quantidade de informação é elevada, mas é minorada no recetor guardando a informação já recebida em memória em vez de esperar que ela chegue toda seguida. A informação é enviada em blocos e alguns podem ser enviados mais vezes que outros, diminuindo o tempo de acesso a ficheiros que sejam usados mais frequentemente.

O facto de o DVB permitir enviar dados na transmissão abre a possibilidade de enviar aplicações interativas. O MHP é um conjunto de especificações [Kra] desenvolvidas pelo projeto DVB que permite isso mesmo, independentemente do meio de transmissão. Define a forma como o sistema operativo do terminal deve tratar a aplicação recebida e a forma como as aplicações

devem ser enviadas, nomeadamente especificando a tabela AIT que contém as características da aplicação.

O DSM-CC não é apenas usado para transmissão de dados [Mor11, ISO98]. Também permite transportar informação temporal para sincronizar o dispositivo recetor. Embora o MPEG já transporte informação temporal no PCR, o recetor pode não interpretar este valor. A informação temporal é denominada Tempo Normal de Reprodução (NPT) e pode começar com um qualquer valor, servindo apenas de referência para o excerto de média em questão. Esta informação pode conter descontinuidades, sejam elas temporalmente positivas ou negativas. Esta informação é embebida numa secção privada do MPEG-2 e informa o equipamento recetor acerca do momento da emissão que está a ser emitido. Isto é feito através de uma informação que acompanha este valor e que indica qual rácio entre si e o tempo do relógio de sistema do MPEG-2 (STC).

Como será visto aquando da sincronização temporal, além da informação temporal é possível enviar eventos para obter sincronismo entre o emissor e o recetor. Este mecanismo é conhecido como eventos de *stream* e são enviados através do DSM-CC.

2.2 Serviços Interativos

Televisão interativa é a tecnologia que permite transmitir conteúdo audiovisual a pedido para a televisão de um utilizador. Estes conteúdos podem ser escolhidos pelo utilizador de acordo com a sua preferência, ou seja, são personalizados, e podem ser transmitidos sem a obrigatoriedade de horários que a programação linear[2] obriga. Esta tecnologia define o *hardware* necessário para a transmissão, bem como a infraestrutura de suporte para a criação e transmissão até ao utilizador final [HS].

Os serviços interativos atualmente disponibilizados ao público são, de uma maneira geral (aqui excluem-se alguns casos, como veremos mais à frente), fornecidos pelos operadores de cabo. São usadas *set-top boxes* proprietárias e os conteúdos interativos são meros complementos à emissão. Ou seja, pela inexistência de um canal de comunicação bidirecional, não permitem a submissão de informação por parte do utilizador.

O tipo mais básico de conteúdos disponibilizados é a informação contextual, nomeadamente informação sobre o programa, os seus atores, apresentadores, etc., bem como legendagem e múltiplas trilhas sonoras (multi-áudio). Serviços como EPGs, estações de rádio ou informação de trânsito e tempo também são serviços que habitualmente encontramos nestas ofertas.

Também foram experimentados conceitos de sinais alternativos, como sistemas multi-câmara, vídeos relacionados com o programa ou publicidade. Com o advento das redes sociais, estas também passaram a marcar presença, bem como a disponibilização de jogos.

Na área da integração, foram efetuadas emissões de legendagem gestual introduzidas a pedido (*on-demand*), ou ainda áudio-descrição.

Muitas vezes, em vez da utilização das *boxes*, foram usados aparelhos complementares, como por exemplo a utilização do rádio para a transmissão da áudio-descrição. Também se tem assistido

[2] Idêntico a conteúdo síncrono.

à evolução do conceito de segundo ecrã, onde informação complementar é disponibilizada recorrendo a *tablets*, *smartphones* ou computadores, de forma síncrona com o conteúdo exibido no ecrã principal.

Por último, foram criados e testados conceitos de partilha de conteúdos como, por exemplo, a utilização de sinais de vídeo produzidos por dispositivos de outros utilizadores e sincronizados com a emissão [GZO+10].

2.3 Anotação de conteúdos

Os metadados, usados na anotação de conteúdos, foram criados para que seja mais eficiente a organização, acesso e interpretação dos conteúdos digitais [SS06]. No caso específico dos conteúdos multimédia, esta informação adicional é particularmente importante, uma vez que estes conteúdos não são suficientemente auto-descritivos para uma interpretação automática.

Existem diversas normas relacionadas com a anotação de conteúdos. No que à área de multimédia diz respeito, devido à extensão do domínio em que esta se insere, não foi possível criar apenas uma que respondesse a todas as necessidades.

2.3.1 SMPTE/EBU

A SMPTE define um conjunto de normas para a troca de informação audiovisual, nomeadamente o contentor MXF (*Material eXchange Format*). Este formato não especifica nenhum esquema de compressão, podendo conter conteúdo vídeo ou áudio em diferentes formatos, mantendo assim uma maior interoperabilidade com diversas aplicações e equipamentos usados ao longo da cadeia de produção de conteúdo televisivo. Todo o conteúdo audiovisual pode ser unificado num só contentor, nomeadamente diferentes camadas de vídeo, áudio, metadados ou outro tipo de ficheiro [Dev02]. É também definida uma norma denominada SMPTE S380M ou DMS-1 [SS06], que especifica os metadados que acompanham esses conteúdos. Estes metadados contêm informação acerca da produção, transmissão e distribuição, e podem estar associados a diferentes níveis de granularidade, como um ficheiro completo ou apenas a uma porção do mesmo. Atualmente a norma baseia-se num modelo de dicionário que define os campos dos elementos possíveis de usar [SFSP01]. De entre a informação usualmente encontrada nestes contentores, destaca-se aquela que identifica o formato dos ficheiros presentes e a informação temporal (*timecode*) dos mesmos. Esta informação temporal ganha particular importância quando é necessário sincronizar os diferentes conteúdos.

2.3.2 TV-Anytime

O TV-Anytime foca-se na definição de especificações para a utilização de conteúdos/serviços audiovisuais por parte do consumidor. Assenta essencialmente em três funcionalidades base: referenciação de conteúdos, metadados e gestão de direitos. A especificação de metadados é feita

em XML e é baseada na linguagem de definição de dados (DDL) do MPEG-7. No entanto, também recorre a um dicionário de definição de elementos e atributos usados nos vários esquemas [SFSP01, ETS11]. É usado especialmente para a transmissão de dados relevantes para a utilização de gravadores eletrónicos (PVRs), nomeadamente informação acerca da programação dos canais (EPGs) [SS06].

2.3.3 MPEG-7

Em contraponto com outras normas MPEG, que definem a representação de conteúdo audiovisual, o MPEG-7 foca-se na normalização de uma interface comum para a descrição de conteúdo multimédia: vídeo, imagem, som, entre outros [NL99]. Esta norma define uma camada de anotação em XML através de uma linguagem de definição de dados (DDL), que é associada ao conteúdo audiovisual, podendo estar fisicamente localizada no mesmo sinal/sistema de armazenamento, ou noutra localização independente. A informação contida nesta camada representa uma descrição detalhada desse conteúdo, referente a diferentes níveis de granularidade (região, imagem, segmento de vídeo e coleção) e a áreas diferentes (descrição de conteúdo, gestão, organização, navegação e interação com o utilizador) [SFSP01]. Esta informação é principalmente usada para gerir, pesquisar e indexar os conteúdos [NL99], mas também pode ser usada para inserir informação complementar, como legendas. Através da utilização de códigos de tempo também é possível associar a informação a momentos temporais específicos, permitindo assim a sincronização com o conteúdo audiovisual.

Vários elementos, como Descritores (D), Esquemas de Descrição (DS) e Linguagem de Definição de Dados (DDL), estão normalizados, contribuindo para uma maior interoperabilidade. Os Descritores associam uma característica a um determinado valor. Os Esquemas de Descrição são os modelos dos objetos multimédia, especificando os Descritores que podem ser usados numa determinada descrição, e quais as relações entre os mesmos ou com outros esquemas de descrição. A Linguagem de Definição de Dados especifica as regras sintáticas para combinar Descritores e Esquemas de Descrição. O MPEG-7 estende o esquema do XML, adicionando algumas extensões, nomeadamente primitivas para representação do tempo (*basicTimePoint* e *basicDuration*).

Uma descrição em MPEG-7 começa com o elemento raiz que indica se a descrição é total ou parcial (informação incremental a outra descrição). Em caso de descrição total, segue-se um elemento de alto nível que orienta a descrição para uma determinada tarefa (por exemplo, a descrição de um determinado conteúdo visual) [Sta04]. Para descrever eventos temporais, o MPEG-7 possui os Esquemas de Descrição *Time* e *MediaTime* [Sta04]. O tempo pode ser descrito de forma objetiva (fornecendo o instante), de forma relativa (em relação a outro ponto) ou como um número de intervalos (unidade de tempo previamente especificada). Estas representações derivam dos Descritores *BasicTimePoint* e *BasicDurantionTime*.

Para fazer descrições textuais, o MPEG-7 possui um conjunto de campos estruturados para orientar alguns aspetos que podem ser relevantes para a descrição. Estes campos correspondem às perguntas “Quem?”, “Qual objeto?”, “Qual ação?”, “Onde?”, “Quando?”, “Porquê?” e “Como?”. O elemento principal desta descrição é o Esquema de Descrição *Segment*. Esta é uma classe

abstrata[3] que possui, entre outras, as subclasses[4] *VideoSegment* e *StillRegion*. Estas subclasses representam um conjunto de *frames*[5] numa sequência de vídeo e uma região espacial 2D numa imagem ou *frame*, também de uma sequência de vídeo, respetivamente [Sta04]. Cada segmento pode ser decompostos em sub-segmentos, realizando assim uma ordenação hierárquica. A representação 2D pode ser feita pelo Descritor *Spatial2DCoordinateSystem* e a representação temporal já esta incluída no Esquema de Descrição *VideoSegment*.

Uma vez que é apenas definida a forma como descrever os conteúdos e não os conteúdos em si, o MPEG-7 é independente em termos de plataforma, de tecnologia ou aplicação.

2.3.4 MPEG-21

Com o MPEG-21 tentou estender-se a interoperabilidade, definindo a forma como os elementos de uma infraestrutura de uma aplicação multimédia se relacionam, integram e interagem [BVDWH+03]. Foi criada o contentor *Digital Item* (DI), que corresponde ao encapsulamento dos recursos multimédia (MPEG-1,2,4, ou outros) e os metadados, definidos em MPEG-7, que os anotam. As relações entre estes são estabelecidas através de uma *Digital Item Declaration* (DID). Além de recursos de vídeo, também podem existir elementos de áudio, imagem, ou textuais associados, o que permite a inserção de ficheiros HTML, por exemplo.

Para além destes metadados, são ainda definidos outros tipos de informação, como a proteção e manutenção de direitos de autor (através das especificações *Rights Data Dictionary* e *Rights Expression Language*) ou a adaptabilidade dos conteúdos ao ambiente (*Digital Item Adaptation*) [BVDWH+03]. Este último componente pode ser usado para entregar informação diferenciada, a partir do mesmo conteúdo, dependendo do dispositivo que a acede (por exemplo, fornecer um vídeo com uma resolução diferente dependendo se é acedido por uma televisão ou um *smartphone*).

2.3.5 egtaMETA

Para descrever conteúdo de rádio e televisão, foi definido pela EBU, União de Estações Emissoras Europeias, uma especificação em XML de metadados descritivos e técnicos/estruturais denominada EBUCore [EBU14]. Esta especificação contém um conjunto de informação essencial para que seja possível descrever os programas, facilitando as trocas entre produtores e as estações emissoras.

A partir do EBUCore foi criada uma extensão denominada egtaMETA, que especifica a troca de material publicitário [EBU10]. Esta extensão resultou da colaboração da EBU e da egta, uma associação que comercializa os espaços comerciais de estações de televisão e rádio. Este esquema especifica um conjunto de atributos semânticos, e a sua hierarquia, organizados entre informação descritiva, informação de exploração, créditos e informação técnica de um anúncio publicitário.

[3] Uma classe genérica que providencia uma implementação por omissão, partilhada com as suas possíveis subclasses. Esta não pode ser instanciada.

[4] Classes derivadas que herdam as características das classes mãe.

[5] Um conjunto de algumas das várias imagens que compõem um vídeo.

Da informação descritiva destaca-se a informação sobre o anúncio, a música de fundo, o produto publicitado, a marca e as entidades que contribuíram para a realização do mesmo.

2.3.6 IPTC/SportsML

A IPTC, *International Press Telecommunications Council*, é um consórcio mundial de organizações ligadas à industria da imprensa e tem-se focado no desenvolvimento de normas para a troca de material noticioso [IPT14]. São exemplos, a norma NewsML, o SportsML e EventsML, todos definidos em XML.

A norma SportsML, em específico, cobre um conjunto muito alargado de desportos contendo atributos para descrever eventos, nomeadamente informação sobre a temporada, os patrocinadores, o local de realização, as equipas, estatísticas, ou ações (faltas, golos, substituições, penaltis, ...). Dentro destes atributos é possível ser mais específico, fornecendo informação sobre a capacidade e o nome do estádio, tipo de campo, estatísticas de ocupação e dados meteorológicos. Sobre as equipas é possível especificar o nome, o local onde habitualmente jogam, a cidade de origem, e os jogadores que a compõem. Nos jogadores é possível especificar a carreira, lesões que tenham sofrido, dados biográficos, peso/altura, ou estatísticas de penaltis, prémios e rankings. Possui ainda atributos específicos para cada desporto, como futebol (pontapé de saída), ténis (set, pontuações de serviço, pontuações de receção), entre outros.

2.4 Tecnologias e normas para Smart TVs

O mercado de *Smart TVs* é vasto e, apesar de algumas tentativas de criação de plataformas transversais, os fabricantes têm apostado na utilização de plataformas proprietárias. Cada plataforma tem o seu conjunto de APIs e utilizam diferentes formas de programação, desde linguagens nativas a flash. Apesar disto, as principais marcas suportam HTML5 como linguagem de programação, o que permite converter uma aplicação para as diferentes plataformas. Algumas até dispõem de APIs para a utilização de equipamentos externos ao televisor, como forma de segundo ecrã [Sam14]. No entanto, não existe uma forma nativa de aceder a metadados transmitidos no sinal de vídeo difundido.

De forma a tentar criar uma solução transversal, desenvolveram-se soluções híbridas que conjugam aplicações enviadas no sinal difundido de vídeo com o acesso à Internet. A norma HbbTV® (*Hybrid Broadcast Broadband TV*), presente em diferentes países da Europa, [ETS12] é uma delas, tal como MHEG-5, em Inglaterra, e o MHP, em Itália. Estas soluções são geralmente denominadas por "botão vermelho", uma vez que se recorre a este botão do telecomando para aceder às aplicações.

No HbbTV®, os conteúdos, dados ou aplicações adicionais são transmitidos integrados no sinal radiodifundido, mas podem também ser acedidos através da Internet ou, no caso das aplicações, estarem já nativamente instaladas nos dispositivos terminais. Uma das vantagens desta solução, comparativamente às outras plataformas, é permitir a criação de aplicações terminais empregando

o HTML, em contraste com Java[6], o que permite uma fácil adaptação de páginas e conteúdos já existentes para a Internet. A transmissão de metadados é efetuada através das especificações do *TV-Anytime* mas só está disponível em certos modelos de televisores europeus de diversas marcas, bem como *set-top boxes*. De acordo com dados da IHS [Ned14], na Europa, em 2016, mais de 50% dos televisores serão compatíveis com HbbTV®. O facto de inicialmente estar disponível em poucos países e de não ser acessível por todos os dispositivos, levou a uma adoção lenta por parte do mercado mundial. A figura 2.3 mostra os países que atualmente implementaram a HbbTV®(a verde escuro), aqueles que têm a implementação anunciada (verde claro), aqueles que têm outros planos (a laranja) e os países que não disponibilizaram informação (a cinzento). Além dos países Europeus, também outros (EUA, Japão, Austrália e China) demonstraram interesse na sua implementação.

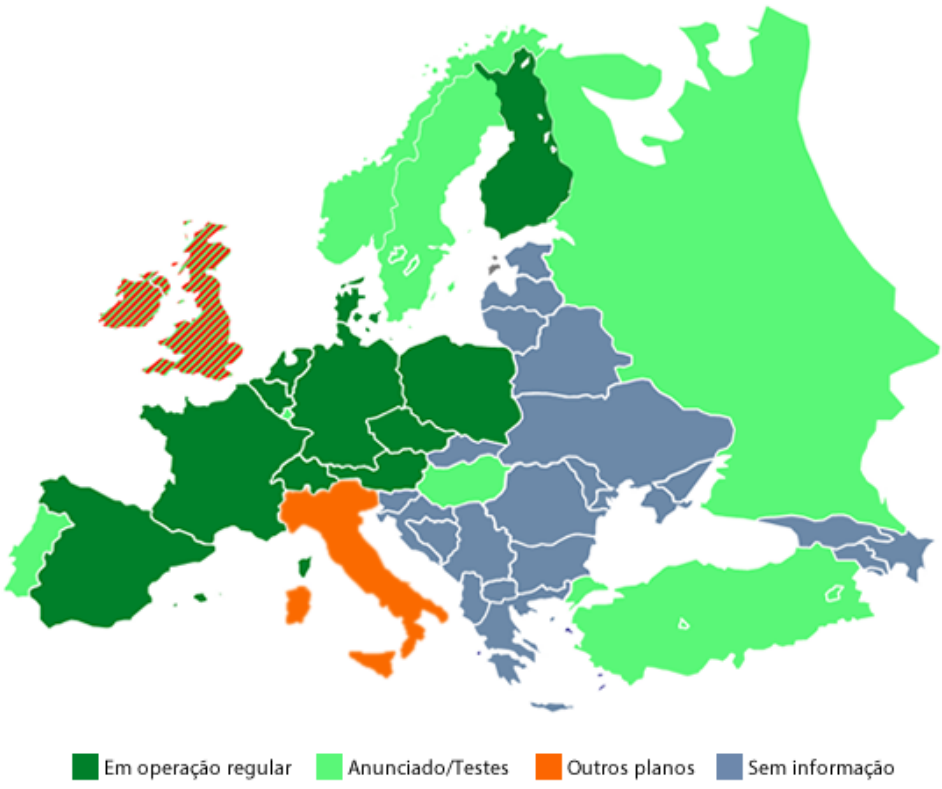

Figura 2.3: Estado da implementação da HbbTV®em 2014 [Ned14]

De forma a criar aplicações compatíveis com diferentes *Smart TVs* dos mais variados fabricantes, foi criada pela BBC uma biblioteca *open-source* (código livre), denominada *TV Application Layer* (TAL) [BBC]. Esta biblioteca permite o desenvolvimento de aplicações HTML5, mesmo que as plataformas disponham de APIs distintas.

A tabela 2.1 faz um resumo das características das tecnologias discutidas neste capítulo.

[6]Linguagem de programação desenvolvida pela Oracle.

Tabela 2.1: Características das tecnologias para *Smart TVs*

	Plataforma fabricantes	**MHEG-5**	**MHP**	**HbbTV®**	**TAL**
Transversal a vários equipamentos	✗	✓	✓	✓	✓
Linguagem de programação	HTML/ JavaScript/ Flash/ Nativas	Proprietária	Java	HTML/ JavaScript	HTML/ JavaScript
Acesso ao conteúdo linear	✗	✓	✓	✓	✗
Não requer instalação	✗	✓	✓	✓	✗
Executa automaticamente	✗	✓	✓	✓	✗

2.4.1 Tipos de aplicações

Como foi referido aquando da HbbTV®, existem diferentes tipos de aplicações para *Smart TVs*. Essencialmente podemos falar de aplicações associadas a uma emissão de TV/programa de TV, ou de aplicações instaladas no dispositivo. As primeiras são transmitidas através do sinal síncrono (com recurso à ligação de dados opcional) e podem ser automaticamente iniciadas durante uma transmissão [ETS10]. As segundas apenas fazem uso da ligação de dados.

2.4.2 Especificação do HbbTV®

A figura 2.4 ilustra os componentes obrigatórios para uma implementação do HbbTV®.

Através da *Broadcast Interface*, é possível enviar a programação linear a par dos dados da aplicação [ETS12]. Esta informação vem complementada com os dados da tabela AIT e eventos de *stream*. A tabela AIT contém informação acerca das aplicações que vêm distribuídas no sinal [ETS10]. Nesta tabela podem vir referenciadas mais que uma aplicação, associadas a um valor de prioridade e a um código de controlo referente ao *autostart* (este valor sinaliza se uma aplicação deve iniciar automaticamente ou não). É recomendado que apenas uma aplicação tenha o valor *autostart*. Tanto os eventos como os dados da aplicação são enviados num Objeto Carrossel.

Chegada ao terminal, esta informação é desmultiplexada[7] e distribuída por diversos componentes, nomeadamente pelo *Runtime Environment*, o responsável pela execução da aplicação. Este pode ser dividido em dois subcomponentes principais: o *Application Manager* e o *Browser*. O *Application Manager* tem acesso à tabela AIT, que lhe permite decidir qual aplicação deve ser iniciada automaticamente (com *autostart* e mais prioritária), e efetua o controlo do ciclo de vida dessa execução. A aplicação é corrida no *Browser*. Um aspeto importante é o facto do *Runtime Environment* ter acesso a funções e informação adicional provenientes do *Broadcast Processing*,

[7] Processo de separação das diferentes mensagens que foram transmitidas em simultâneo, através do mesmo canal de comunicação.

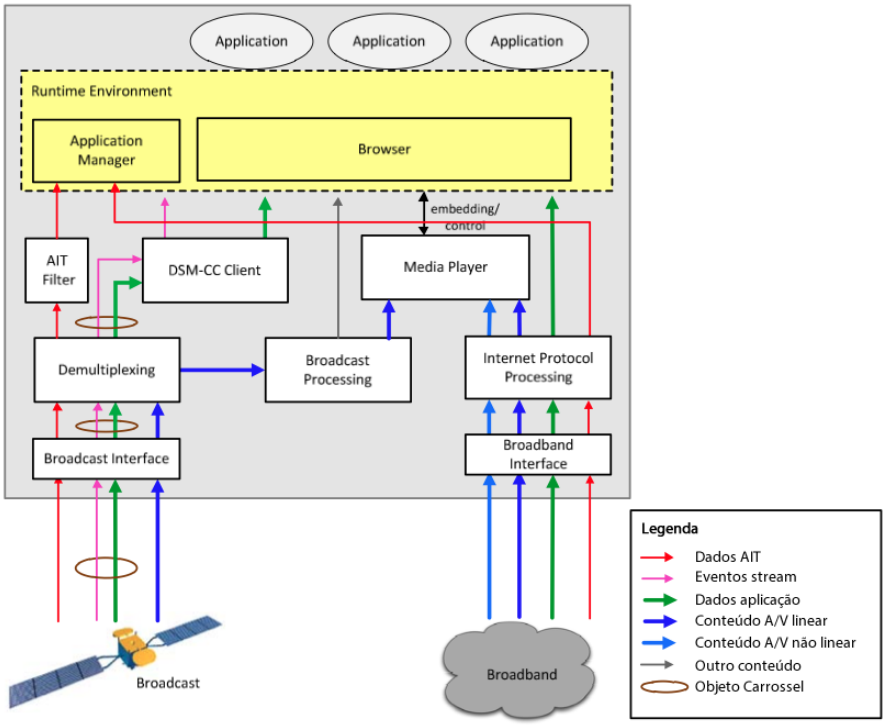

Figura 2.4: Componentes funcionais de um terminal híbrido [ETS12]

componente que trata o sinal difundido. Entre outras, incluem-se aqui funções que fornecem informação temporal, ou a possibilidade de adicionar *handlers*[8] a determinados eventos, de modo a garantir a sincronização com a programação linear.

2.4.2.1 Suites de teste

Embora não exista documentação pública de livre acesso sobre como criar aplicações compatíveis com HbbTV®, é possível consultar o código das aplicações desenvolvidas pelos canais que as implementam e existe uma empresa que disponibiliza uma suite com testes às funcionalidades do HbbTV®. Esta suite (HBBTV tests) foi criada pela MIT-xperts, está disponível com código aberto [Mx14] e pode ser testada com uma transmissão televisiva de demonstração também por eles disponibilizada.

Entre os testes disponíveis destacam-se:

- Obtenção e controlo da lista de canais;
- Substituição e escalamento da imagem do conteúdo linear;
- Transmissão de áudio e vídeo de um servidor;
- Gestão da aplicação (reiniciar, abrir outra aplicação, libertar memória, ...);

[8] Funções que são executadas quando determinado evento, ao qual estão associadas, é despoletado.

- Controlo das teclas do telecomando;
- Obtenção dos dados da tabela de eventos;
- Envio de eventos de *stream*.

2.5 Sincronização temporal

Embora possam ser enviadas através do mesmo canal de transmissão, as aplicações e a programação linear são entidades separadas. Havendo necessidade de, por exemplo, a aplicação reagir em função da transmissão linear, é necessário que estas se encontrem sincronizadas. Uma possível forma de sincronismo seria a obtenção da informação temporal dos programas que estão a ser transmitidos pela estação emissora. Dessa forma bastaria subtrair o tempo atual com o tempo de início do programa, tendo assim o tempo relativo ao início do mesmo. Esta informação é enviada no sinal difundido dentro de uma tabela EIT (Event Information Table) [ETS98], e é usada pelos equipamentos recetores para gerar o EPG. No entanto, não há qualquer garantia que esta informação esteja atualizada, uma vez que é gerada com antecedência pela estação emissora, e pode não ser atualizada caso surja a necessidade de redefinir os horários de emissão.

Outra possibilidade para sincronização é-nos dada pela especificação do DVB. Recorrendo ao DSM-CC, é possível criar um *stream* de eventos que transporta informação, despoletando uma ação no cliente (caso este os tenha subscrito).

2.5.1 Tipos de eventos no DVB

São três os tipos de eventos disponibilizados pelo DVB [ETS10]:

1. Eventos de *stream* programados do DSM-CC, definidos com referência à informação temporal da emissão, nomeadamente NPT;
2. Eventos de *stream* "*do it now*" do DSM-CC, independentes da informação temporal da emissão;
3. Eventos sincronizados do DVB, também eles independentes da informação temporal da emissão, mas mais precisos que os "*do it now*".

Os eventos programados do DSM-CC são eventos que são associados a uma determinada marca temporal da emissão. O equipamento ao recebê-los deverá manter a informação e aguardar que a marca temporal seja atingida para despoletá-lo. Para aumentar a probabilidade de serem recebidos a tempo, estes deverão ser enviados pelo menos uma vez por segundo. No entanto, a possibilidade de haver uma quebra nas marcas temporais enviadas pode resultar na desregulação dos eventos.

Por outro lado, os eventos sincronizados do DVB não utilizam as marcas temporais da emissão, recorrendo a uma informação temporal que é enviada numa estrutura de dados auxiliar. Esta

estrutura de dados é enviada num *stream* elementar de dados que, pelo facto de ser sincronizado com os outros *streams*, garante que os eventos também estejam sincronizados.

Por fim, os eventos do tipo "*do it now*" são eventos que mal sejam recebidos no equipamento recetor devem despoletar o *handler* associado. No entanto, um evento deste tipo é inserido no *payload* de uma secção do MPEG, logo sem qualquer sincronização com os outros *streams*. Ou seja, não é possível prever com grande precisão o momento temporal da emissão em que este será executado no terminal. Cópias da instância de um determinado evento são ignoradas, exceto se estas possuírem um número de versão diferente. Assim é possível enviar um evento múltiplas vezes com uma nova versão e, por exemplo, dados privados diferentes, que o recetor irá recebê-los todos.

A seguinte tabela ilustra as variáveis presentes num evento de *stream* [For14]:

Tabela 2.2: Estrutura interna de um evento

Campo	Descrição
name	Nome do evento.
data	Dados do evento codificados em hexadecimal.
text	Dados privados codificados como texto.
status	Estado do evento. Pode assumir os valores de "*trigger*", quando recebido em resposta a um evento e com sucesso, ou "*error*", quando em situação de erro, nomeadamente na tentativa de adicionar um *handler* a um evento que não existe.

2.6 Emuladores/Servidores de emissão HbbTV®

As aplicações criadas segundo a norma do HbbTV®são essencialmente HTML, pelo que é possível fazer testes preliminares num navegador (*browser*) de computador. No entanto, como existem especificações próprias e são fornecidas várias APIs de interação com o equipamento, nomeadamente *handlers* para as teclas do telecomando, são necessárias soluções adaptadas. Existem ferramentas/emuladores para que se possa testar localmente num computador, mas para se validar com um equipamento real é necessário enviar a aplicação numa transmissão DVB, sendo necessário um servidor de emissão e equipamento físico que gere o sinal.

2.6.1 Opera HbbTV Emulator

A Opera disponibiliza uma imagem para uma máquina virtual[9] que permite emular um equipamento real. Tem suporte para os objetos com as APIs do HbbTV®, permite simular o envio da aplicação através de DSM-CC, ou via Internet, impor restrições ao nível da memória do equipamento, e possui um telecomando virtual para simular a interação do espetador. Não possui forma de testar eventos de emissão.

[9]http://dev.opera.com/tv

2.6.2 Plugin FireHbbTV

O FireHbbTV é um plugin[10] para o navegador Firefox que adiciona o suporte para os objetos com as APIs do HbbTV®, permite usar o teclado como telecomando e simula as dimensões e características de um ecrã de televisão. Embora anuncie a possibilidade de simular eventos de transmissão, nos testes efetuados tal funcionalidade gera uma exceção[11], não permitindo a sua utilização.

2.6.3 Opencaster

O Opencaster[12] é um gerador e manipulador de pacotes de *Transport Streams* MPEG2 desenvolvido pela Avalpa. É grátis, de código aberto, permite gerar as tabelas necessárias para a transmissão DVB, é compatível com HbbTV®, permite gerar e multiplexar o *Transport Stream* do DSM-CC, incluindo eventos, e, com recurso a um modulador[13] de rádio frequência compatível, permite enviar o sinal de DVB-T para um televisor com sintonizador digital. Não implementa todas as opções do DVB e só suporta eventos do tipo "*do it now*", mas permite ainda a transmissão por *multicast*[14], através de UDP[15]. No entanto é necessário software específico para reproduzir aplicações HbbTV®, tal como o DVBViewer (http://www.dvbviewer.com/).

2.6.4 Comparação

A tabela 2.3 contém um resumo das características que cada uma das ferramentas/emuladores/servidor suportam.

Tabela 2.3: Comparação das características das ferramentas/emuladores

Ferramenta	HTML/JS/CSS	HbbTV®	Eventos	DSM-CC
Browser	✓	✗	✗	✗
Opera	✓	✓	✗	✓
FireHbbTV	✓	✓	?	✗
Avalpa	✓	✓	✓	✓

O servidor Avalpa é a ferramenta mais completa e a única que permite avaliar a aplicação num equipamento real, apesar de necessitar de equipamento adicional.

[10] Módulo de software que adiciona funcionalidades a um determinado programa.
[11] Evento lançado durante a execução de um programa que sinaliza a ocorrência de uma disrupção do seu funcionamento normal.
[12] http://www.avalpa.com/
[13] Conversor de sinal digital para sinal analógico.
[14] Técnica de transmissão de informação um-para-muitos sob a rede IP, em que só os clientes que subscreveram a informação a recebem.
[15] Protocolo de comunicação sob a rede IP.

2.7 Entrega de conteúdos

De todas as alterações que a indústria televisiva tem sofrido, nem todas são visíveis para os espetadores. De facto, muitas das evoluções tecnológicas, além de permitirem a melhoria da qualidade dos programas de televisão, permitem a melhoria dos processos de produção e transmissão. A utilização de ferramentas mais sofisticadas torna possível a automatização e otimização de alguns processos de produção, outrora manuais.

Uma das maiores e mais recentes transformações foi a mudança do paradigma de entrega, troca e armazenamento de conteúdos, de cassetes para sistemas baseados em ficheiros. Possível devido à proliferação do computador, a troca digital de conteúdos, através da Internet ou unidades de armazenamento digital, veio reduzir custos e simplificar os processos. Apesar disto, a proliferação de formatos de codificação incompatíveis veio dificultar a integração nos processos dos intervenientes da cadeia de produção e transmissão.

2.7.1 MXF

O número de formatos disponíveis para codificação de vídeo, áudio, metadados ou outros, é considerável. No entanto, todos podem ser divididos em dois tipos: exclusivamente essência ou contentores/embrulhos (*wrappers*). Um formato do tipo contentor pode transportar diversos tipos de essências A/V ou outros dados, sincronizados e confinados num único ficheiro. Este tipo de formato pode ser muito útil para o transporte e organização de informação que de alguma forma se relacione [Kov12].

Como foi referido aquando da anotação de conteúdos, o MXF é um formato aberto, criado pela SMPTE, em colaboração com a indústria [Dev02]. A sua criação nasce da necessidade de definir uma norma para a partilha de conteúdo audiovisual, dentro de uma cadeia de produção, aumentando a interoperabilidade entre os intervenientes. Funciona como um contentor que pode albergar diferentes tipos de conteúdos multimédia, associados a outros dados, como metadados.

Um ficheiro MXF contém um cabeçalho e um rodapé, com grupos de itens pelo meio, como essências, metadados descritivos e metadados técnicos [Fer10, Wil00]. Um item é caracterizado por um tripleto "chave", "comprimento", "valor" (K, L, V). O valor da "chave" identifica o tipo de conteúdo presente em "valor", o "comprimento" identifica o tamanho de "valor", e "valor" contém o conteúdo que compõe o item. A utilização desta estrutura permite que o interpretador do ficheiro possa ignorar os itens que desconhece, avançando o seu comprimento até ao item seguinte.

2.8 Separação do conteúdo da apresentação

Um princípio importante no desenvolvimento de software é o princípio da responsabilidade única [Mar03, cap. *SRP: The Single Responsibility Principle*]. Este princípio estabelece que os componentes não devem ter mais que uma responsabilidade, neste caso concreto, que não deve haver mistura entre a forma de organizar os dados e a forma de os apresentar. Primeiro, porque são conceitos distintos. Depois, porque deixam de estar dependentes entre si e podem ser alterados

independentemente. Isto permite que alguém defina o aspeto, enquanto outro define os dados que vão ser mostrados. Se um componente possui mais que uma responsabilidade, estas passam a ser dependentes e uma pode prejudicar ou inibir a possibilidade de evolução da outra. Este tipo de situações gera designs frágeis que podem ter como consequência erros inesperados.

2.8.1 Handlebars

O Handlebars [Han13] é um motor de *templates*[16] sob a forma de uma biblioteca JavaScript, que permite separar os dados da sua apresentação. Corre do lado do cliente e, ao receber dados e um *template*, gera o código HTML resultante da junção dos dois, com a informação inserida em locais relevantes. Para que isso seja possível os *templates* necessitam de conter expressões que indicam a informação que deve constar nessa localização.

2.9 Conclusões

Partiu-se para o levantamento do estado da arte desta dissertação com o objetivo de identificar tecnologias existentes que pudessem ser utilizadas para a concretização do projeto. Pretendia-se garantir interoperabilidade com equipamentos existentes, de modo a garantir uma solução flexível e fácil de implementar, aproveitando os equipamentos que os utilizadores já dispõem em sua casa.

Existem várias formas de anotar conteúdos, contudo verifica-se que várias se baseiam no MPEG-7, especializando-se em determinada área funcional. O MPEG-7 é assim mais genérico, permitindo uma maior adaptação, personalização e espectro de aplicação. A sua adaptação ao propósito desta dissertação deverá ser acompanhada de uma aproximação ao esquema das normas que especificam anotações para trocas de material audiovisual. Dessa forma permitirá que haja uma fácil conversão entre aquelas e a estrutura criada.

Em relação à disponibilização de aplicações para *Smart TVs*, podem-se distinguir as aplicações instaladas nos terminais das que são transmitidas pela norma do HbbTV®. Uma vez que um equipamento compatível com HbbTV®necessita de cumprir um número de requisitos mínimo, estas últimas apresentam diversas vantagens, como não necessitarem ser adaptadas às diferentes APIs de cada fabricante, não necessitarem ser instaladas e poderem ser lançadas automaticamente sem intervenção do utilizador. O facto de ser uma tecnologia já implementada e em expansão, também garante que uma solução que recorra dela atingirá um mercado mais amplo, conferindo uma maior atratividade para quem a distribui. Recorrendo aos eventos de *stream* do DVB, é possível garantir o sincronismo entre estas aplicações e a programação linear, respondendo a um dos objetivos desta dissertação. A aplicação a desenvolver deverá ainda tirar partido da separação entre o conteúdo e a sua forma de apresentar, garantindo uma maior distribuição das responsabilidades pela cadeia de produção e distribuição audiovisual. A utilização do MXF também poderá trazer vantagens na integração da solução nesta cadeia.

[16] Modelos que especificam o grafismo, por exemplo de uma página de internet.

De modo a garantir o cumprimento dos objetivos, a solução encontrada deverá ser testada num ambiente o mais próximo da realidade, nomeadamente com recurso ao servidor Avalpa e a um equipamento com a norma HbbTV®.

Capítulo 3

Especificação e Desenvolvimento

Como referido em 1.2, pretende-se definir um modelo de dados e desenvolver uma solução integrada para transmissão e apresentação/conversão desses em conteúdos interativos, de modo que não seja preciso equipamento complementar, tanto para o espetador como para o emissor, diminuindo a complexidade e mantendo o sincronismo com o sinal difundido. Para se atingir este objetivo é necessário:

- Identificar o tipo de interatividade que deve ser oferecido ao utilizador;
- Identificar o tipo de metadados e como representá-los;
- Definir como adicionar estes metadados adicionais ao sinal linear de TV;
- Especificar uma camada de software para tratar os metadados do lado do cliente e a forma de envio e execução nesse mesmo lado;
- Criar de um protótipo que demonstre as possibilidades das soluções propostas, simplificando e permitindo a interação dos consumidores com o conteúdo adicional.

Por fim, é necessário estabelecer a forma de avaliar o resultado final e relacioná-lo com os objetivos propostos.

3.1 Modelo de interatividade

Na solução a implementar pretendia-se um modelo de interatividade semelhante ao que é atualmente usado no mercado, ou seja, um modelo assente na distribuição unidirecional de informação complementar à programação linear.

Pretendia-se que os produtores de conteúdo pudessem inserir informação complementar no sinal que acompanha a transmissão, nomeadamente informação contextual, dando ao utilizador a hipótese de a consultar de uma forma interativa, ou seja, passível de navegar localmente através do

telecomando da televisão. Para além disso, que fosse possível inserir apontadores para fontes externas, podendo os utilizadores acedê-las através da ligação de dados (*broadband*). A informação poderá conter texto, imagem ou vídeo, bem como ligações para sites relevantes.

3.2 Requisitos do sistema

Para guiar o desenvolvimento da aplicação foram estabelecidos requisitos que se pretendiam ver suportados na solução. Tratando-se de uma solução que percorre e engloba toda a cadeia de produção de conteúdos audiovisuais, estabeleceram-se requisitos diferenciados para os intervenientes em cada fase dessa cadeia. Uma vez que nesta dissertação foi focada a área da publicidade, são adicionalmente considerados intervenientes desta área.

Tabela 3.1: Requisitos das agências de publicidade

ID	Descrição
AP1	Uma agência de publicidade deve poder fornecer metadados formatados segundo um formato aberto com informação contextual referente a um conteúdo audiovisual e, possivelmente, localizada espacial e temporalmente (caso as anotações sejam referentes a um anúncio por si criado).
AP2	Uma agência de publicidade deve poder fornecer um apontador para metadados disponíveis num servidor web.
AP3	Uma agência de publicidade deve poder definir a forma como a apresentação dos metadados será feita no equipamento recetor do espetador.

Tabela 3.2: Requisitos dos produtores de conteúdos

ID	Descrição
P1	Um produtor deve poder fornecer metadados formatados segundo segundo um formato aberto com informação contextual, localizada espacial e temporalmente, referente a um conteúdo audiovisual.
P2	Um produtor deve poder fornecer um apontador para metadados disponíveis num servidor web.
P3	Um produtor deve poder definir a forma como a apresentação dos metadados será feita no equipamento recetor do espetador.

Tabela 3.3: Requisitos das estações emissoras

ID	Descrição
B1	Uma estação emissora deve poder definir a forma como a apresentação dos metadados será feita no equipamento recetor do espetador.
B2	Uma estação emissora deve poder enviar a aplicação multiplexada no seu sinal.
B3	Uma estação emissora deve poder enviar eventos para sincronizar a emissão.

Tabela 3.4: Requisitos dos espetadores

ID	Descrição
E1	Um espetador deve poder ver o conteúdo contextual, sincronizado com a emissão do canal.
E2	Um espetador deve poder ignorar o conteúdo contextual caso seja essa a sua vontade.
E3	Um espetador deve poder navegar na aplicação recebida através do seu telecomando.

3.3 Arquitetura funcional do sistema

O objetivo primordial desta dissertação é desenvolver uma solução de interatividade no contexto do consumo de conteúdo síncrono, evitando a complexidade e fornecendo aos produtores a possibilidade de especificar os conteúdos. A solução é, portanto, transversal a toda a cadeia de produção e emissão. A partir dos requisitos identificados é possível observar que alguns aspetos são comuns a mais que um interveniente e permitem definir um conjunto de características que a solução deve possuir:

- Deve permitir definir os conteúdos e a sua apresentação de uma forma flexível e com responsabilidades distribuídas;
- Deve permitir distribuir os elementos que constituem os conteúdos interativos por diferentes plataformas;
- Deve ser fácil de implementar, recorrendo a tecnologias que já existem e permitindo adaptar os processos já implementados;

Desta forma a solução deverá conter módulos de produção de conteúdos a implementar do lado dos produtores e emissores, e módulos de exibição e tratamento desses conteúdos a implementar no cliente. Os conteúdos também devem ser normalizados, permitindo a partilha correta entre estes diferentes módulos.

Segundo os objetivos específicos e de acordo com o referido no parágrafo anterior, após identificado o tipo de interatividade a disponibilizar, é necessário definir o formato de uma camada de metadados. Esta camada deve permitir que um determinado conteúdo adicional (uma anotação) seja associado a algo que apareça na transmissão linear. Por essa razão é importante que

os metadados contenham uma referência temporal, permitindo a sincronização com o tempo da transmissão. A referência temporal deve identificar o instante a partir do qual a anotação é válida e um outro instante a partir do qual deixa de o ser. Estes instantes devem ser referentes ao início do programa de TV que acompanham, podendo o último ser representado através da duração da validade (duração da validade mais o instante a partir do qual é válido é igual ao instante a partir do qual deixa de ser válido). Outro aspeto de interesse é a localização espacial. Esta informação permite enquadrar a anotação no ecrã do espetador. Para tal são necessárias as coordenadas dos eixos X e Y e a altura e largura.

Outra característica que é importante na definição dos metadados é a sua flexibilidade para definir os dados. Esta flexibilidade obtém-se permitindo que seja definida qualquer variável e qualquer hierarquia. No entanto, para que possa existir interoperabilidade entre os utilizadores dos metadados é necessário limitar a liberdade na definição destas variáveis e hierarquias. Aqui introduz-se o conceito de perfil. Um perfil define o domínio de um problema, através de variáveis e hierarquias pré-definidas que o caracterizam. Algumas das áreas que têm interesse para o meio audiovisual e interativo são a área de publicidade, de informação/notícias, desporto, teatro, cinema, etc. Em todos os casos é importante que exista uma separação dos dados e da forma de os representar. Dessa forma o seu desenvolvimento é independente e não se limitam.

Após a especificação dos metadados é necessário definir o método da sua introdução no sinal transmitido e definir uma camada de *software* que os trate. Pretende-se uma solução integrada e que automatize o processo ao longo da cadeia de produção. Para que esse objetivo possa ser cumprido é necessário distribuir diversos blocos de *software* ao longo dessa cadeia. No produtor dos conteúdos deverá existir um módulo que facilite a criação das anotações e grafismo e um módulo que gere ficheiros que facilitem o transporte/troca dos conteúdos entre os diversos intervenientes. Quem interage com estes módulos deverá ser alguém com responsabilidade para tal. Na estação emissora deverá existir um módulo que analise e trate automaticamente os metadados recebidos. Deve gerar todos os dados necessários para a transmissão, deve transmitir os dados (aplicação do cliente, metadados e outros ficheiros) multiplexados no sinal e deve assegurar o sincronismo da aplicação do cliente. Do lado do cliente deverá existir uma camada de software com um módulo que importe e trate os metadados, um módulo que importe e mostre o grafismo que representa os dados e um módulo que trate da interatividade. Este último módulo será aquele com que o espetador interage, através do telecomando.

O diagrama 3.1 mostra os blocos que constituem o modelo definido, o modo como estes se relacionam e quais os atores que intervêm na sua utilização. Destes, foram implementados os que pertencem ao equipamento do cliente e foi desenvolvido o modelo de dados que é transversal à cadeia. Este diagrama e seguintes foram gerados recorrendo ao *software* Enterprise Architect.

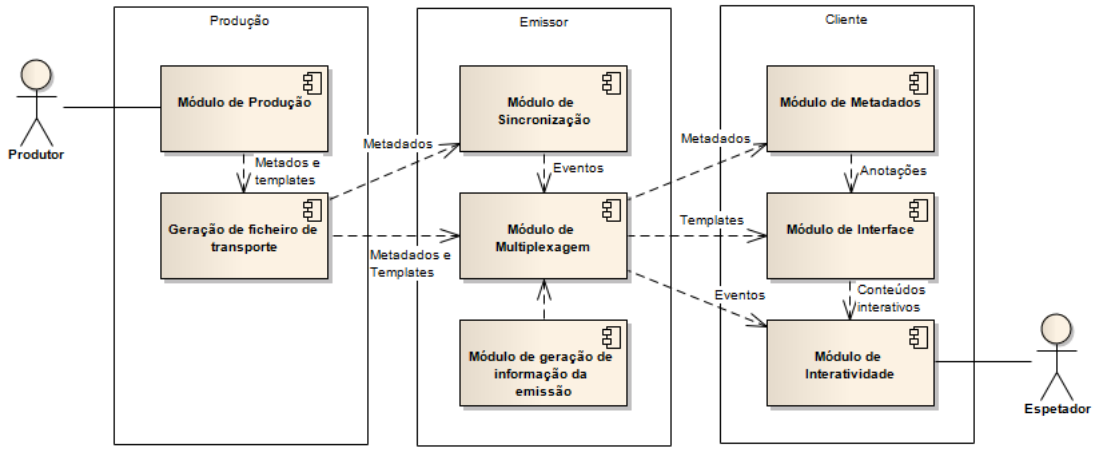

Figura 3.1: Diagrama da arquitetura funcional do sistema

Para implementar a solução, a norma HbbTV®é a tecnologia que mais importância tem. Esta permite o envio de uma aplicação para o cliente, tanto através do DSM-CC multiplexado no sinal difundido, como através da Internet, como permite sincronizar a aplicação através de eventos de *stream*, como descrito em 2.5.1. A aplicação foi desenvolvida em HTML, com recurso ao JavaScript para programar os módulos anteriormente descritos. De forma a facilitar a programação, foi usada a biblioteca jQuery. A divisão do código obedecerá à divisão das áreas funcionais do problema, como ilustra o diagrama 3.1. Para desenvolvimento dos metadados foi usado o MPEG-7, definindo uma extensão com a informação adicional e um perfil de exemplo. Para o módulo da interface foi implementado o sistema de *templates* Handlebars.

3.4 Metadados

3.4.1 Características

Como referido anteriormente, os metadados têm como objetivo fornecer informação complementar a um determinado acontecimento/objeto/entidade, que aparece num instante da emissão. Por essa razão necessitam de ser localizados temporal e espacialmente. Para cumprir com o primeiro objetivo, os metadados deverão conter descrições com informação sobre o tempo a partir do qual são relevantes (início) e durante quanto tempo (duração). Na eventualidade da descrição ser relevante durante todo o período de emissão, os dados temporais deixam de ser necessários e deverão poder ser ocultados.

De modo a oferecer uma solução que possa ser adaptada a diferentes ambientes e requisitos, os dados que anotam o conteúdo deverão poder vir de uma fonte externa, ou ser inseridos diretamente na estrutura dos metadados. Esta solução permite descrições mais personalizadas e direcionadas para o espetador, uma vez que um pedido para obter uma anotação via Internet transporta consigo dados do equipamento recetor. Imaginemos um exemplo: uma agência de publicidade produz material que será inserido numa emissão de um canal de televisão que transmite a nível mundial.

Os metadados que acompanham a emissão contêm uma referência para uma descrição externa, localizada no servidor da agência. Cada vez que o conteúdo publicitário venha a aparecer a um espetador, o servidor da agência recebe um pedido via Internet. Este pedido permite-lhe analisar dados como a sua localização, fornecendo assim um anúncio adaptado ao mercado, além de permitir obter informação estatística acerca da visualização e alcance da campanha.

Uma anotação normal deverá conter informação que descreva espacialmente a descrição e uma hierarquia de etiquetas que definam os dados da mesma. Estas etiquetas deverão ser duplos [variável, descrição], em que a descrição poderá estar definida em diferentes línguas. Deverão ainda existir perfis que delimitem as etiquetas, definindo uma hierarquia de variáveis possíveis, e permitindo que haja interoperabilidade entre os diferentes intervenientes da cadeia de produção. A hierarquização das variáveis também é um aspeto importante, uma vez que permite uma melhor organização da informação, facilitando a sua obtenção.

Uma anotação externa deverá ser uma anotação normal unitária, sem informação temporal (uma vez que esta se encontra nos metadados que a referenciam) e que se encontra localizada num servidor externo, referenciado por um apontador (*url*).

A figura 3.2 ilustra o modelo conceptual tendo em conta as características descritas.

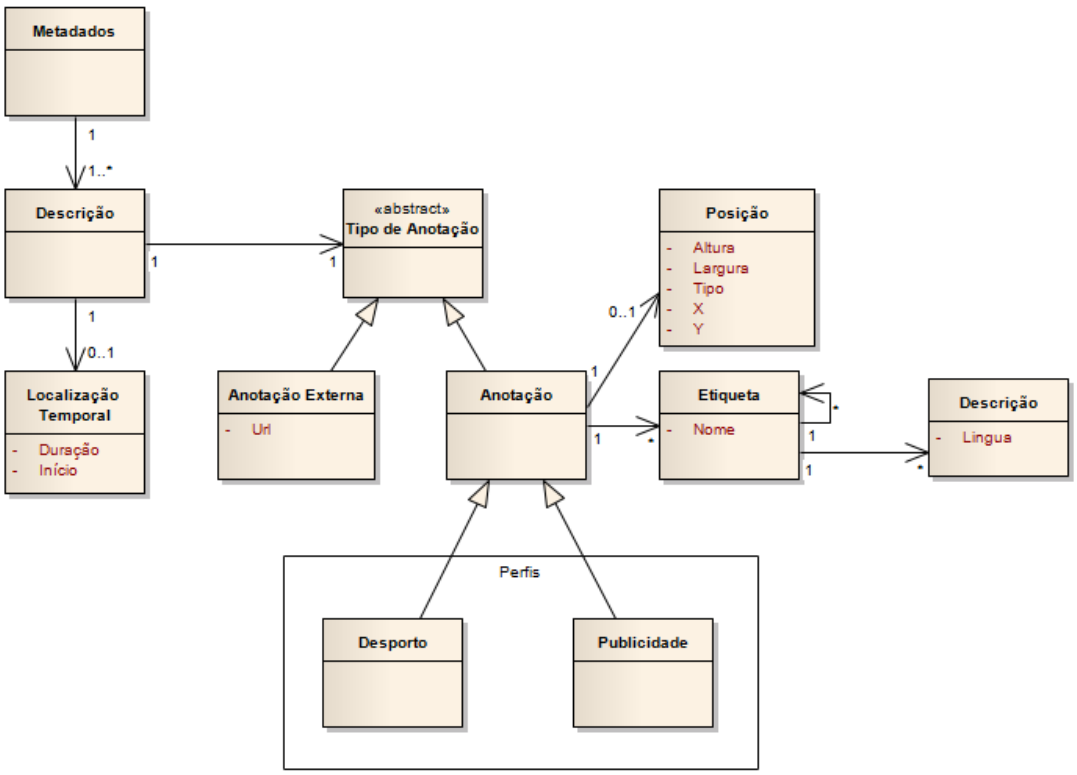

Figura 3.2: Modelo conceptual dos metadados

De modo a acomodar as características propostas e oferecer uma solução baseada em normas abertas e portanto com maior potencial de aceitação, foi criada uma extensão do MPEG-7. Como

foi referido em 2.3.3, o MPEG-7 permite localizar temporalmente uma descrição, pelo que as anotações serão integradas sob esta etiqueta. Com uma extensão é possível adicionar uma anotação voltada para a apresentação de conteúdo acessório como aquela que é pretendida nesta dissertação, juntamente com todas as outras anotações que o MPEG-7 já permite. Assim, é possível transmitir a informação toda concentrada num único ficheiro. Na eventualidade de existir *software* compatível com MPEG-7 responsável por interpretar este ficheiro, poderá fazê-lo com os dados que compreende, ignorando aqueles que foram adicionados nesta extensão.

Uma anotação MPEG-7 começa com a etiqueta raiz *Mpeg7* (figura 3.3). Esta pode conter uma ou mais descrições (etiqueta *Description*), ou uma descrição unitária (etiqueta *DescriptionUnit*), usada nas anotações externas.

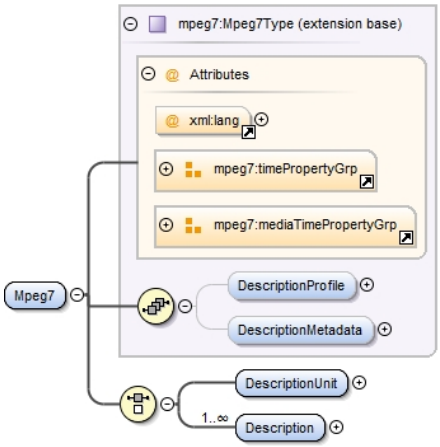

Figura 3.3: Estrutura do MPEG-7

A etiqueta *Description* faz referência a uma classe abstrata[1] pelo que é necessário referir qual a classe da qual se pretende herdar as características. Isto faz-se identificando, na etiqueta do XML, o atributo *type="nome da classe"*. Posteriormente, para fazer uma anotação de um conteúdo multimédia, é necessário indicar a classe *ContentEntityType*, uma vez que esta contém uma ou mais etiquetas do tipo *MultimediaContent*. Esta etiqueta volta novamente a referenciar uma classe abstrata, pelo que é necessário, mais uma vez, indicar a classe filha. Entre as escolhas existem diversas classes referentes a diferentes tipos de conteúdo multimédia mas, como se pretende anotar um vídeo, a escolha recai sobre a classe *VideoType*, que contém a etiqueta *Video* (do tipo*VideoSegmentType*). É esta a etiqueta que recebe a extensão discutida.

Quando se trata de uma anotação externa, dentro da etiqueta *DescriptionUnit* apenas existe a etiqueta *Video*.

A figura 3.4 e seguintes, retiradas da documentação gerada pelo *software* Oxygen XML Developer, ilustra o código XML desenvolvido.

[1] A terminologia usada nos parágrafos seguintes baseia-se nos conceitos da Programação Orientada a Objetos.

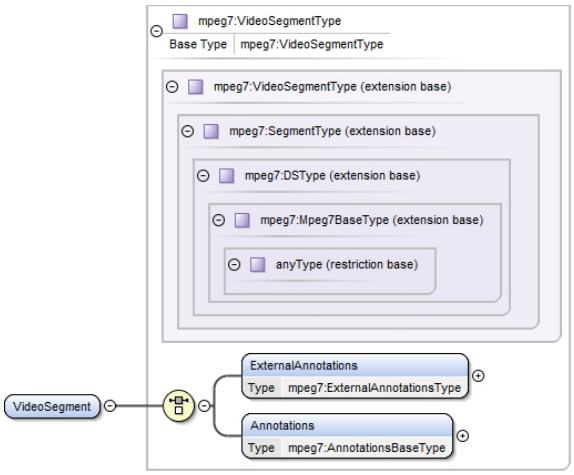

Figura 3.4: Extensão a *VideoSegmentType* (só os elementos novos visíveis)

A classe *VideoSegmentType* contém a etiqueta *MediaTime*, onde é localizada temporalmente a descrição, e com a extensão passa a ter a etiqueta *Annotations* ou *ExternalAnnotations*, conforme é uma descrição normal ou externa. Sendo uma anotação externa, a etiqueta apenas contém o atributo *url* para que se possa indicar a localização do ficheiro externo.

Tratando-se de uma anotação normal (imagem 3.5), a etiqueta contém o atributo *template* que indica qual o modelo que deverá ser usado aquando da apresentação do conteúdo no ecrã. Contém ainda uma etiqueta *Position*, que permite localizar espacialmente através dos atributos *X*, *Y*, *largura* e *altura*, e uma ou mais etiquetas *Annotation*, que representam o duplo [variável, descrição] discutido anteriormente. Todas estas etiquetas se encontram definidas dentro do esquema *AnnotationsBaseType*.

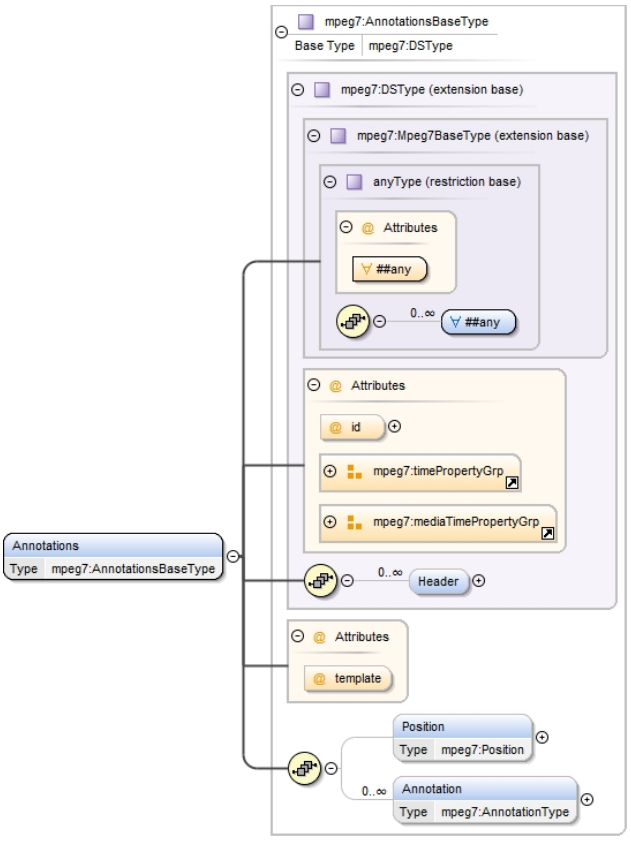

Figura 3.5: Vista de pormenor de *Annotations*

Estes duplos contêm o atributo *typeLabel*, que identifica o nome da variável, e sub-etiquetas *Definition* com a descrição associadas a uma linguagem. Para permitir a hierarquização, podem ainda conter outras etiquetas *Annotation*, como pode ser visto na imagem 3.6.

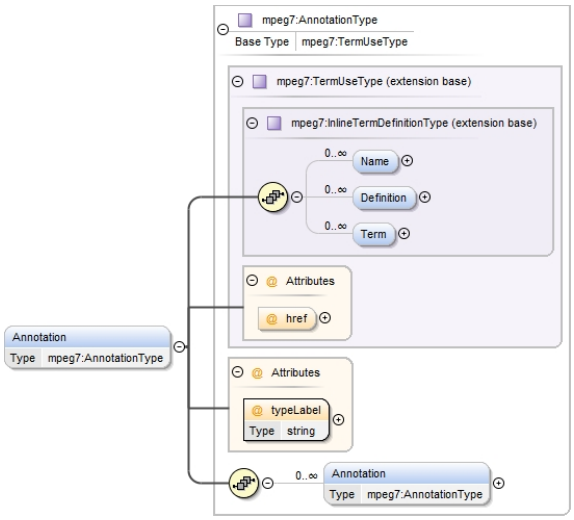

Figura 3.6: Vista de pormenor de *Annotation*

O código desta extensão ser consultado no anexo A.1.

3.4.2 Perfis de dados

Como foi referido em 3.3, a existência de perfis de dados prende-se com a necessidade de interoperabilidade, ou seja, de garantir que são geradas estruturas idênticas e espectáveis, permitindo a normalização das etiquetas que compõem a cadeia de produção.

Estes perfis continuam a permitir a utilização de variáveis livres, definíveis por quem cria os metadados, como descrito em 3.4.1, mas fornecem uma lista de conceitos organizados hierarquicamente, relacionados com a área a que se destinam. Apesar disto, a estrutura deverá assemelhar-se à das variáveis livres. Ou seja, um conceito poderá conter múltiplas definições associadas a uma linguagem, através do uso da etiqueta *Definition*. No entanto, em atributos não sujeitos a linguagem, a sua definição pode ser inserida diretamente, omitindo as sub-etiquetas.

Um exemplo concretizado pode ser visto nas secções seguintes, nas quais são descritos perfis voltados para a publicidade e para o desporto.

3.4.2.1 Publicidade

O perfil de publicidade tem dois objetivos: descrever um produto que apareça durante a emissão de um programa de TV, fornecendo etiquetas para que possa ser apresentada informação/publicidade ao espetador; descrever um anúncio publicitário, fornecendo informação acerca do produto e do anúncio em si.

A EBU possui uma especificação de metadados para descrever a publicidade, o egtaMETA. Esta foi a base usada para o desenvolvimento do perfil, permitindo assim uma fácil conversão entre o material fornecido pelas agências de publicidade e as anotações para a aplicação. No entanto, foram acrescentados alguns campos para que sejam fornecidas mais informações relevantes para um serviço interativo, tal como informação específica acerca da música de fundo de um anúncio (figura 3.7).

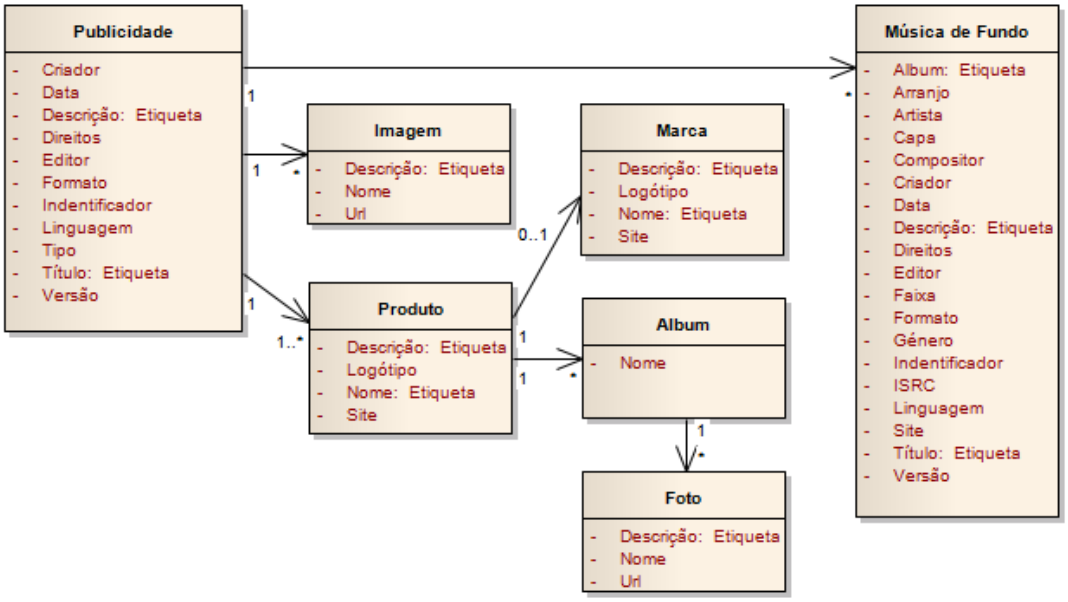

Figura 3.7: Modelo conceptual do perfil Publicidade

O anúncio pode ser descrito pelo título, criador, data, entre outros, e ainda por imagens e música de fundo. A música de fundo contém as variáveis relevantes para que se possa localizar a música, nomeadamente a informação do artista, o nome do álbum, etc., e pode conter informação sobre a capa do álbum e sobre um site, por exemplo uma loja de música.

O produto contém informação sobre o nome, descrição, logótipo e site, bem como informação idêntica sobre a sua marca e fotos que o possam ilustrar.

Todas as variáveis presentes no modelo conceptual (figura 3.7) que são do tipo Etiqueta, podem conter descrições em diferentes línguas.

As seguintes imagens ilustram as etiquetas mais importantes do esquema XML deste perfil. A estrutura das variáveis herda as propriedades das variáveis livres, ou seja, recorre à etiqueta *Definition* para definir uma variável em diferentes línguas.

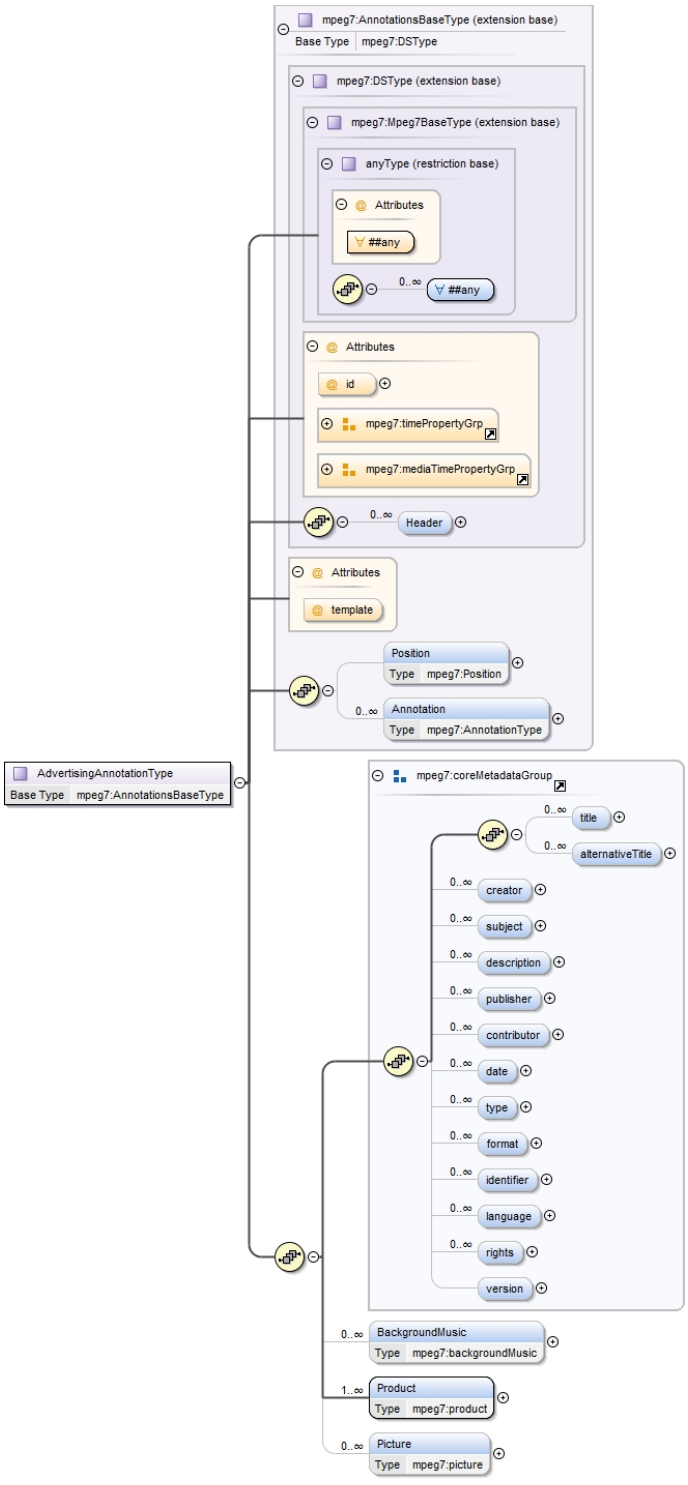

Figura 3.8: Visão geral do perfil Publicidade

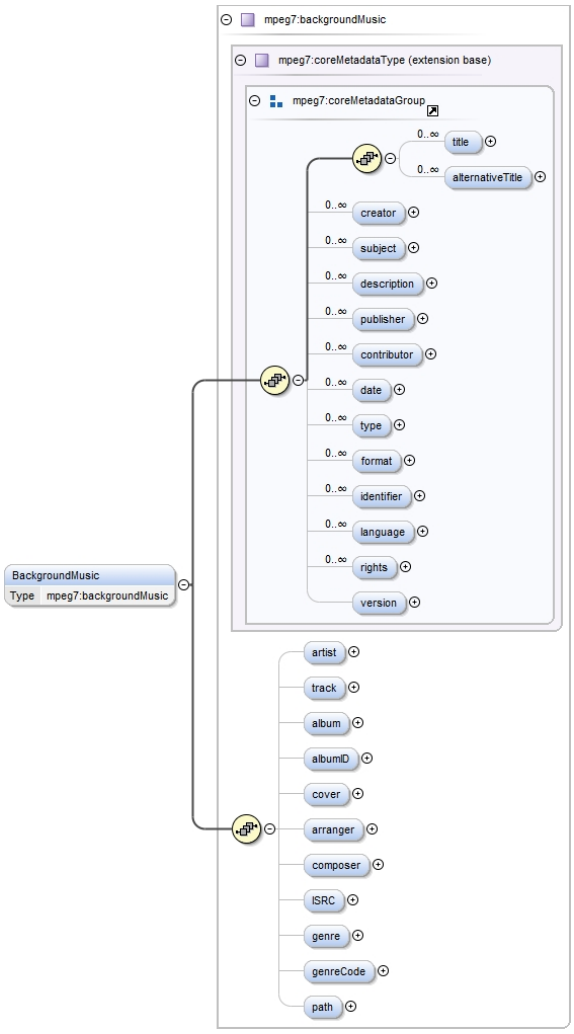

Figura 3.9: Detalhe da estrutura da etiqueta música de fundo

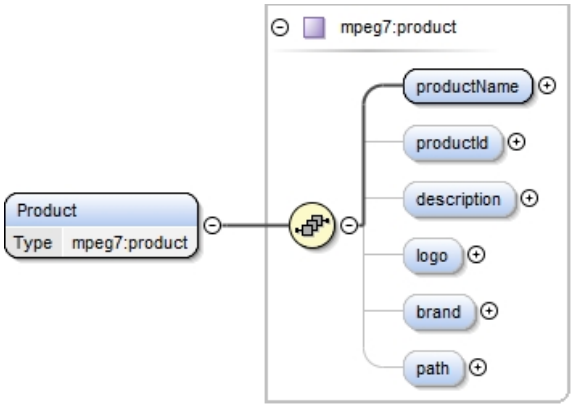

Figura 3.10: Detalhe da estrutura da etiqueta produto

Como pode ser observado na figura 3.8, o perfil herda as propriedades de *AnnotationsBaseType*, que é o esquema que define uma anotação na extensão criada nesta dissertação. É pelo facto de ser uma subclasse desta que é possível definir que a anotação é do tipo *AdvertisingAnnotationType*, o tipo do perfil Publicidade.

O código deste perfil ser consultado no anexo A.2.

3.4.2.2 Desporto

Este perfil, embora não tenha sido implementado, foi alvo de estudo. O objetivo primordial era replicar aquilo que foi feito no perfil Publicidade. Ou seja, pretendia-se encontrar um esquema já desenvolvido, que desse para ser adaptado à estrutura definida na extensão do MPEG-7.

Um perfil desta natureza seria relevante principalmente durante a transmissão de um desporto, tanto em direto como em diferido, ou de um programa de análise desportiva. Nestes casos, independentemente da modalidade desportiva em causa, teria interesse fornecer informação acerca da mesma, dos jogadores, das equipas e dos locais dos eventos, nomeadamente calendários, estatísticas, pontuações, lesões, eventos históricos, patrocinadores, etc. Ao fazer-se uma análise do SportsML, verifica-se que todos estes aspetos foram equacionados, e em bastante pormenor. Esta norma contém diversos módulos que permitem aprofundar as descrições das várias modalidades.

A seguinte lista apresenta alguns dos conceitos que são incluídos na norma e que têm interesse para o desenvolvimento de conteúdo interativo complementar:

- Evento desportivo
  - Metadados do evento
    - Identificações (ids, nomes, etc)
    - Temporada
    - Tipo de desporto
    - Metadados específicos
      - Futebol (Pontapé de saída, . . .)
      - Ténis (Set, Pontuação do serviço, Pontuação da receção, . . .)
      - etc.
    - Patrocinadores
    - Local (estádio, . . .)
      - Capacidade
      - Tipo de campo
      - URL
      - Nome
      - Localização (morada)
      - Estatísticas de ocupação
      - Meteorologia

  - Estatísticas
  - Informação das equipas
    - Nome
    - Local (estádio(s))
    - Cidade de origem (morada)
    - Jogadores
      - Carreira
      - Lesões
      - Dados biográficos
      - Peso/altura
      - Estatísticas (Penaltis, Prémios, Rankings, ...)
  - Ações
    - Faltas, golos, pontos, substituições, penaltis, ...
- Torneios
- Calendário

Para que fosse possível incluir este perfil na extensão, seria necessário definir uma subclasse de *AnnotationsBaseType* onde se incluísse toda a hierarquia de etiquetas desejada. As etiquetas onde pudessem existir descrições em múltiplas línguas deveriam ser do tipo *AnnotationType*, permitindo assim ter várias etiquetas *Definition*. Para usar este perfil bastaria incluir na etiqueta *Annotations* o atributo *type* com o valor igual ao nome do perfil.

3.5 Aplicação

Nesta dissertação foi desenvolvida uma aplicação baseada em JavaScript e HTML que tem como função receber as anotações definidas em 3.4, processá-las, recolher os dados adicionais, como imagens, e apresentar uma interface interativa ao utilizador, sobrepondo e sincronizando-a com a programação linear. Para a sua distribuição é usada a norma HbbTV®, que permite o envio de uma aplicação HTML e lançá-la automaticamente no equipamento do espetador. Embora seja sempre a mesma aplicação, o conteúdo e a interface apresentados dependerão daquilo que for especificado nas anotações e nos *templates* que a acompanham.

São duas as formas possíveis de enviar a aplicação, cada uma com as suas vantagens e inconvenientes:

1. Multiplexada[2] no sinal difundido via Objeto Carrossel;
2. Via Internet.

[2]Combinar várias mensagens separadas para transmissão em simultâneo num único canal de comunicação.

No primeiro caso, a aplicação é enviada com o sinal, o que torna facultativa a ligação à Internet do equipamento recetor. No entanto, a transmissão está sujeita à largura de banda e à taxa de envio do sinal, o que pode tornar-se inviável dependendo da quantidade de informação a enviar. No segundo caso é obrigatória a ligação à Internet, mas apenas está dependente da largura de banda da ligação do espetador. Também necessita que exista um servidor web onde se possa alojar a aplicação, o que obriga à existência de uma estrutura de apoio.

Além da aplicação, também é necessário considerar o envio dos dados adicionais, como as imagens e os metadados. Aliás, uma vez que estes poderão ser os elementos de maior tamanho, esta questão poderá ter grande importância na escolha do método de envio da aplicação. A forma como foi desenvolvida a aplicação obriga que o ficheiro dos metadados acompanhe a aplicação, mesmo que estes possam referir anotações externas, enquanto que os outros elementos podem acompanhá-la ou não.

3.5.1 Arquitetura

A aplicação foi dividida em três módulos, representantes das áreas funcionais desta dissertação:

- Módulo HbbTV®: responsável pela gestão da interação com o equipamento, nomeadamente para gerir os eventos que mostram e ocultam a informação (recebidos através do DSM-CC) e para obter informações técnicas, informações sobre a língua de apresentação e informação sobre as teclas do telecomando premidas (acedidas através da interface de um objeto disponibilizado pelo dispositivo).

- Módulo de *Templates*: responsável por importar os *templates* necessários e combiná-los com os metadados, gerando o código HTML que é usado pelo *Browser* do equipamento. Também é responsável por inserir no código HTML os ficheiros externos (CSS, JavaScript) necessários para a correta apresentação dos *templates*.

- Módulo Metadados: responsável por obter e tratar a informação contida nos ficheiros XML que descrevem o conteúdo complementar (anotações MPEG-7 com a extensão desenvolvida). Gera a lista de eventos que o Módulo HbbTV®usa e envia os metadados numa estrutura simplificada para o Módulo de *Templates*. Os metadados são enviados juntamente com a aplicação e podem conter ligações para ficheiros externos que terão que ser importados.

O diagrama 3.11 ilustra a forma como estes módulos interagem entre si, e de onde podem obter os dados.

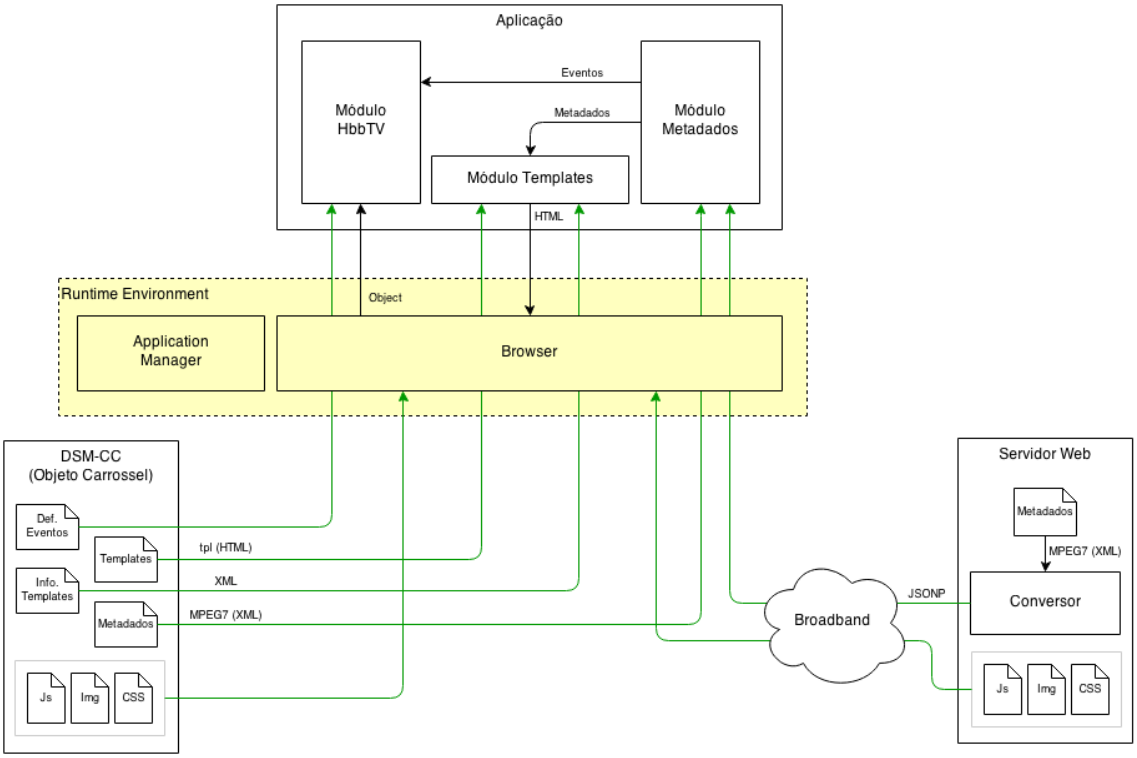

Figura 3.11: Diagrama de interação entre módulos

Este diagrama representa uma possível forma de enviar a aplicação, mas esta não é única, como foi referido no início do capítulo. Aqui podemos verificar que a aplicação é enviada pelo Objeto Carrossel, juntamente com os metadados, os *templates*, o ficheiro de informação dos *templates* disponíveis e os ficheiros complementares (JavaScript, imagens e CSS). A informação dos eventos vai num objeto próprio, também ele multiplexado no Objeto Carrossel. A aplicação corre sobre o *Browser* do equipamento recetor, e possui os módulos discutidos. Opcionalmente, é possível aceder a metadados externos e/ou ficheiros complementares, localizados num servidor web.

Outra forma possível de enviar a aplicação seria através da Internet. Para que tal seja possível, a tabela AIT deverá conter o endereço onde a aplicação está alojada, para que o *Browser* faça o pedido. Neste caso a aplicação também seria acompanhada dos metadados, dos *templates*, do ficheiro de informação dos *templates* disponíveis e dos ficheiros complementares. Também a informação sobre os eventos seria enviada pela Internet, desta vez num ficheiro XML de acordo com a norma. Sendo precisos metadados externos e/ou ficheiros complementares, de um servidor web diferente, tal também seria possível obter.

Outro aspeto observável no diagrama é a interação que existe entre os módulos da aplicação. Podemos verificar que o Módulo Metadados recebe as anotações e encarrega-se de as transformar, enviando para o Módulo HbbTV®a listagem dos eventos, e para o Módulo *Templates* os metadados para serem conjugados com os *templates*. Os eventos, neste contexto, são os momentos temporais em que acontecem as ações mostrar ou ocultar anotação. O Módulo *Templates*, após

combinar os elementos, envia o HTML para o *Browser* para que este possa ser mostrado ao utilizador quando o evento respetivo assim o autorizar. O objeto que é enviado pelo *Browser* para o Módulo HbbTV®é o que permite que a aplicação interaja com o equipamento recetor, uma vez que este disponibiliza uma interface com as funções do HbbTV®.

3.5.2 Extração dos metadados

Embora possa ser contornada, como foi feito no acesso a descrições parciais externas, existe uma política de segurança que impede que sejam acedidos ficheiros XML que não estejam localizados dentro do domínio da aplicação. Por essa razão, os metadados são enviados juntamente com a aplicação, seja o método de transmissão o DSM-CC ou a Internet. A aplicação efetua uma chamada Ajax [3] assíncrona ao ficheiro XML que contém os metadados, permitindo que a aplicação não bloqueie perante uma quantidade de metadados avultada. Esta chamada é efetuada dentro de uma função denominada *open_metadata*. Os dados obtidos são posteriormente convertidos de XML para JSON, uma vez que este é um formato que é fácil de manipular dentro do JavaScript.

Uma vez que uma descrição nos metadados pode apontar para uma descrição parcial externa, localizada num servidor web, a aplicação também é responsável por obtê-los e integrá-los na estrutura interna, como se de uma estrutura única e coesa se tratasse. Todas as chamadas externas são chamadas Ajax assíncronas para que a aplicação não bloqueie. Foi criado um registo onde estas são inseridas, permitindo aferir acerca do seu estado. Deste modo, apenas quando a chamada termina e os dados chegam, se procede ao seu processamento, evitando que a informação seja perdida. Todo este controlo das chamadas permite ainda fornecer informações relevantes para os outros módulos, nomeadamente saber se os dados já foram todos carregados ou se existem chamadas pendentes.

Todas as anotações são guardadas internamente numa variável denominada *metadata*, organizadas por um valor numérico (*id*) que corresponde à sua ordem no ficheiro de metadados. É ainda separada e guardada no Módulo HbbTV®toda a informação referente aos eventos. Ou seja, são guardados os valores numéricos correspondentes ao tempo (provenientes das sub-etiquetas de *MediaTime*), associados a uma ação (mostrar/ocultar) e ao *id* das anotações respetivas.

3.5.2.1 Conversão/Simplificação

Devido a algumas limitações na criação de esquemas XML, e devido à estrutura complexa do MPEG-7, após a extração dos metadados é feita uma conversão para que a estrutura guardada internamente na aplicação tenha uma forma simplificada e mais direta de acesso.

A simplificação da estrutura complexa do MPEG-7 pode ser observada nos seguintes exemplos, onde é ilustrado o código XML original, e o código JSON que representa a informação contida na aplicação. A primeira alteração efetuada consiste na remoção de níveis da hierarquia:

[3] Conjunto de técnicas usadas no desenvolvimento de aplicações web assíncronas.

```
<Mpeg7>
   <Description>
      <MultimediaContent>
         <Video>
            <MediaTime>...</MediaTime>
            <Annotations>...</Annotations>
         </Video>
      </MultimediaContent>
   </Description>
</Mpeg7>
```

Código 3.1: Estrutura original do MPEG-7

```
{
   "description" : {
      "MediaTime" : {...}
      "Annotations " : {...}
   }
}
```

Código 3.2: Representação interna

São ainda simplificadas as etiquetas livres:

```
<Annotation typeLabel="nome_atributo">
   <Definition>valor_do_atributo</Definition>
</Annotation>
```

Código 3.3: Estrutura original do MPEG-7

```
"nome_atributo" : {
   "Definition" : "valor_do_atributo"
}
```

Código 3.4: Representação interna

E as etiquetas de linguagem:

```
<Definition xml:lang="pt">valor_pt</Definition>
<Definition xml:lang="en">valor_en</Definition>
```

Código 3.5: Estrutura original do MPEG-7

```
"Definition" : {
   "pt" : "valor_pt",
   "en" : "valor_en",
}
```

Código 3.6: Representação interna

As etiquetas pertencentes aos diferentes perfis já se encontram na forma simplificada, não sofrendo as simplificações pelas quais passam as etiquetas livres. Estas alterações permitem facilitar o acesso, uma vez que deixa de ser necessário procurar por etiquetas que tenham determinado atributo igual ao nome da variável pretendida, passando a aceder-se diretamente à mesma.

3.5.2.2 Acesso a metadados externos

Como foi referido anteriormente, existe uma política de segurança que impede que seja feita uma chamada Ajax a ficheiros XML localizados fora do domínio da mesma. Para contornar essa situação, é usada a técnica de comunicação JSONP, que aproveita o facto de ser permitido inserir ficheiros JavaScript provenientes de outro domínio. O que acontece é o seguinte: é feita uma chamada para inserir um ficheiro JavaScript externo, fornecendo como parâmetro um nome. O servidor que recebe o pedido para esse ficheiro JavaScript deverá formar uma função com esse nome, que retorna os dados pretendidos. Após a inserção desse ficheiro, a função é invocada, obtendo os dados em formato JSON. Portanto, para aceder às anotações externas, localizadas num servidor externo, é necessário que este contenha uma aplicação que converta o XML do MPEG-7 em JSON e que saiba como responder a uma chamada JSONP. Uma vez recebidos os dados externos na aplicação não é necessário converter o XML em JSON, uma vez que já é este o formato recebido, mas são efetuadas todas as conversões necessárias. Posteriormente, os dados simplificados são inseridos na anotação dos metadados correspondente.

3.5.3 Representação visual

A representação visual dos metadados resulta da inserção destes num *template*, também ele fornecido à aplicação. O Módulo de *Templates* é o responsável por fazer esta tarefa, produzindo código HTML que é inserido no código HTML que acompanha a aplicação.

3.5.3.1 Separação dos dados e apresentação

A separação dos dados da apresentação permite uma maior flexibilidade, uma vez que não obriga o produtor dos metadados a criar os *templates* que os representam. Esta abordagem distribui o trabalho pela cadeia de produção, o que torna a aceitação da solução mais fácil, além de permitir, por exemplo, que uma estação emissora personalize a apresentação dos metadados de acordo com a sua linguagem gráfica. Outra possível vantagem é o facto de os *templates* poderem ser reutilizados com anotações diferentes, diminuindo a quantidade de informação que é enviada para o equipamento recetor, e removendo a necessidade de criar uma representação gráfica por cada anotação.

Os *templates* de uma aplicação podem ter associados ficheiros de estilo CSS e JavaScript externos. Estes ficheiros são especificados num ficheiro XML que caracteriza os *templates*, carregado quando a aplicação inicia. Posteriormente são inseridos no código da aplicação através da função *insert_template_headers*. O ficheiro de caracterização dos *templates* possui uma etiqueta "*template*" com as seguintes etiquetas filhas:

- "*css*", que contém etiquetas "*path*" com o caminho dos ficheiros de estilo;
- "*script*", que contém etiquetas "*path*" com o caminho dos ficheiros JavaScript;
- "*red_button*", que contém o caminho para o *template* do aviso de conteúdo adicional;
- "*renders*", que contém etiquetas "*render*" com o caminho para os ficheiros com o código HTML dos *templates*, e o atributo "*type*" que identifica o nome do mesmo.

A separação das componentes lógica e gráfica implica que exista um *template* associado a cada descrição, havendo a necessidade de as combinar antes de as mostrar. É utilizado um *template engine* (neste caso, o Handlebars) que recebe uma estrutura HTML, formatada com etiquetas especiais, e os metadados, produzindo código HTML compatível com os navegadores web.

A importação dos *templates* é feita através da função *insert_template_model*, que é invocada à medida que determinado *template* é necessário, evitando a sobrecarga da aplicação com *templates* que podem não ser usados. Esta função faz uma chamada Ajax que importa o HTML do *template* e insere-o como um *script*[4] do tipo *text/x-handlebars-template*.

3.5.3.2 Estrutura dos *templates*

Os *templates* contêm código HTML, com etiquetas especiais que são interpretadas pelo Handlebars. Além do acesso à posição (localização espacial) e à anotação dos metadados da sua respetiva descrição, têm acesso ao identificador único da descrição (corresponde ao seu índice), através do atributo *id*, e à linguagem usada no equipamento recetor, através do atributo *language*. Para aceder aos dados da posição, a API do Handlebars permite o seguinte:

Exemplo

[4]Elemento de uma página HTML que contém código JavaScript que deverá ser executado pelo navegador web.

```xml
<Position type="Absolute" origin="BottomRight">
   <X>50</X>
   <Y>65</Y>
</Position>
```

Código 3.7: Código XML

```
{{Position.X}}
{{Position.Y}}
{{Position.type}}
{{Position.origin}}
```

Código 3.8: Acesso às propriedades

Para aceder às anotações, pode ser usada a mesma sintaxe do acesso à posição, ou uma função criada para esta dissertação, denominada *annotation*. Esta função facilita o acesso aos atributos, simplificando a estrutura dos *templates*. Recebe o caminho do atributo, com os níveis separados por ".", e pode, opcionalmente, receber o valor da linguagem pretendida (caso contrário é assumida a linguagem do equipamento do espetador ou, caso não exista, a linguagem por defeito passada como parâmetro da aplicação). No acesso normal, caso haja mais que uma linguagem, esta teria que ser referida obrigatoriamente, mesmo que não se saibam à partida quais existem. Como consequência desta obrigatoriedade, o *template* torna-se mais complexo porque, antes de inserir um atributo, seria necessário verificar a existência de várias linguagens e optar por uma.

Exemplo

```xml
<Annotation typeLabel="AtributoA">
   <Definition xml:lang="pt">Definicao 1</Definition>
   <Definition xml:lang="en">Definition 1</Definition>
   <Annotation typeLabel="AtributoB">
      <Definition xml:lang="pt">Definicao 2</Definition>
      <Definition xml:lang="en">Definition 2</Definition>
   </Annotation>
</Annotation>
```

Código 3.9: Código XML

```
{{annotation AtributoA}}
{{annotation AtributoA.AtributoB}}
{{annotation AtributoA "en"}}
{{annotation AtributoA.AtributoB "pt"}}
```

Código 3.10: Acesso simplificado aos atributos (com a função *annotation*)

```
{{AtributoA.Definition.pt}}
{{AtributoA.AtributoB.Definition.en}}
```

Código 3.11: Acesso normal aos atributos

Os conteúdos interativos/complementares nunca serão mostrados ao espetador sem que este o autorize. Por essa razão, sempre que um novo conteúdo está programado para aparecer, o espetador é notificado com um aviso. Este aviso pode ser personalizado com um *template* especificado no ficheiro XML que caracteriza os *templates*, através da etiqueta *red_button*. A função *change_red_button* importa este *template* e insere-o na aplicação.

3.5.4 Funcionalidades HbbTV®

Para ser possível utilizar o telecomando do equipamento, tirar partido dos eventos, obter dados do equipamento, entre outras funcionalidades, é preciso aceder à API disponibilizada pelo HbbTV®. Para isso, é necessário inserir no código HTML da aplicação três objetos do tipo *application/oipfApplicationManager*, que representa o HbbTV®, *application/oipfConfiguration*, que contém as configurações do terminal, e *video/broadcast*, que representa a emissão linear e permite a subscrição de eventos ou até mesmo a manipulação do tamanho e localização do vídeo transmitido no ecrã. Portanto, é a partir da interação com estes objetos que é efetuada a comunicação com o equipamento físico.

Para que seja possível receber informação das teclas premidas no telecomando, é necessário fornecer ao objeto de configurações uma máscara com as teclas que se pretende que sejam capturadas. Além do botão vermelho, que é a tecla obrigatória reservada pela aplicação desenvolvida, é possível passar como parâmetro os códigos das outras teclas que são necessárias (de acordo com o especificado em [For14]):

Para se poder receber eventos, é necessário registar uma função como *handler*, fornecendo os dados dos eventos, mas isto será discutido de seguida.

3.5.5 Sincronização temporal

No capítulo 2.5 foram referidas as possíveis formas de se sincronizar temporalmente o conteúdo linear com a aplicação. Destas, os eventos do tipo "*do it now*" e eventos sincronizados do

DVB, revelam-se a melhor hipótese, uma vez que não estão sujeitos à imprevisibilidade das grelhas de programação das estações emissoras. Devido às limitações do ambiente de teste (servidor Avalpa), apenas foram considerados eventos do tipo "*do it now*".

Uma vez que estamos a considerar um meio de transmissão, em que cada recetor sintoniza o canal em qualquer instante temporal, independentemente do início da emissão do programa atual, é necessário considerar a questão de eventos passados, mas válidos. O diagrama 3.13 ilustra esta situação.

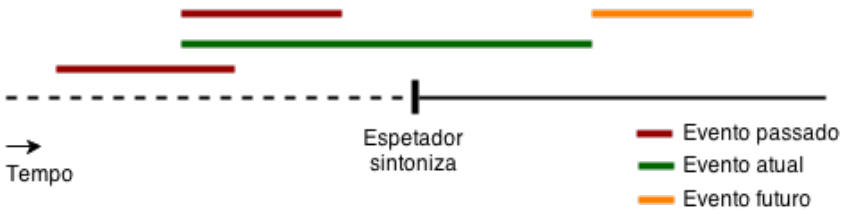

Figura 3.12: Diagrama temporal dos eventos

Caso o espetador sintonize o canal durante a validade (atualidade) de um evento, este deverá ser mostrado. Ou seja, é necessário ter em conta os eventos passados, além dos futuros, quando a aplicação inicia.

Cada marca temporal presente nas anotações dos metadados poderia ser associada a um evento distinto. No entanto, uma vez que os eventos necessitam de ser registados um a um e necessitam de informação que os descreva, as seguintes consequências poderiam acontecer:

- O número de eventos a registar poderia ser tão elevado que atrasaria o arranque da aplicação;
- O número de eventos poderia ser tão elevado que resultaria em imensa informação descritiva que seria necessário transmitir, resultando num atraso da transmissão da aplicação;

Em termos práticos, a combinação das duas situações poderia provocar atrasos significativos no arranque da aplicação. Porém, de acordo com a especificação, é possível usar apenas um evento que, ao ser modificado, gera uma nova notificação no equipamento recetor. Ou seja, é possível enviar um único evento, com a informação sobre a marca temporal. Sempre que o evento é lançado com uma nova marca temporal, atualiza-se o número da versão, fazendo com que o equipamento o trate como um evento novo que efetivamente é. Esta foi a forma como a sincronização foi implementada, tendo sido criado um evento denominado "*action*".

3.5.5.1 Implementação no cliente

Os eventos precisam de ser registados um a um, como se referiu. Definido um único evento, apenas será necessário efetuar um registo. Cada evento tem que ter associada uma descrição, um nome e um *handler*, e é subscrito através da função *addStreamEventListener* do componente que representa a emissão linear do HbbTV®. Caso a aplicação seja transmitida pela Internet, a

descrição do evento consiste num ficheiro XML com o *id* e o nome do evento, associados a um determinado *stream* de eventos que acompanha o programa transmitido (definido na tabela PMT). Caso a aplicação seja transmitida pelo DSM-CC, a descrição do evento é definida com um objeto específico que é enviado multiplexado no sinal, e contém a mesma informação que o ficheiro XML.

No arranque da aplicação é chamada a função *subscribe_events*. Esta invoca *addStreamEventListener*, passando a função *event_listener* como *handler* e o ficheiro com a descrição do evento. No seguinte bloco pode ser visto o ficheiro XML que representa um evento na transmissão via Internet:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <dsmcc xmlns="urn:dvb:mis:dsmcc:2009">
3   <dsmcc_object component_tag="13">
4     <stream_event stream_event_id="1" stream_event_name="action" />
5   </dsmcc_object>
6 </dsmcc>
```

Código 3.12: Descrição do evento *action*

Handlers A função *event_listener* é registada como *handler* do evento. Esta função é chamada sempre que um novo evento é lançado, ou caso exista algum erro a registá-lo (por exemplo, quando a descrição do evento refere um *stream* de eventos que não é sinalizado na tabela PMT). Portanto, é necessário confirmar que o evento é valido, ou seja, contém o estado "*trigger*" e o nome é "*action*".

Os eventos gerados a partir dos metadados são guardados numa lista, ordenados pela sua marca temporal. Depois de confirmada a validade do evento recebido, a lista é percorrida até ser encontrada a marca temporal que corresponde à que foi recebida (primeiro é salvaguardada a sua existência). Se a emissão não tiver sido sintonizada a meio da transmissão, ou não tiver acontecido nenhum erro no envio dos eventos, a marca temporal recebida deverá ser a primeira da lista. Caso contrário, existem eventos anteriores que podem ainda ser válidos. Por essa razão, todos os eventos anteriores são analisados. No fim, para evitar que sejam constantemente analisados, os eventos já tratados são removidos. Todos os eventos válidos são executados para que sejam mostradas (eventos do tipo "*show*") ou ocultadas (eventos do tipo "*hide*") as anotações referenciadas.

Estrutura interna Ao serem importados os metadados, a informação temporal que acompanha as descrições é analisada. Caso não exista esta informação, o *id* da anotação é incluído numa lista de eventos ativos, para que a aplicação, ao iniciar, os mostre imediatamente e os mantenha até à sua destruição. Caso contrário, é mantida uma lista com eventos de dois tipos: "*show*" e "*hide*". Estes eventos estão associados ao *id* da anotação, e são referenciados por uma marca temporal que é a conversão para milissegundos da marca temporal da descrição. Assim, a marca temporal

de *MediaRelTimePoint* (informação temporal relativa ao início do programa de TV que indica a partir de quando é válida a anotação) cria um evento do tipo "*show*" e a marca temporal resultante da soma de *MediaRelTimePoint* com *MediaDuration* (informação temporal relativa à duração da validade da anotação) cria um evento do tipo "*hide*".

Exemplo

```
<MediaTime>
   <MediaRelTimePoint>PT20S</MediaRelTimePoint>
   <MediaDuration>PT2M</MediaDuration>
</MediaTime>
(..)
<MediaTime>
   <MediaRelTimePoint>PT2M20S</MediaRelTimePoint>
   <MediaDuration>PT5M</MediaDuration>
</MediaTime>
```

Código 3.13: Descrições

```
{
   "T20000" :[{
         "id" :1,
         "type" :"show"
      }],
   "T140000" :[{
         "id" :1,
         "type" :"hide"
      },
      {
         "id" :2,
         "type" :"show"
      }],
   "T440000" :[{
         "id" :2,
         "type" :"hide"
      }],
}
```

Código 3.14: Lista de eventos

Por recomendações da norma do HbbTV® e por motivos éticos, este tipo de aplicação não deve ser forçada ao espetador. Por essa razão, um evento que despoleta a ação "mostrar metadados" não faz com que estes sejam mostrados ao espetador. Antes é lançado um aviso a informar que conteúdos complementares estão disponíveis, acessíveis premindo o botão vermelho do telecomando (este aviso pode ser personalizado, como foi referido em 3.5.3.2). Se o espetador premir o botão, a aplicação fica ativa, mostrando a informação complementar. É também modificada a máscara das teclas do telecomando subscritas que passa a ter os códigos de teclas passados como parâmetro no arranque da aplicação. O aviso de informação complementar só aparece caso a aplicação esteja inativa e seja recebido um evento para mostrar novos conteúdos. Se a aplicação estiver ativa são mostrados automaticamente. Se os conteúdos continuarem válidos, mas o utilizador decidir desativar a aplicação, o aviso não é mostrado porque assume-se que o utilizador já foi informado da sua existência.

O seguinte diagrama esquematiza o processo de envio de eventos, incluindo os eventos gerados pelo telecomando.

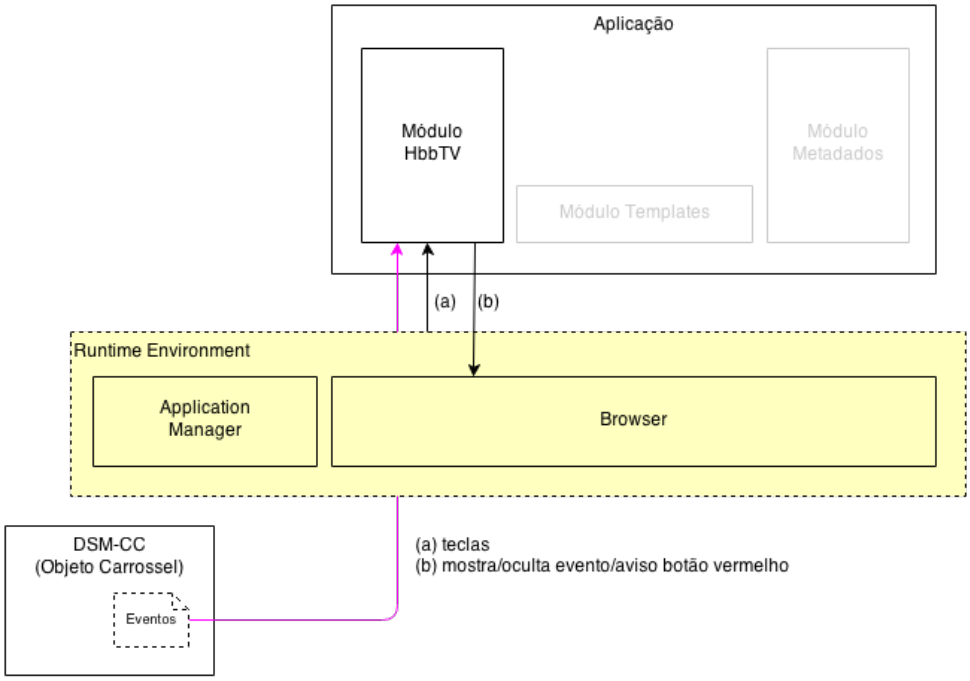

Figura 3.13: Esquema das ações geradas pelos eventos

3.5.5.2 Implementação no emissor

Do lado do emissor é preciso sinalizar a existência de um *stream* de eventos através de uma entrada na tabela PMT e é preciso enviá-los multiplexados no sinal.

Para sinalizar a existência do *stream* de eventos, é necessário adicionar a seguinte informação na tabela PMT:

```
stream_loop_item(
  stream_type = 12,
  elementary_PID = #pid_do_stream_de_eventos,
  element_info_descriptor_loop = [
    stream_identifier_descriptor(
      component_tag = #component_tag_do_stream_de_eventos,
    ),
  ]
)
```

Código 3.15: Extrato da tabela PMT

O tipo de *stream* é 12 de acordo com a norma, o elementary_PID terá que ser o *id* fornecido na codificação do *stream* elementar dos eventos, e o *component_tag* deverá ser o *id* que é usado no ficheiro que descreve os eventos.

Após sinalizado o *stream* podem ser enviados eventos multiplexados no sinal. É gerado um *Transport Stream* com o *id* fornecido na tabela PMT, o *id* do evento e, neste caso, é introduzida a marca temporal no campo privado.

Por cada evento gerado, o número da versão é incrementado. Na eventualidade de se esgotar os números de versão, tal não se traduz num problema, dado que o equipamento recetor apenas compara a versão recebida com a versão recebida anteriormente (e sinalizada na aplicação) para decidir se são diferentes ou não. Se receber um evento com a mesma versão de um outro que tenha sido sinalizado imediatamente antes, descarta-o. Assim, é possível enviar constantemente o mesmo evento, até que seja necessário substitui-lo pelo seguinte. Se um espetador sintonizar o canal após o lançamento inicial do evento, como ele continua a ser enviado, consegue recebê-lo e fazer o sincronismo detalhado em 3.5.5.1.

3.6 Testes e Validação

Tratando-se esta solução de uma prova de conceito/protótipo, a avaliação do cumprimento dos objetivos não passa pela elaboração de métricas de usabilidade, utilidade, ou outras. Pelo contrário, pretendia-se verificar se, efetivamente, tal solução era possível, comprovando que 1) a informação adicional chega ao cliente, 2) é apresentada de acordo com a especificação dos metadados e 3) é usável pelo espetador. É também importante 4) verificar que os metadados se encontram sincronizados com a programação linear, mostrando a informação no período temporal correspondente.

3.6.1 Validação

Para validação do resultado obtido foram definidos alguns testes em que a solução desenvolvida deveria passar. Estes deveriam comprovar o cumprimentos dos requisitos definidos em 3.2.

Tabela 3.5: Casos de teste

ID	Descrição
T1	A aplicação chega ao equipamento recetor e executa.
T2	Os metadados são importados corretamente.
T3	Os metadados externos são importados corretamente.
T4	Os *templates* são importados corretamente.
T5	Os estilos e ficheiros JavaScript externos são importados e executam corretamente.
T6	Os eventos são registados com sucesso.
T7	Os eventos são recebidos no terminal recetor.
T8	As ações dos eventos são executadas corretamente.
T9	As teclas do telecomando são capturadas corretamente.
T10	É possível navegar com o telecomando.
T11	É possível abrir um link externo dentro da aplicação.

Tabela 3.6: Relação entre os requisitos e os testes

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
AP1		✓									
AP2			✓								
AP3				✓	✓						
P1		✓									
P2			✓								
P3				✓	✓						
B1				✓	✓						
B2	✓										
B3						✓	✓	✓			
E1						✓	✓	✓			
E2									✓		
E3									✓	✓	✓

Para a realização destes testes recorreu-se a metadados de exemplo, não havendo uma preocupação em atingir uma qualidade gráfica elevada ou proximidade com um caso real. O objetivo era comprovar que tecnicamente a solução era funcional.

3.6.2 Definição das características do caso de uso

Para demonstração numa situação próxima de um caso real, foi pensado um caso de uso que comprovasse a aplicabilidade da solução. Foi escolhido um caso de uso relacionado com publicidade, uma vez que é uma das áreas que maior interesse poderá ter para produtores de conteúdos.

Embora não tivessem sido usadas todas as funcionalidades disponíveis, estão patentes as principais: envio de uma aplicação para um equipamento recetor, sincronização com conteúdo linear e utilização de uma interface gráfica adaptada.

As anotações direcionadas para a publicidade podem ser divididas em duas vertentes:

- Informação complementar de carácter publicitário de um produto mostrado num programa de TV;
- Informação complementar de um anúncio publicitário.

Nesta dissertação, estas foram exploradas da seguinte forma: Foi mostrada informação complementar de um programa, nomeadamente o nome do produto, a marca, imagens, outra informação relevante e uma ligação para um site com informação. Posteriormente, durante um anúncio publicitário desse mesmo produto, além da informação anterior, foi mostrada informação acerca do anúncio, como informação do ano em que foi feito e a empresa que o produziu. Foi ainda mostrada informação relativa à música de fundo, nomeadamente o nome da faixa, o artista, a capa do álbum em que se insere e um *QR code* com uma ligação para uma loja de música.

Embora não teste todas as funcionalidades da solução, é possível usar o caso de uso para a realização de alguns testes definidos em 3.6.1:

Tabela 3.7: Relação entre o caso de uso e os testes

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
CU	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓

3.6.3 Metadados do caso de uso

Para a concretização do caso de uso foi escolhido um produto, o Renault Captur, e a respetiva campanha publicitária. Foi mostrada informação complementar acerca do mesmo durante um programa de análise de automóveis do site Carbuyer.co.uk e durante o anúncio da campanha "Capture life" de 2013, da agência de publicidade Publicis.

Como informação relevante a mostrar ao espetador, incluem-se o preço, uma galeria de fotografias, uma ligação para o mini-site da campanha que, embora não seja adaptado para utilização em *Smart TVs*, ilustra o conceito de complementaridade, e a informação da música de fundo, a música *Midnight City* dos *M83*.

O código MPEG-7 com as anotações do caso de uso pode ser consultado no anexo B.1.

3.6.4 Ambiente de teste

Para testar a aplicação foram usadas todas as ferramentas descritas em 2.6. Uma vez que se trata de uma aplicação HTML, com componentes especializados, é possível dividi-la e testar toda a lógica de importação dos metadados e *templates* separadamente da lógica do HbbTV®. Concluídos os testes da importação num *Browser*, foi usado o *plugin* FireHbbTV para verificar a

interação entre o Módulo HbbTV®e o Módulo dos *Templates* e Metadados. Após correção dos erros, foi testado o comportamento e a representação gráfica no emulador da Opera, uma vez que este contém um *Browser* semelhante ao usado nas *Smart TVs*. Em todos estes casos, como não foi possível a utilização de eventos, foi simulada a sua receção através de funções temporizadas do JavaScript. Estas funções foram programadas para invocarem o *handler* dos eventos, passando-lhe um objeto com a mesma estrutura daquele que é enviado pelo evento real.

Por fim, para se poderem testar os eventos e a aplicabilidade da solução no mundo real, foi usado o servidor da Avalpa para enviar o sinal para uma *Smart TV*, através de um modulador de RF. Foi testado o envio da aplicação através da Internet e através de DSM-CC, recorrendo-se à aplicação DVBinpector para verificar a conformidade dos *streams* gerados.

Tabela 3.8: Relação entre as ferramentas de teste e os testes

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
Browser		✓	✓	✓	✓			✓			
FireHbbTV		✓	✓	✓	✓			✓			
Opera		✓	✓	✓	✓			✓			✓
Avalpa	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Para a realização destes testes foram usados metadados de teste com informação de dois programas de TV (imagem, descrição e ligação para página externa). Esta informação era mostrada de uma forma simples e foi retirada da página oficial do programa respetivo. Primeiramente aparecia a imagem e dois botões com a ligação "ver descrição" e "visitar página". A primeira ligação mostrava o texto da descrição abaixo da imagem, juntamente com um botão "voltar". A segunda abria a página externa. Estas informações apareciam em momentos distintos, embora se sobrepusessem, permitindo verificar que era possível navegar entre os botões das anotações dos diferentes programas de TV. Foram ainda adicionadas mais três anotações que consistiam apenas numa frase, sendo mostradas no canto inferior esquerdo da imagem. A primeira anotação não continha informação temporal e era usada para garantir que era mostrada durante toda a emissão. A segunda continha a descrição em diversas línguas, comprovando que era mostrada a língua por defeito ou a do equipamento. A terceira provinha de um servidor externo e comprovava que a informação era obtida corretamente.

Para executar os testes através do servidor Avalpa, foi desenvolvido um *script bash* que automatiza todos os passos necessários. Em primeiro lugar era preciso gerar as tabelas de informação do DVB. O servidor Avalpa traz consigo exemplos de configuração das tabelas, pelo que apenas foi necessário personalizá-las. Através de um *script* Python, também ele parte do servidor, as tabelas foram convertidas num TS. Foi criado um *script* Python que analisa os metadados e, através de um outro *script* que faz parte do servidor, gera os TS referentes aos possíveis eventos. Existindo um objeto carrossel, também está disponível um *script* que faz a conversão em TS (oc-update.sh). A partir de todos estes elementos foi usado o módulo do servidor que multiplexa os vários TS num único (*tscbrmuxer*), posteriormente inserido em ciclo numa fila FIFO. Finalmente foi usado

o módulo de envio de dados para o modulador RF (*tsrfsend*), que recolhe o TS presente nessa fila. Manualmente, era possível enviar um evento chamando o *script* bash e indicando o número do evento que se pretendia. Este encarregava-se de o multiplexar no sinal para ser enviado.

3.6.4.1 Browser/Emulador/Plugin

As ferramentas de teste foram corridas sobre ambiente Windows e Mac OS, nas suas versões 8.1 e 10.9.3. O Browser utilizado foi o Mozilla Firefox versão 29, com o *plugin* Firebug versão 1.12.8, para testar o código JavaScript e HTML, e o plugin FireHbbTV, versão 1.3.9.

3.6.4.2 Equipamentos

Para os testes em ambiente real foi usado o modulador de RF OpenCaster special edition (Tx only), juntamente com uma *Smart T*V Samsung, modelo UE32ES5500VXXC.

3.6.4.3 Servidor

O servidor da Avalpa foi instalado na distribuição Linux Debian GNU/Linux, versão 7, com Python versão 2.7.3. Esta foi instalada numa máquina virtual com 3Gb de RAM e dois núcleos de processamento, a correr sobre VirtualBox num Macbook Pro, modelo 9.1, com Mac OS 10.9.3.

3.6.5 Resultados

Durante a execução dos testes houve a necessidade de correção de erros detetados, no entanto o resultado final foi positivo em todos eles.

O envio por DSM-CC permitiu executar, com sucesso, a aplicação no terminal de teste. No entanto, o arranque da aplicação pode ser demorado, mesmo que se trate de uma outra aplicação de teste, cujo conteúdo seja praticamente inexistente. O teste do envio pela Internet foi realizado numa rede local cablada. Apesar da velocidade de transmissão ser mais elevada, comparativamente com a taxa de envio usada no DSM-CC, a aplicação também não é carregada instantaneamente. Uma possível causa para este atrasado poderão ser as capacidades limitadas da televisão mas, principalmente, devido ao intervalo que existe no envio da tabela AIT. No entanto, nesta situação de teste o atraso no arranque nunca foi superior a cinco segundos.

Algo inesperado surgiu quando a aplicação foi enviada por DSM-CC. A mesma aplicação, que enviada pela Internet executou sem falhas, gerou um erro grave ao ser enviada por este método. Apesar de o conteúdo HTML enviado na aplicação ser declarado pelos testes da W3C[5] como válido, chegando ao terminal este era declarado como contendo erros. Entretanto o problema foi detetado e corrigido: as etiquetas HTML únicas necessitam de ser fechadas. Ou seja, enquanto uma etiqueta HTML normal é iniciada por "<*nome_da_etiqueta*>" e terminada por "</*nome_da_etiqueta*>", é aceite pelas normas que uma etiqueta simples apenas contenha "<*nome_da_etiqueta*>".

[5]Organização que estabelece os padrões para a Internet.

No entanto, tal não é aceite pelo *Browser* do HbbTV®. Será necessário fechar sempre a etiqueta simples escrevendo "<*nome_da_etiqueta*/>".

Um dos aspetos pensados inicialmente para tornar a aplicação mais flexível foi a possibilidade de passar parâmetros na invocação da aplicação. No entanto, após a sua introdução no URL da aplicação, na AIT, seja referente ao envio pela Internet ou DSM-CC, verificou-se que eram ignorados. Uma forma de contornar este problema seria a criação de um ficheiro XML de configuração que contivesse esses parâmetros e possivelmente as definições dos *templates* (uma vez que também podem ser considerados parâmetros de configuração).

Como foi indicado no estado da arte, mais precisamente em 2.5.1, não é possível definir com elevada precisão um evento do tipo "*do it now*". No entanto, uma precisão de décimos de segundo, como foi observado, é mais que adequada para aquilo que se pretendia com esta solução.

Um problema detetado na solução implementada é a navegação através do telecomando. As teclas cursor (cima, baixo, esquerda e direita) permitem alternar entre os elementos clicáveis. A ordem pela qual os elementos são acedidos é decidida automaticamente pelo *Browser* do equipamento recetor e nem sempre resultam na interface mais intuitiva. A alternativa passaria pela criação de um sistema de navegação em que se pudesse definir a ordem pela qual os elementos são acedidos. Por exemplo, um botão poderia conter a referência do elemento para o qual o foco se deve dirigir caso seja premida a tecla cima, a referência caso seja premida a tecla baixo, etc, e assim sucessivamente. Embora este fosse, à partida, um processo manual, iria dar alguma liberdade ao criador dos *templates*.

Uma funcionalidade interessante da aplicação é a possibilidade de abrir uma página de Internet. Claro que uma página destas teria que ser adaptada para televisores, mas o simples facto de ser possível iria permitir oferecer mais informação ao utilizador. Ao ser aberta uma ligação, o HbbTV®abre-a no seu próprio *Browser*, como se de outra aplicação se tratasse, em vez de abrir no *Browser* habitualmente disponível nas *Smart TVs*. Embora o resultado final não seja significativamente diferente, uma vez que a página é mostrada da mesma maneira, deixam de estar disponíveis funcionalidades como a possibilidade de guardar nos favoritos.

De modo a tentar acelerar o envio da aplicação, através da diminuição do tamanho da mesma, foi tentada a compressão do código. Foram usadas algumas ferramentas, como o JsMin. No entanto, esta diminuição resultou em erros semelhantes aos descritos anteriormente. Apesar disto, este é o caminho para a obtenção de uma aplicação mais leve, facilitando o envio e tornando-a mais eficiente.

As seguintes imagens ilustram o funcionamento da aplicação. Na 3.14 pode ser visto o aviso de informação complementar e na 3.15 a aplicação ativa (após utilização do botão vermelho do telecomando).

Figura 3.14: Aviso de conteúdo interativo disponível

Figura 3.15: Conteúdo complementar informativo de um programa de análise automóvel

A imagem 3.16 mostra o acesso a um site externo. Ao selecionar o site o utilizar é questionado se pretende continuar e é mostrado um QR Code que lhe permite aceder a essa página através de um dispositivo móvel.

Figura 3.16: Acesso a uma página de Internet externa

A imagem 3.17 mostra a informação complementar durante a transmissão do anúncio publicitário. Aqui pode ser vista a possibilidade de se saber mais informação acerca da música de fundo.

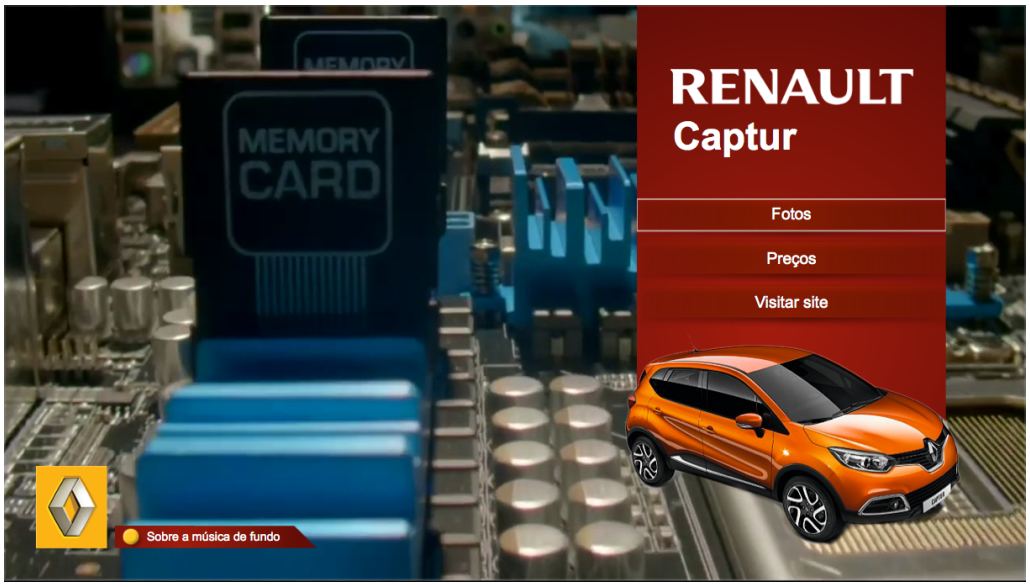

Figura 3.17: Conteúdo complementar informativo do anúncio publicitário

A imagem 3.18 apresenta a informação sobre a música de fundo, enquanto que a 3.19 mostra o QR Code com ligação para a loja de música que aparece quando o utilizador navega com o telecomando até à capa do álbum.

Figura 3.18: Informação acerca da música de fundo

Figura 3.19: QR Code com ligação para uma loja de música

3.7 Integração em ambiente real

Um dos objetivos primordiais desta dissertação, e fulcral para a sua concretização no mundo real, era evitar a necessidade de equipamento dedicado. Do lado do espetador significa poder usar um televisor (ou *set-top box*) que já possua. Desta forma não necessita de ter despesa para ter acesso a uma nova funcionalidade, e garante um mercado maior para alguém que pretenda disponibilizar o serviço, justificando o investimento. No entanto, é igualmente importante que o

investimento do lado do emissor seja reduzido. Mais concretamente, que seja possível utilizar o equipamento de que dispõe, mantendo o processo de emissão implementado inalterado.

Neste capitulo é estudada a forma de integração da solução na cadeia que engloba os intervenientes de produção e transmissão. Cada cadeia é única: cada estação de televisão tem o seu respetivo software, o seu acordo com produtores de conteúdos, etc. Desta forma não foi possível criar um componente de software que pudesse ser integrado e enviasse a aplicação e os eventos. Assim foi estudada a forma de adaptar a cadeia para que possa receber esta funcionalidade, distribuindo a complexidade.

3.7.1 Cadeia de produção

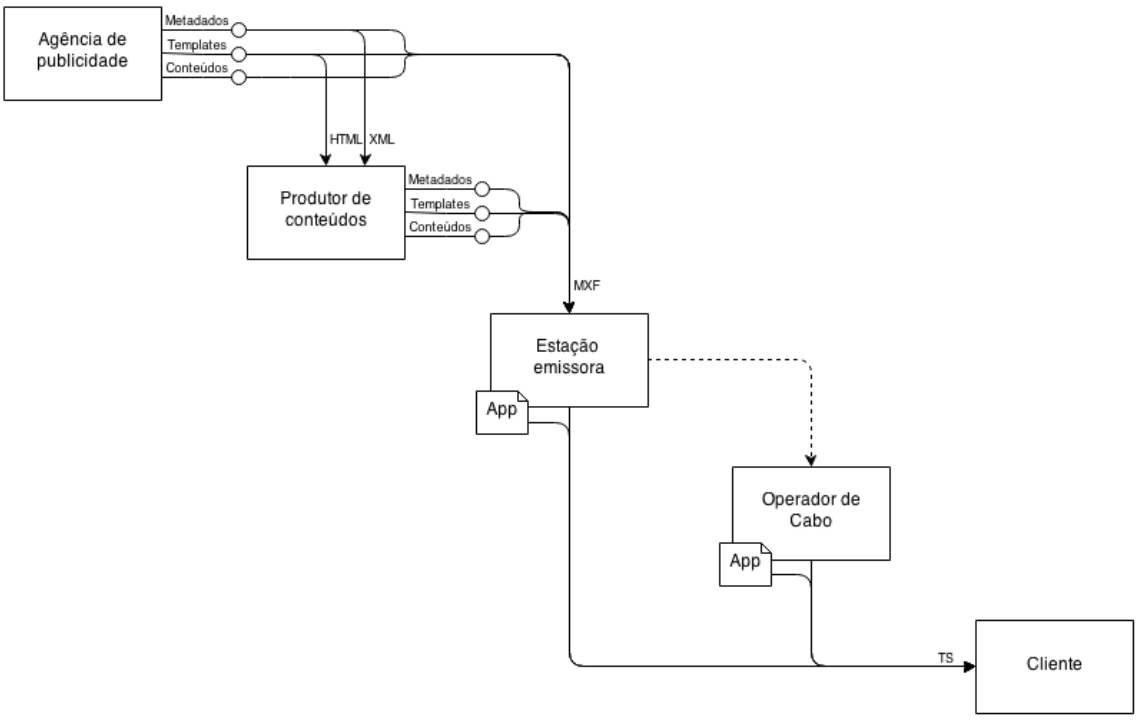

Figura 3.20: Cadeia de produção

Agência de publicidade No início da cadeia está a agência de publicidade. Esta tem como finalidade a criação do material de promoção/publicitário que será posteriormente enviado para os produtores de conteúdos, para que promovam o produto visado nos seus programas. Este material é enviado sob a forma de metadados e, possivelmente, *templates* para a exibição desses mesmos metadados. Uma agência de publicidade pode ainda ser a produtora de um anúncio publicitário que é enviado para uma estação emissora. O anúncio (conteúdo, seja ele vídeo, metadados, etc) é enviado juntamente com os metadados descritivos do serviço interativo e os *templates* para os exibir. O envio é feito através de MXF (ou outra forma acordada com a estação emissora).

Produtor de conteúdos O produtor de conteúdos gera programas de TV para uma estação emissora (séries, informação, entretenimento, etc). Estes programas podem ser acompanhados de publicidade/informação acerca de determinado produto que seja mostrado, que vem da agência de publicidade responsável pelo mesmo. Podem ainda ser acompanhados de outro tipo de informações, nomeadamente informação acerca do programa, atores/apresentadores, etc. Uma vez gerados todos os conteúdos (vídeo, áudio, metadados, *templates*) estes são enviados para uma estação emissora através do contentor MXF (ou outra forma acordada entre as partes).

Estação emissora A estação emissora recebe os conteúdos, tanto das agências de publicidade como produtoras. Cria a *playlist* da emissão e gera o *Transport Stream* que será enviado para o equipamento do espetador (cliente). Ao gerar o TS são multiplexados os conteúdos vídeo, áudio e as tabelas do DVB, juntamente com o Objecto Carrossel (se necessário) que transporta a aplicação, os metadados e *templates*. São também multiplexados os eventos "*do it now*" do DVB, gerados a partir da análise dos dados temporais dos metadados. Mais uma vez, os *templates* podem ser alterados, nomeadamente para estarem em conformidade com o manual de imagem do emissor.

Pode ainda haver a possibilidade de existir um acordo entre a estação emissora e os operadores de cabo, em que estes é que fazem a multiplexagem dos conteúdos e geram o TS. Nesse caso, os conteúdos são enviados para o operador e este é que insere a aplicação no TS. A forma de comunicação entre estes dois atores é variável, dependendo daquilo que é acordado entre as partes.

Operador de cabo Como referido no ponto anterior, havendo um acordo entre a estação emissora e os operadores, são estes últimos que fazem a multiplexagem dos conteúdos e inserem a aplicação no TS que é enviado para o cliente final.

3.7.2 Entrega de conteúdos no MXF

Sendo o MXF um formato dedicado para a partilha de conteúdos e sendo o formato usado pelas estações emissoras por excelência, desempenha um papel importante na cadeia de produção. Enviando os *templates* e os metadados juntamente com o conteúdo visual a que estão associados, é possível automatizar o processo de envio da aplicação para o espetador. Integrando estes elementos em essências, com chaves que identifiquem o tipo de conteúdo, permite que aplicações que desconheçam este conteúdo o ignorem, ao mesmo tempo que aquelas que o reconhecem o possam usar. Ao receber um ficheiro MXF com os *templates* e os metadados, a aplicação responsável pela integração terá como função inseri-los no Objeto Carrossel. Os metadados deverão então ser analisados para que sejam gerados os eventos necessários, e sejam enviados na marca temporal respetiva.

Os metadados a enviar podem ser definidos como uma essência em vez de serem integrados nos metadados do MXF, mantendo a sua estrutura e permitindo que sejam enviados como um todo com a aplicação. Isto, porque a aplicação como foi concebida recebe os metadados todos quando inicia.

3.7.3 Utilização em emissão em direto

Embora a solução encontrada seja para conteúdos transmitidos em diferido, existem diversos casos em que pode ter interesse integrá-la numa emissão em direto. Neste caso será necessário adaptar a aplicação, tarefa fácil de concretizar.

A aplicação está preparada para receber os metadados todos ao iniciar. Numa emissão em direto, será necessário que também o faça sempre que se justifique, ou seja, sempre que haja nova informação. Para isso será possível criar um evento semelhante ao evento *action* que sincroniza a emissão. Esse novo evento, sempre que enviado para o recetor, terá como *handler* uma função que irá atualizar os metadados a um servidor externo. Posteriormente será necessário enviar esse evento para que as anotações sejam mostradas.

Outra forma possível de implementar esta nova solução será semelhante ao método usado pelas aplicações de mensagens instantâneas disponíveis na Internet. Nestas, sempre que uma mensagem nova está disponível, as aplicações dos intervenientes são notificadas para que atualizem a lista de mensagens. Adaptando para a hipótese descrita, sempre que os metadados sejam alterados, uma notificação será enviada para a aplicação através da Internet. Deste modo será possível atualizar os dados.

Embora a aplicação desta dissertação não tenha sido alterada para a utilização em direto, foi criado um protótipo para comprovar esta última possibilidade. Recorrendo à plataforma Node.js, foram usados *websockets* entre uma aplicação web e um servidor. Sempre que um ficheiro JSON de metadados é alterado, o servidor notifica, via *websockets*, a aplicação web, para que esta atualize a sua informação. Apesar de funcional, desconhecem-se os problemas relativos à escalabilidade desta solução, uma vez que seria necessário estabelecer a comunicação com todas as aplicações que estivessem a correr em simultâneo. Usando um evento "*do it now*" seria uma solução mais escalável, embora obrigasse a que o servidor onde fosse feita a multiplexagem do sinal estivesse preparado para ser informado das atualizações em tempo real.

O seguinte diagrama ilustra a alteração sugerida.

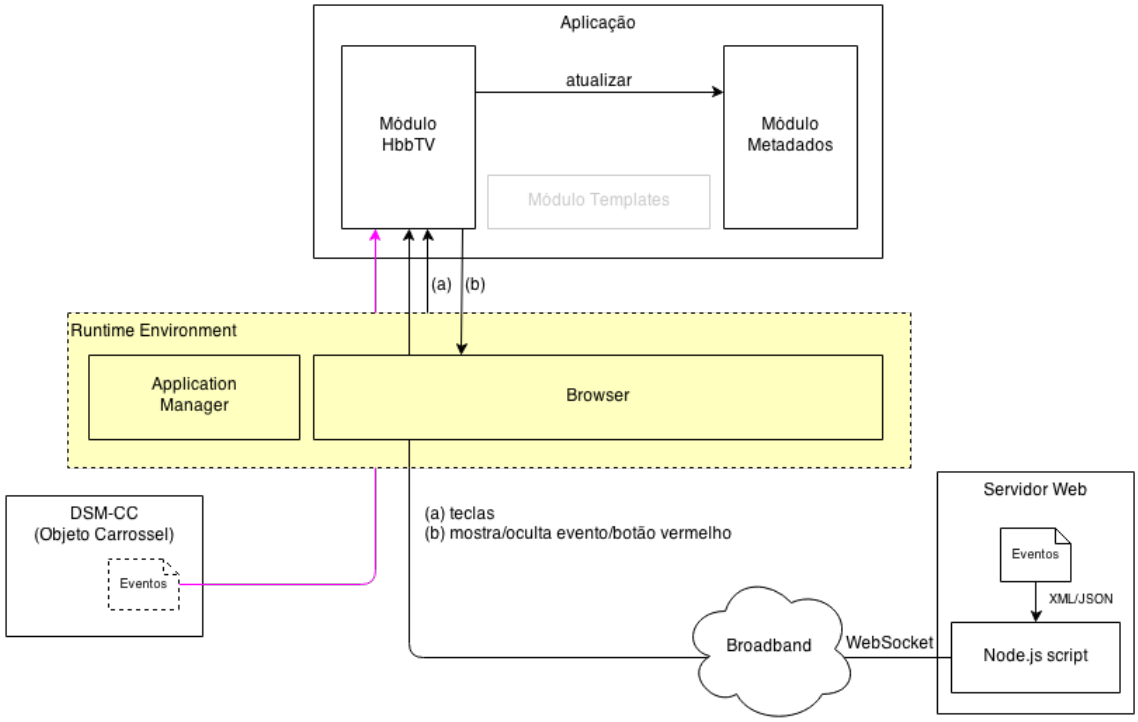

Figura 3.21: Esquema da adaptação necessária para funcionar em transmissões em direto

Capítulo 4

Conclusões

Esta dissertação tinha como objetivo desenvolver uma solução de interatividade que evitasse a complexidade e garantisse o sincronismo com a emissão linear que acompanha.

Primeiro foi definido o modelo de interatividade que se pretendia. Este consistiu num modelo semelhante ao que é atualmente usado no mercado, ou seja, que permitisse a distribuição unidirecional de informação complementar à emissão linear. Esta decisão direcionou o desenvolvimento da solução apresentada.

Os conteúdos interativos são caracterizados por metadados, que descrevem o conteúdo, e por *templates*, que definem a forma/aspeto como esses dados são apresentados. Para a definição dos metadados foi criada uma extensão ao MPEG-7, permitindo que as anotações que descrevem os conteúdos interativos coexistam com outras anotações próprias desta norma. Para facilitar a interoperabilidade foi introduzido o conceito de perfil. Um perfil é uma representação de uma temática com interesse para o mercado audiovisual (publicidade, desporto, teatro, cinema, etc), definindo um conjunto de variáveis que a descrevem, hierarquicamente organizadas. Nesta dissertação foram criados e/ou estudados perfis para a área da publicidade e do desporto. Uma vez que existem normas para a anotação destas áreas, a solução desenvolvida usou-as como base e integrou-as na extensão criada. Foram ainda implementados mecanismos que facilitam a criação de *templates*, nomeadamente através da disponibilização de uma função que permite o acesso às anotações dos metadados.

Para que os metadados e os *templates* que caracterizam os conteúdos interativos sejam combinados e apresentados ao espetador, foi desenvolvida uma aplicação em HTML que é enviada através da norma HbbTV®. O envio tanto pode ser feito através do DSM-CC, multiplexado na transmissão, como através da Internet. Esta aplicação importa os *templates* e os metadados que a acompanham, converte os últimos numa estrutura simplificada e combina-os com os *templates*. Caso os metadados referenciem anotações externas, importa-os e combina-os com as outras anotações, como se de uma estrutura coesa e única se tratasse.

Para que os conteúdos sejam apresentados ao utilizador final, foi implementado o sistema de eventos de *stream*. Do lado do emissor estes são multiplexados no sinal, tendo a aplicação a função de os receber e apresentar/ocultar as anotações respetivas.

4.1 Satisfação dos objetivos

Para verificar que eram cumpridos os objetivos da dissertação foi definido um conjunto de testes aos quais a aplicação foi submetida. Após a sua execução, verificou-se que a aplicação era enviada e recebida com sucesso no cliente, que todos os elementos necessários eram importados, que era possível apresentar a informação ao espetador e que esta estava sincronizada temporalmente com a emissão.

Todos os testes foram efetuados numa *Smart TV* já hoje disponível no mercado, garantindo a compatibilidade com equipamentos existentes. Este aspeto é extremamente importante uma vez que garante um mercado/público alvo maior do que aquele que uma solução completamente nova conseguiria. Assim a solução terá mais interesse para quem a disponibiliza e para quem é consumidor, uma vez que não necessita de adquirir novos equipamentos para usufruir de uma nova funcionalidade.

A realização dos testes com sucesso também comprovou a aplicabilidade técnica do conceito. Devido à multitude de hipóteses possíveis para a implementação de uma solução de envio da aplicação por parte das estações emissoras, este componente não foi implementado. No entanto, foi estudada a forma de integrar a solução nesta cadeia. O facto de as alterações necessárias serem reduzidas é uma vantagem para a sua real implementação. Isto porque os processos e equipamentos disponíveis poderão ser adaptados para comportarem a nova solução.

Além dos custos reduzidos tanto para o espetador como para o emissor, a solução encontrada poderá traduzir-se num melhor serviço prestado. Desta forma poderá ser usada para a obtenção de novas receitas que podem ser canalizadas para a melhoria dos conteúdos. Esta melhoria beneficia o espetador, que também passa a dispor de novas funcionalidades que enriquecem a sua experiência de consumo de televisão.

O facto de, na sincronização serem considerados todos os eventos enviados de modo a verificar a existência daqueles que, embora passados continuam válidos, garante que a experiência de utilização seja uniforme. Este aspeto é importante para garantir uma melhor experiência de utilização e, em termos comerciais, para garantir que os conteúdos disponibilizados efetivamente chegam a todos os possíveis espetadores.

Pode ainda concluir-se que a solução é flexível. Permite uma elevada personalização dos conteúdos e da interface, permite enviar a aplicação de duas maneiras distintas, de acordo com os requisitos técnicos e comerciais dos emissores, e permite uma distribuição e colaboração entre todos os intervenientes da cadeia de produção no desenvolvimento dos serviços interativos. Referir ainda que, embora na introdução fosse considerada a necessidade de utilização de infraestruturas de suporte como aspeto negativo, o envio da aplicação pela Internet também requer que exista um servidor onde esta possa ser alojada. No entanto, este tipo de infraestruturas já se encontra

garantidamente implementado, uma vez que são necessárias para disponibilizar uma página de Internet das estações emissoras.

Todos os aspetos aqui referidos contribuem para aferir o efetivo cumprimento dos objetivos do projeto.

4.2 Trabalho futuro

A partir da análise da execução da aplicação, e de acordo com o que foi descrito em 3.6.5, o arranque da aplicação é demorado. Este será um dos aspetos a melhorar. Para fazê-lo poderá ser equacionada novamente a questão da minimização e a otimização da aplicação. O primeiro aspeto será importante para que a aplicação seja enviada em menos tempo para o recetor. O segundo para que a importação dos elementos seja mais célere, apresentando mais rapidamente os conteúdos ao espetador. Para otimizar poderá ser considerada uma distribuição temporal da importação dos eventos, ou seja, em vez de carregar tudo no arranque, apenas as informações essenciais deverão ser importadas inicialmente. Isto implica que o ficheiro de anotação esteja dividido em secções, pelo que deverá ser uma funcionalidade complementada do lado do emissor. Do lado do emissor também poderá ser relevante que apenas as anotações válidas no presente, ou que o serão no futuro, sejam enviadas para o cliente. Neste caso, o emissor deverá analisar o ficheiro de metadados, removendo as anotações que não são válidas.

Uma vez que a aplicação também pode ser enviada pela Internet, esta poderia ser mais rapidamente mostrada ao espetador se algum do trabalho feito fosse delegado para o servidor. Ou seja, as anotações externas poderiam já ir embebidas nos metadados, ou, inclusive, a aplicação poderia transportá-los simplificados na sua estrutura interna, com os templates necessários embebidos no código. Esta é outra forma de otimizar o envio da aplicação, sendo algo a desenvolver futuramente.

Um outro aspeto apontado como negativo é a navegação entre os elementos da página. Como é referido em 3.6.5, este problema poderá ser corrigido desenvolvendo uma ferramenta que permita ao produtor dos *templates* definir a ordem com que os elementos são acedidos.

Outra funcionalidade interessante, para quem tem potencial interesse em disponibilizar a solução, é a possibilidade de atualizar os metadados durante a execução da aplicação. Por oposição ao modelo atual, no qual os metadados são carregados no instante em que a aplicação arranca, esta melhoria irá permitir que a aplicação seja usada numa transmissão em direto.

Finalmente, de forma a colmatar uma fragilidade da aplicação, poder-se-á investigar uma abordagem que permita intercalar os conteúdos interativos com anúncios publicitários. Imagine-se o seguinte cenário: são criados metadados que acompanham a transmissão de um determinado programa de entretenimento. Este tipo de programas muito provavelmente será intercalado com publicidade da estação emissora. Imagine-se que existe uma informação que é apresentada antes do intervalo comercial e deve continuar após a sua conclusão. Neste caso, a estação emissora poderá retirar a aplicação da transmissão e obrigar o equipamento recetor a destruir a versão que possui. Ao recomeçar o programa de entretenimento será necessário que a aplicação seja novamente sinalizada no sinal, o que obrigará a recarregar toda a informação no arranque. Uma hipótese mais

simples será alterar a aplicação de modo a que esta possa receber um evento que a oculte completamente. Assim, durante o intervalo esse evento será enviado, desativando a aplicação no cliente. Ao voltar ao programa, a aplicação receberá outro evento que a ativará, evitando que os dados sejam carregados novamente. Embora simples, esta solução não comporta a existência de outra hipótese: a possibilidade do anúncio comercial ser também ele acompanhado de informação complementar. Nesta situação ou se enviará uma nova aplicação com os novos metadados e conteúdos (a única forma possível pela implementação atual) ou se obrigará a aplicação a carregar novos metadados durante a sua execução (como referido no parágrafo anterior). No primeiro caso, uma vez mais a aplicação terá que carregar tudo de novo ao ser reiniciada. No segundo caso, a aplicação enviada manter-se-á sempre em funcionamento enquanto o utilizador esteja sintonizado na emissão, alterando os metadados e conteúdos de acordo com a informação que vá sendo enviada. Esta solução permitirá alternar entre os metadados dos diferentes programas de TV e anúncios, mas obrigará a uma implementação mais complexa do lado do emissor.

Referências

[BBC] BBC. Tv application layer. URL: http://fmtvp.github.io/tal/index.html [último acesso em 11/01/2014].

[BVDWH+03] I. Burnett, R. Van De Walle, K. Hill, J. Bormans e F. Pereira. Mpeg-21: goals and achievements. *MultiMedia, IEEE*, 10(4):60–70, 2003.

[Dev02] B. Devlin. Mxf - the material exchange format. *EBU Technical Review*, 2002.

[DVB14] DVB. Dvb worldwide, 2014. URL: https://www.dvb.org/news/worldwide [último acesso em 13/06/2014].

[EBU10] EBU. Metadata for the file exchange of advertising material (egtameta), 2010.

[EBU14] EBU. Ebu core metadata set (ebucore), 2014.

[ETS98] ETSI. Digital video broadcasting (dvb); specification for service information (si) in dvb systems, 1998.

[ETS10] ETSI. Digital video broadcasting (dvb); signalling and carriage of interactive applications and services in hybrid broadcast/broadband environments, 2010.

[ETS11] ETSI. Broadcast and on-line services: Search, select, and rightful use of content on personal storage systems ("tv-anytime"); part 3: Metadata; sub-part 1: Phase 1 - metadata schemas, 2011.

[ETS12] ETSI. Hybrid broadcast broadband tv, 2012.

[Fer10] P. Ferreira. Mxf - a progress report. *EBU Technical Review*, 2010.

[For14] Open IPTV Forum. Release 2 specification; volume 5 - declarative application environment, 2014.

[Gro05] The Moving Picture Experts Group. Mpeg-2 systems, 2005. URL: http://mpeg.chiariglione.org/standards/mpeg-2/systems [último acesso em 03/06/2014].

[GZO+10] D. Goergen, J. Zoric, J. O'Connell, O. Friedrich e B. Zachey. A session model for cross-domain interactive multi-user iptv. *IEEE Communications Society*, 2010.

[Han13] Handlebars. Handlebars.js, 2013. URL: http://handlebarsjs.com [último acesso em 05/06/2014].

[HS] R. L. Haskin e F. L. Stein. A system for the delivery of interactive television programming. In *Compcon '95.'Technologies for the Information Superhighway', Digest of Papers.*, pages 209–215.

[IPT14] IPTC. Iptc web, 2014. URL: http://www.iptc.org/ [último acesso em 02/06/2014].

[ISO98] ISO/IEC. Information technology – generic coding of moving pictures and associated audio information – part 6: Extensions for dsm-cc, 1998.

[Kov12] A. Kovalick. A quick tour of wrappers and mxf. 2012.

[Kra] Tomas Kratochvil. From analog to digital television - the common way how to digitize european broadcasting. In *History of Telecommunications Conference, 2008. HISTELCON 2008. IEEE*, pages 164–169.

[Mar03] Robert Cecil Martin. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, 2003.

[Mor11] S. Morris. How to become an expert in dsm-cc, 2011. URL: http://www.interactivetvweb.org/tutorials/dtv_intro/dsmcc [último acesso em 11/06/2014].

[Mx14] MIT-xperts. Tests for hbbtv implementations, 2014. URL: https://github.com/mitxp/HbbTV-Testsuite [último acesso em 31/04/2014].

[Ned14] HbbTV Forum Nederland. Overview of interactive television services according to the hbbtv standard in europe. Report, 27/03/2014 2014.

[NL99] Frank Nack e A. T. Lindsay. Everything you wanted to know about mpeg-7. 1. *MultiMedia, IEEE*, 6(3):65–77, 1999.

[Sam14] Samsung. Multi-screen sdk, 2014. URL: http://www.samsungdforum.com/Guide/d30/index.html [último acesso em 10/01/2014].

[SFSP01] Chang Shih-Fu, T. Sikora e A. Purl. Overview of the mpeg-7 standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):688–695, 2001.

[SS06] J. R. Smith e P. Schirling. Metadata standards roundup. *IEEE MultiMedia*, 13(2):84–88, 2006.

[Sta04] International Organisation for Standardisation. Mpeg-7 overview, 2004.

[Sta14] DTV Status. Atsc, dtmb, dvb-t/dvb-t2, and isdb-t, 2014. URL: http://en.dtvstatus.net/ [último acesso em 13/06/2014].

[Wil00] J. Wilkinson. The smpte data coding protocol and dictionaries. *SMPTE Journal*, pages 579–586, 2000.

[Yos02] Toshiro Yoshimura. Overview of mpeg-2 systems. *Technologies and Services on Digital Broadcasting*, (11):16–23, 2002.

Anexo A

Documentos XML

A.1 Extensão do MPEG-7

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
  targetNamespace="urn:mpeg:mpeg7:schema:2004"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <include schemaLocation="new-schema-advertising.xsd"/>

  <redefine schemaLocation="mpeg7-v2.xsd">
    <complexType name="VideoSegmentType">
      <complexContent>
        <extension base="mpeg7:VideoSegmentType">
          <choice minOccurs="1" maxOccurs="1">
            <element name="ExternalAnnotations" type="mpeg7:
                ExternalAnnotationsType"/>
            <element name="Annotations" type="mpeg7:AnnotationsBaseType
                "/>
          </choice>
        </extension>
      </complexContent>
    </complexType>
  </redefine>

  <complexType name="AnnotationType">
    <complexContent>
      <extension base="mpeg7:TermUseType">
```

```
24        <sequence>
25          <element name="Annotation" type="mpeg7:AnnotationType"
              minOccurs="0" maxOccurs="unbounded"/>
26        </sequence>
27        <attribute name="typeLabel" type="string" use="required"/>
28      </extension>
29    </complexContent>
30  </complexType>
31
32  <complexType name="AnnotationsBaseType">
33    <complexContent>
34      <extension base="mpeg7:DSType">
35        <sequence>
36          <element name="Position" type="mpeg7:Position" minOccurs="0"
              maxOccurs="1"/>
37          <element name="Annotation" type="mpeg7:AnnotationType"
              minOccurs="0" maxOccurs="unbounded"/>
38        </sequence>
39       <attribute name="template"/>
40      </extension>
41    </complexContent>
42  </complexType>
43
44  <complexType name="ExternalAnnotationsType">
45    <attribute name="url"/>
46  </complexType>
47
48  <complexType name="Position">
49    <sequence>
50      <element name="X" type="integer" minOccurs="0" maxOccurs="1"/>
51      <element name="Y" type="integer" minOccurs="0" maxOccurs="1"/>
52      <element name="Width" type="integer" minOccurs="0" maxOccurs="1
          "/>
53      <element name="Height" type="integer" minOccurs="0" maxOccurs="
          1"/>
54    </sequence>
55     <attribute name="type">
56      <simpleType>
57        <restriction base="string">
58          <pattern value="Absolute|Relative"/>
```

```
59        </restriction>
60      </simpleType>
61     </attribute>
62     <attribute name="origin">
63      <simpleType>
64        <restriction base="string">
65          <pattern value="TopLeft|TopRight|BottomLeft|BottomRight"/>
66        </restriction>
67      </simpleType>
68     </attribute>
69   </complexType>
70
71 </schema>
```

Código A.1: *Schema* da Extensão do MPEG-7 declarado no ficheiro new-schema.xsd

A.2 Perfil de Publicidade

```
1  <!-- ################################################ -->
2  <!-- Advertising profile -->
3  <!-- ################################################ -->
4
5  <complexType name="AdvertisingAnnotationType">
6    <complexContent>
7      <extension base="mpeg7:AnnotationsBaseType">
8        <sequence>
9          <group ref="mpeg7:coreMetadataGroup"/>
10         <element name="BackgroundMusic" type="mpeg7:backgroundMusic"
               minOccurs="0" maxOccurs="unbounded"/>
11         <element name="Product" type="mpeg7:product" minOccurs="1"
              maxOccurs="unbounded"/>
12         <element name="Picture" type="mpeg7:picture" minOccurs="0"
              maxOccurs="unbounded"/>
13       </sequence>
14     </extension>
15   </complexContent>
16 </complexType>
```

Código A.2: *Schema* do Perfil de Publicidade declarado no ficheiro da extensão do MPEG-7

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:ebucore="urn:ebu:metadata-schema:ebuCore_2010"
 targetNamespace="urn:mpeg:mpeg7:schema:2004"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
  <import namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
 <import namespace="http://purl.org/dc/elements/1.1/"
    schemaLocation="egtaMETA/simpledc20021212.xsd"/>
 <import namespace="urn:ebu:metadata-schema:ebuCore_2010"
    schemaLocation="egtaMETA/EBU_CORE_20100820.xsd"/>
 <include schemaLocation="mpeg7-v2.xsd"/>

 <complexType name="backgroundMusic">
  <complexContent>
   <extension base="mpeg7:coreMetadataType">
    <sequence>
     <element name="artist" type="string" minOccurs="0" maxOccurs
        ="1"/>
     <element name="track" type="integer" minOccurs="0" maxOccurs
        ="1"/>
     <element name="album" type="mpeg7:TermUseType" minOccurs="0"
         maxOccurs="1"/>
     <element name="albumID" type="string" minOccurs="0"
        maxOccurs="1"/>
     <element name="cover" type="anyURI" minOccurs="0" maxOccurs=
        "1"/>
     <element name="arranger" type="string" minOccurs="0"
        maxOccurs="1"/>
     <element name="composer " type="string" minOccurs="0"
        maxOccurs="1"/>
     <element name="ISRC" type="string" minOccurs="0" maxOccurs="
        1"/>
     <element name="genre" type="string" minOccurs="0" maxOccurs=
        "1"/>
     <element name="genreCode" minOccurs="0" maxOccurs="1">
      <simpleType>
```

```
28            <restriction base="integer">
29              <minInclusive value="1"/>
30              <maxInclusive value="28"/>
31            </restriction>
32          </simpleType>
33        </element>
34        <element name="path" type="anyURI" minOccurs="0" maxOccurs="
              1"/>
35      </sequence>
36    </extension>
37  </complexContent>
38 </complexType>
39
40 <complexType name="product">
41   <sequence>
42     <element name="productName" type="mpeg7:TermUseType" minOccurs=
           "1"/>
43     <element name="productId" type="ebucore:identifierType"
           minOccurs="0"/>
44     <element name="description" type="mpeg7:TermUseType" minOccurs=
           "0"/>
45     <element name="logo" type="anyURI" minOccurs="0" maxOccurs="1"/
           >
46     <element name="brand" type="mpeg7:brand" minOccurs="0"/>
47     <element name="path" type="anyURI" minOccurs="0" maxOccurs="1"/
           >
48     <element name="photoAlbum" type="mpeg7:photoAlbumType"
           minOccurs="0" maxOccurs="1"/>
49   </sequence>
50 </complexType>
51
52 <complexType name="picture">
53   <sequence>
54     <element name="name" type="string"/>
55     <element name="description" type="mpeg7:coreMetadataType"/>
56     <element name="container" type="string">
57       <![CDATA[]]>
58     </element>
59   </sequence>
60 </complexType>
```

```
61
62  <complexType name="photoAlbumType">
63    <sequence>
64      <element name="name" type="string" minOccurs="0"/>
65      <element name="photo" type="mpeg7:photoType" minOccurs="1"
            maxOccurs="unbounded"/>
66    </sequence>
67  </complexType>
68
69  <complexType name="photoType">
70    <sequence>
71      <element name="name" type="string" minOccurs="0"/>
72      <element name="description" type="mpeg7:coreMetadataType"
            minOccurs="0"/>
73      <element name="path" type="anyURI" minOccurs="1" maxOccurs="1"/
            >
74    </sequence>
75  </complexType>
76
77  <complexType name="brand">
78    <sequence>
79      <element name="brandName" type="mpeg7:TermUseType" minOccurs="0
            "/>
80      <element name="description" type="mpeg7:TermUseType" minOccurs=
            "0"/>
81      <element name="brandId" type="ebucore:identifierType" minOccurs
            ="0"/>
82      <element name="logo" type="anyURI" minOccurs="0" maxOccurs="1"/
            >
83      <element name="path" type="anyURI" minOccurs="0" maxOccurs="1"/
            >
84    </sequence>
85  </complexType>
86
87  <complexType name="coreMetadataType">
88    <group ref="mpeg7:coreMetadataGroup"/>
89  </complexType>
90
91  <group name="coreMetadataGroup">
92    <sequence>
```

```
93      <sequence>
94        <element name="title" type="mpeg7:TermUseType" minOccurs="0"
              maxOccurs="unbounded"/>
95        <element name="alternativeTitle" type="mpeg7:TermUseType"
              minOccurs="0" maxOccurs="unbounded"/>
96      </sequence>
97      <element name="creator" type="ebucore:entityType" minOccurs="0"
            maxOccurs="unbounded"/>
98      <element name="subject" type="mpeg7:TermUseType" minOccurs="0"
            maxOccurs="unbounded"/>
99      <element name="description" type="mpeg7:TermUseType" minOccurs=
            "0" maxOccurs="unbounded"/>
100     <element name="publisher" type="ebucore:entityType" minOccurs="
            0" maxOccurs="unbounded"/>
101     <element name="contributor" type="ebucore:entityType" minOccurs
            ="0" maxOccurs="unbounded"/>
102     <element name="date" type="ebucore:dateType" minOccurs="0"
            maxOccurs="unbounded"/>
103     <element name="type" type="ebucore:typeType" minOccurs="0"
            maxOccurs="unbounded"/>
104     <element name="format" type="ebucore:formatType" minOccurs="0"
            maxOccurs="unbounded"/>
105     <element name="identifier" type="ebucore:identifierType"
            minOccurs="0" maxOccurs="unbounded"/>
106     <element name="language" type="ebucore:languageType" minOccurs=
            "0" maxOccurs="unbounded"/>
107     <element name="rights" type="ebucore:rightsType" minOccurs="0"
            maxOccurs="unbounded"/>
108     <element name="version" type="dc:elementType" minOccurs="0"/>
109    </sequence>
110  </group>
111
112 </schema>
```

Código A.3: *Schema* do Perfil de Publicidade declarado no ficheiro new-schema-advertising.xsd

Anexo B

Ficheiros dos testes

B.1 Metadados do caso de uso

```
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ebucore="urn:ebu:metadata-schema:ebuCore_2010"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <MediaTime>
          <MediaRelTimePoint>PT5S</MediaRelTimePoint>
          <MediaDuration>PT15S</MediaDuration>
        </MediaTime>

        <Annotations xsi:type="AdvertisingAnnotationType" template="
            programm_car">
          <Position type="Absolute" origin="BottomRight">
            <X>50</X>
            <Y>55</Y>
          </Position>

          <Annotation typeLabel="prices">
            <Annotation typeLabel="tce90">
              <Name>TCe 90</Name>
              <Annotation typeLabel="expression">
```

```
25              <Name>Expression</Name>
26              <Definition>15.600</Definition>
27            </Annotation>
28            <Annotation typeLabel="exclusive">
29              <Name>Exclusive Energy</Name>
30              <Definition>18.560</Definition>
31            </Annotation>
32          </Annotation>
33        </Annotation>
34
35        <Annotation typeLabel="textLogo">
36          <Annotation typeLabel="white">
37            <Definition>templates/img/logo_text_renault_white.png</
                Definition>
38          </Annotation>
39          <Annotation typeLabel="color">
40            <Definition>templates/img/logo_text_renault_color.png</
                Definition>
41          </Annotation>
42        </Annotation>
43
44        <Annotation typeLabel="carImg">
45          <Definition>templates/img/captur3.png</Definition>
46        </Annotation>
47
48        <title>
49          <Definition>Capture life</Definition>
50        </title>
51        <creator>
52          <ebucore:organisationDetails>
53            <ebucore:organisationName>Publicis</ebucore:
                organisationName>
54          </ebucore:organisationDetails>
55        </creator>
56        <description>
57          <Definition>Campanha publicitaria de apresentacao do novo
              crossover da Renault.</Definition>
58        </description>
59        <date>
60          <dc:date>12/03/2013</dc:date>
```

```
61          </date>
62
63          <BackgroundMusic>
64            <title>
65              <Definition>Midnight City</Definition>
66            </title>
67            <artist>M83</artist>
68            <track>1</track>
69            <album>
70              <Definition>Midnight City</Definition>
71            </album>
72            <cover>templates/img/album_cover.jpg</cover>
73            <path>https://itunes.apple.com/us/album/midnight-city-ep/
                id828137168</path>
74          </BackgroundMusic>
75
76          <Product>
77            <productName>
78              <Definition>Captur</Definition>
79            </productName>
80            <description>
81                <Definition>O Captur reflete o concept car "CAPTUR"
                    que ilustra a segunda etapa do ciclo da vida e o
                    Design da Renault designou por "Explorar". Este
                    crossover divertido e atletico convida "a explorar
                    o mundo a dois".</Definition>
82              </description>
83            <brand>
84              <brandName>
85                <Definition>Renault</Definition>
86              </brandName>
87              <description>
88                    <Definition xml:lang="en">Renault S.A. is a French
                        multinational vehicle manufacturer established
                        in 1899. The company produces a range of cars
                        and vans, and in the past, trucks, tractors,
                        tanks, buses/coaches and autorail vehicles.</
                        Definition>
89                    <Definition xml:lang="pt">A Renault S.A. e' um
                        fabricante frances de veiculos fundada em 1898
```

```
                    por Louis Renault. Produz desde automoveis
                    pequenos e medios, furgoes, autocarros e camioes
                    .</Definition>
90              </description>
91            <logo>templates/img/logo_renault.jpg</logo>
92            <path>http://www.renault.com/</path>
93          </brand>
94          <path>http://www.captur.renault.com/</path>
95          <photoAlbum>
96            <name>album1</name>
97            <photo>
98              <name>foto1</name>
99              <path>templates/img/fotos/1.jpg</path>
100           </photo>
101           <photo>
102             <name>foto2</name>
103             <path>templates/img/fotos/2.jpg</path>
104           </photo>
105           <photo>
106             <name>foto3</name>
107             <path>templates/img/fotos/3.jpg</path>
108           </photo>
109           <photo>
110             <name>foto4</name>
111             <path>templates/img/fotos/4.jpg</path>
112           </photo>
113         </photoAlbum>
114       </Product>
115
116     </Annotations>
117   </Video>
118  </MultimediaContent>
119 </Description>
120 </Mpeg7>
```

Código B.1: Metadados do caso de uso

B.2 Tabelas DVB

```

#
# Shared values
#

demo_transport_stream_id = 1# demo value, an official value should
   be demanded to dvb org
demo_original_network_id = 1# demo value, an official value should
   be demanded to dvb org
demo_service_id = 1# demo value
pmt1_pid= 1031
ait1_pid = 2001
ste1_pid = 2002
dsmcc_association_tag = 0xB
dsmcc_carousel_id = 1
stream_event_component_tag = 0xD;



# parameters reported into ths AIT to signalize a broadcast
   application.
appli_name = "app"
appli_root = "http://192.168.0.22/Tese/App/" #URL of App
   <<<<<<<<<<<<<<<<<<<<<<<<
appli_path = "App/index.html"
organisationId_1 = 10# this is a demo value
applicationId_1 = 1001# this is a demo value. This number
   corresponds to a HbbTV trusted application
organisationId_2 = 11
applicationId_2 = 1002

# below, the video and audio PIDs reported into the PMT
video1_pid = 2064
audio1_pid = 2068
carousel_pid = 2003



#
# Network Information Table
```

```
35 # this is a basic NIT with the minimum desciptors, OpenCaster has a
      big library ready to use
36 #
37
38 nit = network_information_section(
39  network_id = 1,
40       network_descriptor_loop = [
41      network_descriptor(network_name = "OpenHbb",),
42           ],
43  transport_stream_loop = [
44      transport_stream_loop_item(
45    transport_stream_id = demo_transport_stream_id,
46    original_network_id = demo_original_network_id,
47    transport_descriptor_loop = [
48       service_list_descriptor(
49      dvb_service_descriptor_loop = [
50         service_descriptor_loop_item(
51       service_ID = demo_service_id,
52       service_type = 1,# digital tv service type
53         ),
54      ],
55       ),
56    ],
57      ),
58    ],
59       version_number = 1,# you need to change the table number every
             time you edit, so the decoder will compare its version
           with the new one and update the table
60       section_number = 0,
61       last_section_number = 0,
62       )
63 #
64 # Program Association Table (ISO/IEC 13818-1 2.4.4.3)
65 #
66
67 pat = program_association_section(
68  transport_stream_id = demo_transport_stream_id,
69       program_loop = [
70         program_loop_item(
71         program_number = demo_service_id,
```

```
72        PID = pmt1_pid,
73          ),
74          program_loop_item(
75          program_number = 0,# special program for the NIT
76        PID = 16,
77          ),
78        ],
79        version_number = 1,# you need to change the table number every
              time you edit, so the decoder will compare its version
             with the new one and update the table
80        section_number = 0,
81        last_section_number = 0,
82        )
83
84  #
85  # Program Map Table (ISO/IEC 13818-1 2.4.4.8)
86  # this is PMT for HbbTV interactive applications
87  #
88
89  pmt = program_map_section(
90    program_number = demo_service_id,
91    PCR_PID = video1_pid, # usualy the same than the video
92    program_info_descriptor_loop = [],
93    stream_loop = [
94      stream_loop_item(
95        stream_type = 2,# mpeg2 video stream type
96        elementary_PID = video1_pid,
97        element_info_descriptor_loop = []
98      ),
99      stream_loop_item(
100       stream_type = 3,# mpeg2 audio stream type
101       elementary_PID = audio1_pid,
102       element_info_descriptor_loop = []
103     ),
104     stream_loop_item(
105       stream_type = 5,# AIT stream type
106       elementary_PID = ait1_pid,
107       element_info_descriptor_loop = [
108          application_signalling_descriptor(
109         application_type = 0x0010, # HbbTV service
```

```
110       AIT_version = 1,# current ait version
111         ),
112       ]
113     ),
114     stream_loop_item(
115       stream_type = 12,# Stream Event stream type
116       elementary_PID = ste1_pid,
117       element_info_descriptor_loop = [
118         stream_identifier_descriptor(
119           component_tag = stream_event_component_tag,
120         ),
121       ]
122     )
123   ],
124       version_number = 1,# you need to change the table number every
              time you edit, so the decoder will compare its version
             with the new one and update the table
125       section_number = 0,
126       last_section_number = 0,
127       )
128
129   ##############################################
130
131 pmt_dsmcc = program_map_section(
132   program_number = demo_service_id,
133   PCR_PID = video1_pid, # usualy the same than the video
134   program_info_descriptor_loop = [],
135   stream_loop = [
136     stream_loop_item(
137       stream_type = 2,# mpeg2 video stream type
138       elementary_PID = video1_pid,
139       element_info_descriptor_loop = []
140     ),
141     stream_loop_item(
142       stream_type = 3,# mpeg2 audio stream type
143       elementary_PID = audio1_pid,
144       element_info_descriptor_loop = []
145     ),
146     stream_loop_item(
147       stream_type = 5,# AIT stream type
```

```
148     elementary_PID = ait1_pid,
149     element_info_descriptor_loop = [
150       application_signalling_descriptor(
151       application_type = 0x0010, # HbbTV service
152       AIT_version = 1,# current ait version
153       ),
154     ]
155   ),
156   stream_loop_item(
157     stream_type = 12,# Stream Event stream type
158     elementary_PID = ste1_pid,
159     element_info_descriptor_loop = [
160       stream_identifier_descriptor(
161         component_tag = stream_event_component_tag,
162       ),
163     ]
164   ),
165   stream_loop_item(
166     stream_type = 11,# DSMCC stream type
167     elementary_PID = carousel_pid,
168     element_info_descriptor_loop = [
169       # a number of descriptors follow specifying the DSMCC
            properites
170       association_tag_descriptor(
171         association_tag = dsmcc_association_tag, # this association
              tag identifys the carousel, it is used also while
              generating the DSMCC with oc-update.sh and referenced by
              the AIT
172         use = 0,# some default values follow, don't change then,
              different values are not supported
173         selector_lenght = 0,# ...
174         transaction_id = 0x80000000, # ...
175         timeout = 0xFFFFFFFF, # ...
176         private_data = "",
177       ),
178       stream_identifier_descriptor(
179         component_tag = dsmcc_association_tag, # it is the same as
              the assocation tag, some decoders will look for the
              component tag, others for the association tag, the same
              value should be used
```

```
180       ),
181       carousel_identifier_descriptor(
182         carousel_ID = dsmcc_carousel_id, # carousel id number, it's
              a different number from association/component tag, but it
               has a similiar purpouse: identifying the carousel
183         format_ID = 0,# no enhanced boot supported
184         private_data = "",
185       ),
186       data_broadcast_id_descriptor(
187         data_broadcast_ID = 291,# for HbbTv Carousel (http://www.
              dvbservices.com/identifiers/data_broadcast_id?page=3)
188         ID_selector_bytes = "",
189       ),
190     ]
191   )
192  ],
193       version_number = 1,# you need to change the table number every
            time you edit, so the decoder will compare its version
           with the new one and update the table
194       section_number = 0,
195       last_section_number = 0,
196       )
197 #
198 # Service Description Table (ETSI EN 3004685.2.3)
199 # this is a basic SDT with the minimum desciptors, OpenCaster has a
     big library ready to use
200 #
201 sdt = service_description_section(
202  transport_stream_id = demo_transport_stream_id,
203  original_network_id = demo_original_network_id,
204       service_loop = [
205     service_loop_item(
206       service_ID = demo_service_id,
207       EIT_schedule_flag = 0,# 0no current even information is
            broadcasted, 1broadcasted
208       EIT_present_following_flag = 0,# 0no next event information is
             broadcasted, 1is broadcasted
209       running_status = 4,# 4service is running, 1not running, 2
            starts in a few seconds, 3pausing
```

```
210       free_CA_mode = 0,# 0means service is not scrambled, 1means at
              least a stream is scrambled
211       service_descriptor_loop = [
212         service_descriptor(
213           service_type = 1,# digital television service
214           service_provider_name = "mediatvcom",
215           service_name = "Dissertacao",
216         ),
217       ],
218     ),
219       ],
220       version_number = 1,# you need to change the table number every
              time you edit, so the decoder will compare its version
              with the new one and update the table
221       section_number = 0,
222       last_section_number = 0,
223       )
224 #
225 # Application Informaton Table (ETSI TS 10181210.4.6)
226 #
227 #
228
229
230 ait = application_information_section(
231       application_type = 0x0010,
232       common_descriptor_loop = [
233    external_application_authorisation_descriptor(
234     application_identifiers = [[organisationId_1,applicationId_1] ,
          [organisationId_2,applicationId_2]],
235     application_priority = [   5   ,    1   ]
236     # This descriptor informs that 2applications are available on
          the program by specifying the applications identifiers (
          couple of organization_Id and application_Id parameters) and
           their related priorities (5 for the first and 1for the
          second).
237     # Actualy our service contains only one application so this
          descriptor is not relevent and is just here to show you how
          to use this descriptor.
238     # This descriptor is not mandatory and you could remove it (i.e
          . common_descriptor_loop = []).
```

```
239     )
240   ],
241      application_loop = [
242   application_loop_item(
243     organisation_id = organisationId_1, # this is a demo value, dvb
          .org should assign an unique value
244     application_id = applicationId_1,
245
246     application_control_code = 1,
247         # 2is PRESENT, the decoder will add this application to the
              user choice of application
248         # 1is AUTOSTART, the application will start immedtiatly to
              load and to execute
249         # 7is DISABLED, The application shall not be started and
              attempts to start it shall fail.
250         # 4is KILL, it will stop execute the application
251     application_descriptors_loop = [
252       transport_protocol_descriptor(
253         protocol_id = 0x0003, # HTTP transport protocol
254         URL_base = appli_root,
255         URL_extensions = [],
256         transport_protocol_label = 3,# HTTP transport protocol
257       ),
258       application_descriptor(
259         application_profile = 0x0000,
260           #0x0000 basic profile
261           #0x0001 download feature
262           #0x0002 PVR feature
263           #0x0004 RTSP feature
264         version_major = 1,# corresponding to version 1.1.1
265         version_minor = 1,
266         version_micro = 1,
267         service_bound_flag = 1,# 1means the application is expected
              to die on service change, 0will wait after the service
              change to receive all the AITs and check if the same app
              is signalled or not
268         visibility = 3,# 3the applications is visible to the user, 1
              the application is visible only to other applications
269         application_priority = 1,# 1is lowset, it is used when more
              than 1applications is executing
```

```
270         transport_protocol_labels = [3],# If more than one protocol
               is signalled then each protocol is an alternative
               delivery mechanism. The ordering indicates
271                      # the broadcaster's view of which transport
                            connection will provide the best user
                            experience (first is best)
272       ),
273       application_name_descriptor(
274         application_name = appli_name,
275          ISO_639_language_code = "FRA"
276       ),
277       simple_application_location_descriptor(initial_path_bytes =
             appli_path),
278     ]
279   ),
280
281   ],
282       version_number = 1,
283       section_number = 0,
284       last_section_number = 0,
285  )
286
287  ################################################
288
289 ait_dsmcc = application_information_section(
290       application_type = 0x0010,
291       common_descriptor_loop = [],
292       application_loop = [
293     application_loop_item(
294       organisation_id = organisationId_1, # this is a demo value,
             dvb.org should assign an unique value
295       application_id = applicationId_1,
296       application_control_code = 1,
297       application_descriptors_loop = [
298         transport_protocol_descriptor(
299           protocol_id = 0x0001, # the application is broadcasted on
                 a DSMCC
300           transport_protocol_label = 1,# the application is
                broadcasted on a DSMCC
301           remote_connection = 0,# shall be 0for HbbTV
```

```
302             component_tag = 0xB, # carousel common tag and
                    association tag
303         ),
304         application_descriptor(
305             application_profile = 0x0000,
306             version_major = 1,# corresponding to version 1.1.1
307             version_minor = 1,
308             version_micro = 1,
309             service_bound_flag = 1,# 1means the application is
                    expected to die on service change, 0will wait after
                    the service change to receive all the AITs and check
                    if the same app is signalled or not
310             visibility = 3,# 3the applications is visible to the user
                    , 1the application is visible only to other
                    applications
311             application_priority = 1,# 1is lowset, it is used when
                    more than 1applications is executing
312             transport_protocol_labels = [1],# carousel Id
313         ),
314         application_name_descriptor(application_name = appli_name),
315         simple_application_location_descriptor(initial_path_bytes =
                appli_path),
316       ]
317     ),
318   ],
319       version_number = 1,
320       section_number = 0,
321       last_section_number = 0,
322  )
323
324
325
326 #
327 # Stream Event
328 #
329
330 tap = str_event_use_tap()
331 tap.set(
332       id = 0,
```

```
333  assocTag = stream_event_component_tag, # association tag defined
         into PMT for the Stream Event
334  )
335
336 taps = Taps (
337       taps_count = 1,
338  tap_loop = [tap,],
339    )
340
341 event_count = 1# number of events
342
343 event_names = Event_names (
344       eventnames_count = event_count,
345       event_name_loop = ["action"], # name of the events
346    )
347
348 event_ids = Event_ids (
349       eventids_count = event_count,
350       event_id_loop = [1],# id of the events
351    )
```

Código B.2: Tabelas DVB