UNILASALLE
CENTRO UNIVERSITÁRIO LA SALLE
Curso de Bacharelado em Ciência da Computação

HENRIQUE DAMASCENO VIANNA

MODELO DE AMBIENTE ORIENTADO A SERVIÇOS MPEG-7

CANOAS, 2008

HENRIQUE DAMASCENO VIANNA

MODELO DE AMBIENTE ORIENTADO A SERVIÇOS MPEG-7

Trabalho de conclusão apresentado para a banca examinadora do curso de Ciência da Computação do Centro Universitário La Salle, com exigência parcial para a obtenção do grau de Bacharel em Ciência da Computação, sob orientação do Prof. Me. Gaspare Giulliano Elias Bruno.

Canoas, 2008

TERMO DE APROVAÇÃO

HENRIQUE DAMASCENO VIANNA

MODELO DE AMBIENTE ORIENTADO A SERVIÇOS *MPEG-7*

Trabalho de conclusão aprovado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Curso de Ciência da Computação do Centro Universitário La Salle - UNILASALLE, pela seguinte banca examinadora:

Prof. Me. Marcos Ennes Barreto
Unilasalle

Prof. Me. Mozart Lemos Siqueira
Unilasalle

Canoas, 4 de julho de 2008.

DEDICATÓRIA

Dedico este trabalho aos meus pais, à minha avó e minha madrinha, pilares de minha formação. Aos meus irmãos, por serem sempre aquilo o que essa palavra significa. E a Cristiane, pela sua paciência.

AGRADECIMENTOS

Acima de tudo agradeço àqueles entes mais próximos (Seu Ernesto, Dona Neuci, Vó Maria, Dinda, André, Filipi e Igor), por terem sido um porto seguro por toda minha vida, e estarem presentes nas horas em que mais precisei.

Àqueles amigos de verdade: Ao William, que embora tenha trocado de curso esteve junto por toda essa caminhada. Ao Clemilson, meu espelho profissional. Ao Rodrigo Wolf, pela grande parceria que temos. Ao Maurício (primo), companheiro nas piores apresentações do Inter que já presenciei.

Também agradeço a Cristiane, pela sua companhia, grande amizade e por saber me acalmar quando estava desesperado. Ao seu João e a Dona Cristina por terem me aceito tão bem.

A todos os professores por irradiarem e compartilharem os seus conhecimentos. E ao meu orientador, o Professor Gaspare Bruno, pela confiança e assistência.

A Deus por sua presença e paz de espírito nas horas turbulentas.

RESUMO

Com o aumento do uso de conteúdo multimídia pela internet, criou-se a necessidade de padrões que as descrevessem de forma a facilitar sua pesquisa e acesso. MPEG-7, também conhecido como Interface de Descrição de Conteúdo Multimídia, possui capacidade de descrever conteúdo multimídia. Existe uma grande quantidade de estudos relacionados a este padrão, mas a maioria se foca em um problema particular de extração, pesquisa, anotação ou consumo. Este trabalho apresenta um modelo de ambiente orientado a serviços, com objetivo de integrar os diferentes tipos de necessidade inerentes a aplicações MPEG-7. O modelo de ambiente orientado a serviços é composto por três camadas, base de descritores, serviços e diretório de serviços. A base de descritores é responsável por armazenar as descrições no formato MPEG-7. A camada de serviços é encarregada de oferecer meios para pesquisa, criação, manutenção e transformação de descrições MPEG-7. E o diretório de serviços deve registrar e disponibilizar uma forma para que as aplicações possam encontrar os serviços do modelo.Com o auxílio da composição dos serviços do ambiente, é possível criar as mais variadas aplicações multimídia que fazem uso do padrão MPEG-7. Além disso, como o ambiente é disposto através de uma arquitetura orientada a serviços fica fácil a adição e aquisição de serviços no ambiente, como também a criação de novas funcionalidades no modelo.

ABSTRACT

With the growth of multimedia content usage over the internet there was need of a standard that can ease search and access of such content. MPEG-7 also known as Multimedia Content Description Interface was created to describe multimedia meta-data. A great number o MPEG-7 studies has been made but none of this try to create a model which can integrate different kinds o MPEG-7 applications. This work presents such model using service oriented architecture.The service oriented environment model is composed by three layers: descriptors base, services and services directory. The descriptors base has the responsibility of store MPEG-7 descriptions. The service layer offers methods for search, creation, management and transformation of MPEG-7 descriptions. The service directory must register and give a way which applications could find services of the model.With aid of environment service composition it’s possible to create many kinds of multimedia applications that make use of MPEG-7 standard. The environment is disposed in a service oriented architecture that makes easy to create and acquire new functionalities to the model.

LISTA DE ABREVIATURAS E SIGLAS

CI	Computational Intelligence
D	Descriptor
DDL	Description Definition Language
DOM	Document Object Model
DS	Description Scheme
DSType	Description Scheme Type
DType	Descriptor Type
ER	Entidade Relacionamento
FLWOR	FOR, LET, WHERE, ORDER BY and RETURN
FTP	File Transfer Protocol
HTML	HyperText Markup Language
ID3	IDentify an MP3
IETF	Internet Engineering Task Force
IEC	International Electrotechnical Comission
ISO	International Organization for Standartization
MIRROR	MPEG-7 Image Retrieval Refinement based On Relevance Feedback
MP3	MPEG Audio Layer III
MPEG	Moving Picture Expert Group
MSE	Meta Search Engine
ORB	Object Request Broker
PHP	PHP: Hypertext Preprocessor
RPC	Remote Procedure Call
SAPM	Search Agent Processing Model
SCORM	Sharable Content Object Reference Model
SMIL	Synchronized Multimedia Integration Language
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol

SQL	Structured Query Language
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XAML	Extensible Application Markup Language
XLINK	XML Linking Language
XML	Extensible Markup Language
XPATH	XML Path Language
XPOINTER	XML Pointer Language
XQUERY	XML Query Language

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

MPEG-7 ou "*Multimedia Content Description Interface*" é um padrão ISO/IEC desenvolvido pelo *Moving Pictures Expert Group* (MPEG) (Martínez, 2004). O padrão MPEG-7 oferece uma maneira de descrever conteúdos de itens multimídia, por exemplo, uma imagem, uma música (ou uma coleção de músicas), vídeos. Essa descrição pode não somente representar os dados codificados de um item multimídia, como por exemplo, o histograma de cores de uma imagem, mas também relações entre elementos do contexto da imagem (Martínez, 2004). Por exemplo, em uma imagem de crianças jogando bola é possível descrever, de forma semântica, a criança que está com a bola, quem está ao seu lado entre outras peculiaridades da imagem. Outras informações também podem ser descritas como classificação do material descrito, direitos autorais, etc.

O padrão MPEG-7 é dividido em dez partes, sendo quatro delas relacionadas ao formato de descrição do conteúdo multimídia (Martínez, 2004).

MPEG-7 *Description Definition Language*, ou DDL, define a sintaxe de como devem ser utilizados os elementos de descrição MPEG-7 (Martínez, 2005). Então, DDL é uma linguagem de descrição sintática de uma classe de aplicação, no caso aplicações multimídia. Criado a partir da XML *Schema* - cuja intenção é de possibilitar ao usuário a criação de suas próprias classes de documentos XML (com suas restrições e sintaxes) (XML Schema, 2001). A linguagem de descrição DDL possibilita o intercâmbio com outros sistemas como também a extensão de sua gramática pela criação de novos esquemas (Martínez, 2004), (Martínez, 2005).

Baseados na linguagem DDL, tem-se as partes 3 (*Visual*), 4 (*Audio*) e 5 (*Multimedia description schemes*) do padrão MPEG-7. As partes 3 e 4 lidam, em sua maioria, com descritores de baixo nível que podem ser extraídos automaticamente de arquivo multimídia (Martínez, 2005). Os esquemas de descrição (parte 5) formam

um conjunto mais complexo de ferramentas de descrição (Martínez, 2004). Por este conjunto é possível, não somente descrever conteúdos que estão codificados nos itens multimídia, mas também criar esquemas que descrevem comportamentos e características da percepção humana sobre o conteúdo. Sendo assim, pelo uso destes descritores é possível criar a relação entre diferentes elementos de um conteúdo, sumários do conteúdo, ligação entre elementos e recursos externos, informações sobre direito de uso, coleções de objeto, preferências de usuário, e muitas outras (ISO/IEC 15938-5, 2003).

A quantidade de descritores é vasta, como também a quantidade de aplicações que podem utilizar o padrão MPEG-7.

MPEG-7 pode ser utilizado em aplicações de televisão digital, com sumarização, categorização de conteúdo (ISO/IEC 15938-5, 2003). Em ambientes colaborativos pode ser utilizado para descrição de conteúdos audiovisuais.

Em publicações jornalísticas, seria possível coletar diferentes tipos itens multimídia referentes a algum tipo de assunto, para que seja utilizado para compilar uma matéria (Tran-Thuong e Roisin, 2003). Pesquisa por amostra de conteúdo também é um tipo de aplicação que é possível criar com o uso do padrão, por exemplo, encontrar uma música a partir de um trecho cantado (Matushima et al., 2004).

A quantidade de categorias das aplicações é muito maior do que as já citadas, mas alguns comportamentos são compartilhados pelas aplicações.

No artigo "*Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching*" (Matushima et al., 2004) é descrito o desenvolvimento de uma ferramenta de classificação de gêneros musicais. A partir das músicas classificadas pelo sistema, descritores MPEG-7 eram extraídos para que fosse possível executar buscas de músicas através de suas amostras. Neste modelo os autores utilizaram uma base para armazenamento dos descritores e um w*ebservice,* para que fosse possível acessar os descritores por outras aplicações.

Em "*SOLO: AN MPEG-7 OPTIMUM SEARCH TOOL*" (Lay e Guan, 2000) os autores apresentam um protótipo de ferramenta de pesquisa chamada SOLO. SOLO é uma *engine* de pesquisa distribuída baseada no conceito de interfaces MSE (*Metasearch Engine ou Motor de Metabusca*). Neste conceito uma pesquisa é

distribuída entre seus membros. Os resultados obtidos são então qualificados e retornados ao cliente.

Wong, Cheung e Po (Wong, Cheung e Po, 2005) escrevem sobre o projeto MIRROR (*MPEG-7 Image Retrieval Refinement based On Relevance feedback*) que é um sistema de busca baseado na realimentação do conteúdo. Este sistema utiliza técnicas de realimentação por importância (*relevance feedback*) e consultas por amostras para refinar os resultados que serão apresentados ao usuário. O projeto utiliza uma base de descritores MPEG-7 que serve como filtro para as pesquisas de imagem.

Em "*Structuring Interactive TV Documents*" (Goularte, Moreira e Pimentel, 2003) é apresentada uma implementação que cria anotações MPEG-7 em conteúdos multimídia de vídeo. A aplicação é baseada na hierarquia de conteúdo de objetos do padrão MPEG-4, de maneira que é possível construir anotações MPEG-7 conscientes do contexto. Por exemplo, em uma cena onde existem duas crianças em um parque, é possível através da aplicação descrita, acessar o elemento criança e criar anotações orientadas ao contexto do tipo onde, como e o que.

Em geral as aplicações MPEG-7 possuem alguns tipos de operações em comum. Embora não seja uma regra possuir todas essas operações, são elas:

- Armazenar/Modificar a base de descrição multimídia
- Pesquisar informações na base de descrição
- Extrair informações de um item multimídia
- Filtrar informações pertinentes a aplicação dentro de alguma descrição selecionada
- Apresentar o conteúdo encontrado/selecionado de uma forma que facilite o seu consumo pelo usuário

Nas bibliografias estudas estas operações ficam encapsuladas em sua própria arquitetura, e algumas não oferecem uma maneira para que outras aplicações utilizem operações já existentes, obrigando-as a criar suas próprias soluções.

Imaginou-se, então, criar uma arquitetura baseada em *webservices*. Esta arquitetura possui características úteis as aplicações que utilizam o padrão MPEG-7:

- São independentes de plataforma (Curbera et al., 2002)
- Podem ser distribuídos através da internet, podendo ser utilizados por outras aplicações ou *webservices* (Curbera et al., 2002)

- Utilizam padrões abertos e aceitos internacionalmente (entre eles XML e XML *Schema*, utilizados por MPEG-7) (Curbera et al., 2002)
- Podem ser localizados pelas suas características através de serviços de diretório (Curbera et al., 2002)

Sendo assim, uma arquitetura orientada a serviços, que implemente estas operações recorrentes em aplicações MPEG-7, auxiliaria o desenvolvimento rápido de novas aplicações que utilizem este padrão. Sem que haja necessidade de recriar um novo modelo, ou aplicações que já existam.

Esta arquitetura será composta por cinco tipos de serviços: Pesquisadores, Armazenadores, Extratores, Filtros, Apresentadores.

Os Serviços Armazenadores terão a responsabilidade de gerir o repositório de documentos MPEG-7 e devem possuir as seguintes funcionalidades:

- Adicionar novos documentos MPEG-7 ao seu repositório
- Recuperar um documento ou fragmento de um documento MPEG-7
- Inserir fragmentos MPEG-7 em documentos que já estão no repositório.
- Remover fragmentos MPEG-7 de um documento.

Os pesquisadores terão como finalidade pesquisar o repositório de documentos MPEG-7, para retornar um conjunto de esquemas de descrição que estão de acordo com os critérios de consulta fornecidos pelo usuário. Como se trata de um serviço, as aplicações não têm necessidade de conhecer o funcionamento do motor de busca, somente em usar a sua interface de acesso. Uma aplicação pode escolher o serviço de pesquisa que mais se adéqua as suas características, ou também, utilizar vários serviços de pesquisa para aumentar a quantidade de resultados para uma determinada consulta.

Os apresentadores receberão como entrada um documento XML que possua um ou mais fragmentos MPEG-7. Estes fragmentos serão processados e transformados em um formato de conteúdo que facilite o consumo pelo usuário. Neste caso, podem existir serviços de apresentação que recebam a mesma entrada de dados, mas que retornem saídas diferentes, como por exemplo, uma página HTML e uma apresentação SMIL.

Os serviços de filtro receberão um fragmento de documento MPEG-7 e retornarão um ou mais fragmentos que estejam de acordo com o critério de filtro

criado pelo usuário. Este serviço é útil para extrair elementos pertinentes a uma dada aplicação, facilitando as atividades de transformação ou processamento deste conteúdo.

Os extratores são encarregados por criar um documento MPEG-7 a partir dos dados coletados automaticamente de um item multimídia. Assim, descrições que não precisam ser criadas com assistência humana, ou são repetitivas, podem ser extraídas através deste serviço. Por exemplo, através deste serviço seria possível extrair descritores espectrais de um item de áudio. Estes descritores são técnicas utilizadas para reconhecimento de som (Mitrovic, Zeppelzauer e Eidenberger, 2006).

O presente trabalho foi organizado em outras cinco partes: Referencial Teórico, Tecnologias Utilizadas, Modelo Orientado a Serviços MPEG-7, Resultados Obtidos e Conclusão.

O capítulo Referencial teórico explica as arquiteturas implementadas nos trabalhos estudados que utilizam MPEG-7. O capítulo Tecnologias Utilizadas descreve as principais tecnologias utilizadas na implementação do trabalho. Em Modelo Orientado a Serviços MPEG-7 é explicado o funcionamento do modelo criado e também como foi feita a sua implementação. A aplicação criada, como também os resultados de desempenhos feitos em cima dos serviços de extração e pesquisa, são descritos no capítulo Resultados Obtidos. O capítulo conclusão finaliza o trabalho mencionando os benefícios do modelo, aperfeiçoamentos que devem ser feitos e a descrição de trabalhos futuros.

2 REFERENCIAL TEÓRICO

No decorrer da pesquisa foi estudada uma série de trabalhos que envolvem o uso do padrão MPEG-7. Cada trabalho envolve um problema ou aplicação particular no uso de descritores MPEG-7, e utilizam apenas uma parte das ferramentas de descrição especificadas pelo padrão.

Um ponto comum entre as bibliografias estudadas e que foi mencionado no capítulo anterior, é que seus modelos arquiteturais são muito particulares e alguns não possibilitam, ou não descrevem uma forma de integração com outros sistemas. O que é justamente a proposta apresentada neste trabalho de conclusão.

A seguir será feita uma revisão das bibliografias estudadas focando nos seus modelos arquiteturais. Após esta revisão será exposta uma tabela comparativa entre estes modelos e tópicos propostos por este trabalho.

2.1 Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching

Este trabalho descreve a construção de uma solução de biblioteca digital de áudio e vídeo em que o requisito principal era que a catalogação do material digital fosse feita sem a intervenção humana (Matsushima et al., 2004).

Para alcançar seu objetivo os autores criaram os seguintes elementos: classificador de gênero musical, gerenciador de classificação, mecanismo de indexação baseados em MPEG-7, ferramenta de consulta por amostra e *webservice* para compartilhar a informação extraída.

A pesquisa teve foco no uso dos descritores MPEG-7 de áudio que são utilizados pela ferramenta de consulta por amostra, e que são extraídos no momento da classificação da canção.

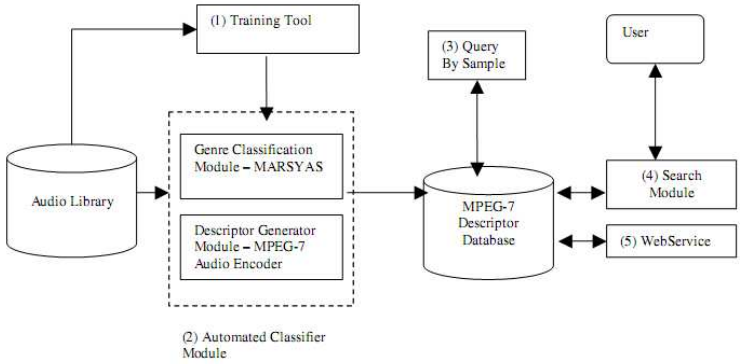

Figura 1 - Modelo da arquitetura do trabalho Integrating MPEG-7 Descriptors and Pattern Recognition
Fonte: Matsushima et al., 2004.

A arquitetura foi dividida de acordo com suas responsabilidades: análise e processamento da canção e auxílio a "busca" de canções. O processo de classificação utilizou os seguintes descritores de áudio MPEG-7: *AudioPowerType*, *AudioSpectrumCentroidType*, *AudioWaveformType*, *AudioSpectrumEnvelopeType*, *AudioSpectrumFlatnessType*, *AudioSpectrumSpreadType*. Estes descritores são gerados pelo módulo gerador de descritor (*Descriptor Generator Module* – MPEG-7 *Audio Enconder*) (Matsushima et al., 2004).

O módulo de treinamento é utilizado para auxiliar sistema em sua classificação de gêneros musicais. A sua função é de treinar a ferramenta de classificação a reconhecer os padrões extraídos de uma determinada canção, categorizando-a em algum gênero musical.

O módulo de classificação é composto por dois componentes: Classificador de Gênero (*Genre Classification Module*) e Gerador de Descritor (*Descriptor Generator Module*). O Classificador de Gênero identifica automaticamente o gênero que mais se aproxima ao da canção processada, utilizando os padrões treinados pelo módulo de treinamento.

O gerador de descritor é um codificador MPEG-7. Este codificador extrai descritores MPEG-7 de baixo nível que vieram através de um arquivo de áudio de entrada e geram uma saída XML no formato MPEG-7 para aqueles descritores extraídos. Os descritores extraídos são inseridos em uma base dados relacional criada. Neste caso os autores optaram pelo uso de um banco de dados relacional por acreditarem que os bancos de dados XML não estão maduros o suficiente. O

modelo ER (Entidade Relacionamento) representa a granularidade em nível de descritores MPEG-7 (Matsushima et al., 2004).

Na ferramenta de busca por amostra, um trecho de áudio é enviado, os descritores MPEG-7 do trecho são extraídos e a consulta é realizada na base de dados de descritores tentando encontrar a canção que mais se aproxime ao padrão extraído.

No módulo de busca o usuário tem a opção de pesquisar por uma determinada canção pelo seu nome e gênero, após selecionar os critérios de pesquisa e executar a consulta, uma lista de todas as canções que estejam de acordo com os critérios é exibida, dando ao usuário a opção de escutá-las.

O *webservice* criado tem a finalidade de compartilhar os descritores extraídos para uso em outras aplicações que necessitem utilizá-los.

2.1 SOLO: AN MPEG-7 OPTIMUM SEARCH TOOL

SOLO é uma *engine* de buscas MPEG-7. De acordo com os autores os estudos que cercam MPEG-7 se concentram mais nos assuntos relacionados a extração de dados do que em buscas (Lay e Guan, 2000).

O modelo é baseado no conceito de motores de metabusca (*meta-search engine*, MSE). O motor de metabusca é uma espécie de intermediário entre outros motores de busca. Ele traduz a consulta de entrada para expressões associadas a outros motores de busca. As consultas são enviadas para cada motor de busca e os resultados são classificados e retornados ao usuário.

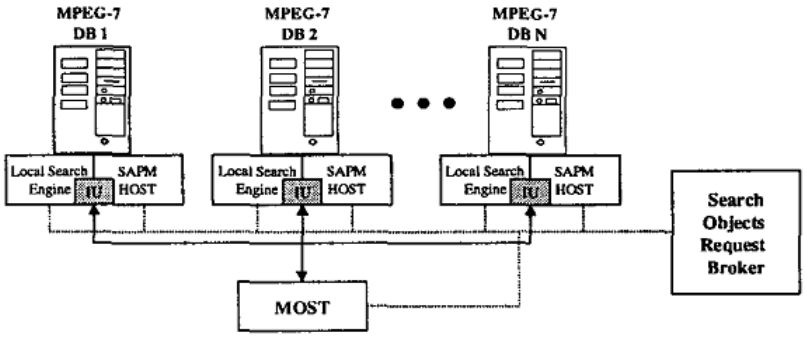

Figura 2 - Modelo da arquitetura do motor de busca SOLO
Fonte: Lay e Guan, 2000.

A figura 2 exibe a arquitetura do motor de busca SOLO. Cada base de dado membro do motor de metabusca possui um motor de pesquisa local e um servidor de agentes de consultas móveis que são utilizados para executar buscas avançadas.

O funcionamento da arquitetura é feito da seguinte maneira: Um usuário entra com uma nova consulta. Esta consulta é enviada ao intermediário do motor de busca (*Search Object Request Broker*) que envia a requisição aos motores de buscas locais (*Local Search Engin*e). Estes retornam o resultado da consulta que é então classificado (Lay e Guan, 2000).

O usuário recebe o resultado da consulta e tem a opção de selecionar aqueles resultados com maior significado a sua busca. Uma vez que o usuário seleciona estes resultados e envia uma nova consulta a *engine*, o intermediário de busca aciona os servidores de agentes móveis (SAPM *Hosts*) que são executados para qualificar as consultas nas bases de dados dos membros do ambiente. A idéia é que estes agentes quebrem as restrições impostas pelos motores de busca locais refinando o resultado de acordo com o critério executado pelo agente (Lay e Guan, 2000).

Ainda, SOLO utiliza mecanismos de inteligência computacional (*Computational Intelligence*, CI) para limitar o espaço de busca apenas naqueles membros relevantes ao domínio das consultas.

2.3 MIRROR (MPEG-7 Image Retrieval Refinement based On Relevance feedback)

MIRROR é um sistema de busca de imagens que utiliza técnicas de refinamento do conteúdo baseado nos itens que o usuário julga como tendo maior relevância. Ou seja, o usuário entra com um conjunto de palavras chaves inicial e envia esta consulta para o responsável pela busca. Uma vez retornado o resultado, o usuário tem a possibilidade de selecionar aquelas imagens que ele julga mais semelhantes com o resultado desejado para refinar a sua consulta.

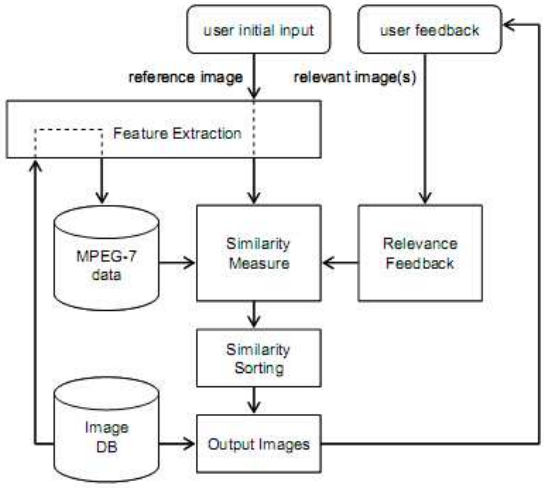

Figura 3 - Modelo de Arquitetura do Sistema de Buscas MIRROR
Fonte: Wong, Cheung e Po, 2005.

O sistema consiste de três módulos principais: Extrator de Características (*Feature Extraction*) que é responsável por extrair os descritores da imagem e converter essa informação no formato MPEG-7. Medidor de Similaridade (*Similarity Measure*) utiliza como parâmetro descritores MPEG-7 extraídos das imagens que o usuário selecionou como referência para selecionar imagens semelhantes. Realimentação por Importância (*Relevance Feedback*) recebe a realimentação dos resultados que o usuário julgou ter maior importância gerando uma nova consulta ao medidor de similaridade (Wong, Cheung e Po, 2005).

2.4 Structuring Interactive TV Documents

Este trabalho apresenta um modelo para estruturar e organizar objetos e programas de televisão interativa. O modelo relaciona a hierarquia de objetos MPEG-4 com os esquemas de descrição baseados em MPEG-7 de forma a facilitar a criação automática de sumários, índices e busca de objetos em aplicações de televisão digital (Goularte, Moreira e Pimentel, 2003).

Na aplicação demonstrada, a estrutura hierárquica dos objetos MPEG-4 é utilizada para que os usuários possam interagir ou tomar conhecimento dos elementos existentes em um vídeo nesse formato (Goularte, Moreira e Pimentel, 2003). O vídeo funciona como uma aplicação. Conforme ele é executado, o usuário pode interagir com elementos pertencentes à cena e tomar conhecimento de informações como: o que é este objeto? Ou, qual é o seu objetivo na cena?

Para tanto, estes elementos foram descritos em um esquema XML próprio chamado *MediaObjectSchema*.

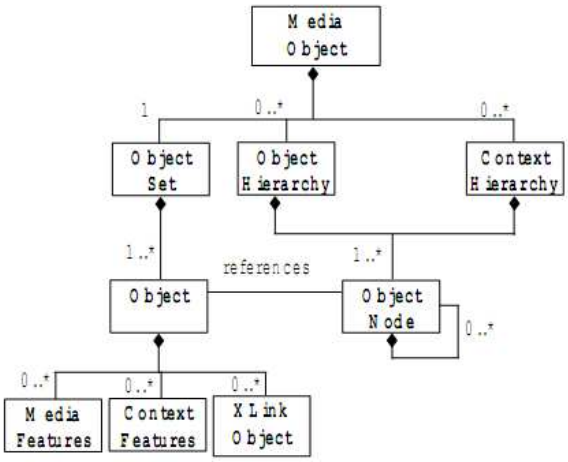

Figura 4 - Esquema criado pelo trabalho Structuring Interactive TV Documents
Fonte: Goularte, Moreira e Pimentel, 2003

Este esquema estende, o tipo MPEG-7 *StructuredAnnotationType*, simplifica o tipo *GraphType* e incorpora o esquemas descrição *MediaFormat*, *Image* e *StillRegion3D*. Esta extensão foi criada para que fosse possível pela aplicação referenciar os objetos existentes no vídeo MPEG-4, como também relacionar objetos da mídia através do padrão XLINK (Goularte, Moreira e Pimentel, 2003).

O elemento *MediaObject* consiste de um elemento *ObjectSet*, zero ou mais elementos *ObjectHierarchy* e zero ou mais *ContextHierarcy*. A idéia é que esses elementos representem uma mídia como um conjunto de objetos conectados através de relações hierárquicas e não hierárquicas.

Para representar um mídia, todos os elementos que fazem parte do conteúdo da mídia são inseridos através do elemento *Object* dentro do elemento *ObjectSet*. Estes elementos *Object* é que referenciam os itens que fazem parte do conteúdo da mídia (por exemplo, um vídeo, uma imagem, uma região).

O elemento *ObjectHierarchy* organiza uma árvore de elementos *ObjectNode*. *ObjectNode* são utilizados para referenciar os elementos *Object*, que constituem a mídia. Desta maneira é possível criar uma hierarquia constituída de um objeto vídeo, que possui em seus elementos filhos dois objetos do tipo região.

O elemento *ContextFeature* é uma extensão do esquema de descrição MPEG-7 *StructuredAnnotationType*. Diferente do *StructuredAnnotationType* que permite apenas a descrição de elementos textuais *Name*, *Definition* e *Term* nos seus

elementos *Who*, *What*, *When*, etc (ISO/IEC 15938-5, 2003). O elemento *ContextFeature* permite uma descrição mais qualificada utilizando informações pessoais, localização, idade, etc.

O elemento *XlinkObject* utiliza o padrão W3C Xlink e permite a criação de ligações do objeto descrito na mídia com outros documentos ou elementos do documento, assim as descrições da mídia e as descrições dos seus elementos podem ser feitas de forma separada, isto é fora do item multimídia (Goularte, Moreira e Pimentel, 2003).

2.5 Análise dos trabalhos estudados

Para comparar os trabalhos estudados foi criada uma tabela que lista algumas funções características do modelo de arquitetura proposto (por exemplo, utiliza funções de anotação, utiliza funções de pesquisa, etc.). Esta tabela analisa a existência das funções desenvolvidas nos serviços do modelo proposto (dispostas na coluna mais a esquerda) dentro das aplicações estudadas que utilizam o padrão MPEG-7. Através destas comparações pode-se validar a real necessidade das funcionalidades que deveriam ser criadas pelos serviços do modelo.

Tabela 1 - Comparativo dos ambientes estudados

	An Environment for Multimedia Indexing and Searching	**SOLO**	**MIRROR**	**Structuring Interactive TV Documents**
Utiliza funções de anotação	NÃO	NÃO	NÃO	SIM
Utiliza funções de pesquisa	SIM	SIM	SIM	NÃO
Utiliza funções de extração de dados	SIM	NÃO	SIM	NÃO
Utiliza funções de apresentação	SIM	NÃO	SIM	SIM
Armazena novos documentos MPEG-7	SIM	NÃO	SIM	SIM
Acessa documentos externos	NÃO	NÃO	NÃO	SIM
Possui interface para acesso a seus serviços	SIM	SIM	NÃO	NÃO

Fonte: Autoria própria, 2008

Com exceção do trabalho *Structuring Interactive TV Documents*, todos os trabalhos utilizam algum tipo de pesquisa de descrições MPEG-7. Uma interface baseada em serviços beneficiaria novas aplicações baseadas em MPEG-7 em duas maneiras: agilizaria o desenvolvimento destas aplicações (já que é comum as aplicações utilizarem algum método de pesquisa), proporcionaria uma rápida troca pela aplicação do serviço de pesquisa por outro serviço mais adequado.

Tanto o trabalho *Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching* quanto o projeto MIRROR utilizam métodos de extração de descritores MPEG-7. O primeiro trabalho poderia utilizar um serviço que proporcionasse a extração dos descritores de áudio de baixo nível juntamente com a catalogação do gênero.

Apenas o projeto SOLO não armazena novos documentos MPEG-7. Todos os outros trabalhos criam documentos MPEG-7 em suas aplicações. Tanto o projeto MIRROR quanto o trabalho *Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching,* inserem documentos MPEG-7 em uma base de dados. No trabalho *Structuring Interactive TV Document* fragmentos MPEG-7 são utilizados, mas não é especificada a forma como eles são armazenados. De certa forma existem algumas contradições nesses quesitos.

Algumas aplicações criam modelos de dados relacionais especialmente para lidar com o acesso aos dados de descritores (Matushima et al., 2004), isso acaba atrapalhando a integração com outras aplicações, já que elas acabam perdendo facilidades de uso de padrões XML, tais como XPATH, XQUERY, XLINK, XPOINTER, etc. Esta situação pode ser contornada com interfaces de serviços padronizadas para acesso e armazenamento dos descritores. Outro problema que pode ocorrer nestes casos é o uso de elementos e descritores não padronizados (Goularte, Moreira e Pimentel, 2003), o que acaba incompatibilizando o uso de uma descrição por aplicações diferentes.

O trabalho *Structuring Interactive TV Document* foi o único estudado que mencionou o acesso a documentos externos. Este trabalho utiliza o padrão XLINK para fazer relações com documentos. Outra maneira de atingir este objetivo é com o uso padrão XPATH, que é também utilizado dentro do padrão MPEG-7 para ligação e relacionamento de descritores (ISO/IEC 15938-5, 2003). Esta aplicação funciona como uma ferramenta de anotação onde é possível criar descrições de alto nível que podem ser utilizadas por serviços de televisão digital ou busca para pesquisa através

de objetos em cena ou contextos (por exemplo, um gol numa partida de futebol) (Goularte, Moreira e Pimentel, 2003).

A interface para acesso a serviços é uma maneira interessante de propiciar à outras aplicações acessos a funcionalidade do sistema. O artigo *Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching* descreve a construção de um serviço para acesso aos seus descritores de áudio, mas não descreve a sua interface, ou como isso é feito. O projeto SOLO descreve uma interface de serviço no padrão ORB (O*bject Request Broker*), mas o mesmo poderia utilizar os padrões de *webservice* sendo igualmente independente de plataforma e reutilizável por outras aplicações.

De acordo com os estudos mencionados e a análise de funcionalidade criada (tabela 1), pode-se averiguar as operações que são comuns nas aplicações estudadas que utilizam o padrão MPEG-7 e que são implementadas no modelo proposto por este trabalho. Estas operações foram categorizadas em serviços e são melhores descritas no capítulo Modelo Orientado a Serviços MPEG-7.

3 TECNOLOGIAS UTILIZADAS

O desenvolvimento do ambiente de serviços MPEG-7, bem como das aplicações que utilizaram este ambiente, foi marcado pelo uso das seguintes tecnologias: SOA/*Webservices*, MPEG-7 DDL e MDS, XPATH e XQUERY. Embora outras tecnologias tenham sido utilizadas, estas foram as mais marcantes e serão comentadas a seguir.

3.1 SOA

A noção básica de *SOA* diz respeito a uma arquitetura de entidades de software distribuídas, as quais implementam um conjunto de funções relacionadas ao seu negócio – *coarse grained* – que são acessadas por outras aplicações através de uma interface previamente publicada em algum tipo de catálogo, para que as mesmas sejam acessadas por outras entidades ou aplicações (Elfatatry, 2007), (Ort, 2005), (Endrei et al., 2004).

Em outras palavras pode-se dizer que *SOA* é uma arquitetura formada por serviços ("*Services*") que possuem uma interface ("*Service description*") publicada em algum repositório de serviços ("*Service Registry*"). Estes serviços ficam disponibilizados em algum provedor ("*Service Provider*"). Uma aplicação cliente ("*Service Consumer*") procura por estes serviços através do repositório de serviços, após sua localização a aplicação cliente conecta-se ao provedor para utilizar as funcionalidades do serviço (Papazoglou e Georgakopoulos, 2003), (Endrei et al., 2004).

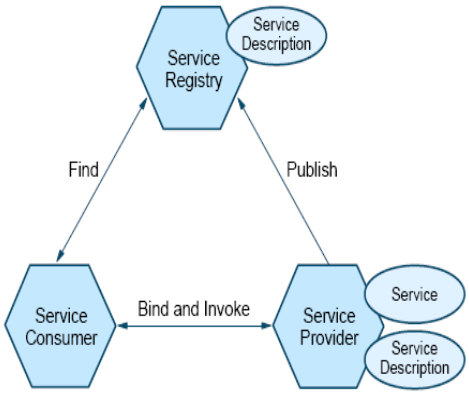

Figura 5 - Funcionamento de uma arquitetura orientada a serviços
Fonte: Endrei et al., 2004

Como vemos na figura 5, a arquitetura *SOA* possui alguns elementos principais (embora estes não sejam os únicos).

Service: Componentes de software auto-descritivos (Papazoglou e Georgakopoulos, 2003) que interagem com outras aplicações ou serviços (Elfatatry, 2007) acessados através de uma interface pública e localizável por métodos de comunicação padronizados baseados em troca de mensagem. (Endrei et al., 2004)

Service Description: São as informações sobre o serviço como interface, comportamento, qualidade, descrevendo a maneira que o consumidor deve interagir com o provedor (Papazoglou e Georgakopoulos, 2003), (Endrei et al., 2004). A partir destas informações um serviço consumidor pode localizar serviços de seu interesse.

Service Registry: Camada que armazena informações sobre um serviço (Endrei et al., 2004). Os provedores de serviço utilizam esta camada para publicar informações sobre seus serviços enquanto os consumidores de serviço utilizam esta camada para descobrir informações e disponibilidade de serviços (Yang, 2003), (Ort, 2005), (Endrei et al., 2004).

Service Consumer: Também conhecido como cliente, é aplicação que requisita um serviço através do registro. A requisição é vinculada a um meio de transporte por onde o cliente invoca a função do serviço (Endrei et al., 2004), por exemplo HTTP, FTP, SMTP*, etc.*

Service Provider: Semelhante ao conceito de servidor em uma arquitetura cliente-servidor. É a entidade que executará as requisições do cliente. Além disso, é responsabilidade do provedor de serviço publicar as informações de seus serviços no registro de serviço (Endrei et al., 2004).

3.1.1 Webservices

Webservice é uma implementação de arquitetura orientada a serviço (Ort, 2005), (Endrei et al., 2004) nas quais aplicações auto-contidas (serviços) utilizam a internet como meio para executar suas funcionalidades (Yang, 2003). Padrões baseados em *eXtended Markup Language* (XML) provêem os mecanismos de comunicação (SOAP), descrição de serviços (WSDL) e descoberta de serviços (UDDI) definidos na arquitetura SOA (Papazoglou e Georgakopoulos, 2003), (Yang, 2003), (Curbera et al., 2002).

Os *webservices* possuem os mesmos elementos de SOA: *Service, Service Description, Service Registry, Service Consumer e Service Provider.* A diferença é que os w*ebservice*s implementam estes elementos através de padrões documentados e aceitos internacionalmente.

S*ervices descriptions* são feitos utilizando a linguagem de descrição de *webservices – Web Services Description Language* (WSDL).

Service registry, em geral, é implementado utilizando as especificações *Universal Description, Discovery, and Integration* (UDDI) que fornecem mecanismos para publicação e de localização de serviços (Curbera et al., 2003).

Toda comunicação entre os elementos de SOA – isto inclui publicação e pesquisa de serviços, como também execução de procedimentos remotos - são feitas utilizando o protocolo simples de acesso a objetos, ou *Simple Object Access Protocol* (SOAP). As aplicações *webservices* são caracterizadas pelo trio SOAP*,* WSDL*,* UDDI.

SOAP ou, *Simple Access Object Protocol* é um protocolo baseado em XML (Ort, 2005), (Curbera et al., 2002) para troca de mensagem e execução de procedimentos remotos (RPC) . O protocolo define uma mensagem que é dividida em duas partes: corpo e cabeçalho. (Curbera et al., 2002)

WSDL ou, *Web Service Description Language* define as formas de acesso ao serviço dividindo a descrição em duas partes. A primeira define a descrição do serviço via nível de aplicação: Descreve as mensagens que são aceitas pelo serviço e a forma como é executada a interação destas mensagens entre cliente e servidor. (Ort, 2005), (Curbera et al., 2002)

A segunda parte define questões de acesso ao serviço como protocolo a ser utilizado, como executar as interações dentro do protocolo e qual é o destino da conexão. (Curbera et al., 2002)

UDDI*,* ou *Universal Description, Discovery, and Integration* (UDDI) é uma espécie de serviço de diretório, semelhante a uma lista telefônica. (Ort, 2005), (Curbera et al., 2002)

UDDI implementa uma interface SOAP na qual é possível consultar e publicar informações do tipo “Quem”, “O que”, “Onde”, “Como” (UDDI Version 2.04, 2002). “Quem” são as informações relacionadas ao mantenedor de um determinado serviço. “O que” descreve o que faz o serviço. “Onde” e “Como” são informações técnicas. “Onde” descreve o local de acesso do serviço, este pode ser uma URL (Endereço *web*), endereço de *e-mail*, ou até mesmo um número de telefone. “Como” descreve as informações técnicas de uso de determinado serviço. (Curbera et al., 2002), (UDDI Version 2.04, 2002)

O formato padrão para intercâmbio de dados utilizados pelos *webservices* é o XML, que também é a linguagem utilizada para descrever conteúdo multimídia através de MPEG-7. Na próxima seção será explicado o que é MPEG-7, e de que forma é feita a descrição do conteúdo multimídia.

A descrição de SOA e w*ebservices* foram resumidas neste trabalho. Seria estender demasiadamente a redação da monografia, descrever os aspectos mais técnicos que englobam a arquitetura dos componentes de *webservices*. As ferramentas de desenvolvimento mais atuais ocultam grande parte destes aspectos e deixam visíveis aqueles aspectos com os quais o desenvolvedor é mais familiar.

Outra peculiaridade é que este trabalho não faz uso de registros UDDI. Esta decisão foi tomada pelos seguintes motivos:

O “contrato” com os serviços é feito dentro das aplicações. São apenas três aplicações que utilizam estes serviços, a aplicação de pesquisa e apresentação de sumários, o teste de desempenho do serviço de extração e o teste de desempenho do serviço de pesquisa. Caso seja necessário o uso de um registro UDDI, basta corrigir estas três aplicações.

Sendo uma aplicação conceitual, todos os serviços estão armazenados no mesmo servidor. No momento em que estes serviços forem dispostos em localizações separadas e estas localizações estejam propensas a troca, será interessante o uso de um registro UDDI.

3.2 MPEG-7

MPEG-7 ou “*Multimedia Content Description Interface*” é um padrão ISO/IEC desenvolvido pelo *Moving Pictures Expert Group* (MPEG). O padrão MPEG-7 possui capacidade de descrever conteúdo multimídia (imagem, som, etc.) através de ferramentas de descrição (*description tools*). (Chang, Sikora e Atu, 2001), (Martínez, 2005), (Martínez, Koenen e Pereira, 2002)

É possível, utilizando o padrão MPEG-7*,* descrever conteúdos multimídia de alto e baixo nível. Descrições de alto nível, ou *Descriptors Schemes* representam as informações contextuais de um conteúdo multimídia e são formadas por um conjunto de descritores (*Descriptors*). As descrições de baixo nível são informações implícitas no conteúdo multimídia tais como cores, texturas, formatos, timbres, etc.

Por exemplo, em uma imagem de “um cachorro correndo numa praia”, os descritores de baixo nível seriam os aspectos de cores, luz, etc, aplicados ao formato binário da imagem. Informações do tipo "Quem", "o que", "onde" são relacionadas a descrições de alto nível e que geralmente não podem ser extraídas automaticamente. (Martínez, Koenen e Pereira, 2002)

O padrão *MPEG-7* é consutituído de dez partes: *Systems*, *Description Definition Language*, *Visual*, *Audio*, *Multimedia Description Schemes*, *Reference Software*, *Conformance Testing*, *Extracion and use of descriptions*, *Profiles and levels* e *Schema Definition* (Martínez, 2004). Este trabalho abordará apenas as partes *Description Definition Language* e algumas partes do *Multimedia Description Schemes,* já que são estas duas partes que envolvem as aplicações criadas no presente trabalho.

3.2.1 Description Definition Language (DDL)

Description Definition Language ou DDL é a linguagem utilizada pelo padrão MPEG-7 para descrever o relacionamento sintático e semântico dos descritores (D's) de conteúdo multimídia especificados pelo padrão. Além disso, a linguagem possibilita a criação de novos descritores e esquemas de descritores (DS's), bem como a sua extensão e modificação (Hunter, 2001), (Chang, Sikora e Atu, 2001), (Martínez, 2005), (Hunter, 2000).

Para que haja interoperabilidade entre os padrões XML, definidos pelo *World Wide Consortium (W3C)*, foi decidida a adoção da linguagem *XML Schema* (Hunter, 2000), (Hunter, 2001). Contudo, para satisfazer os requisitos propostos para MPEG-7 DDL, foi necessária a definição de extensões a *XML Schema Language*, já que a mesma não fora projetada para descrever conteúdo multimídia (Hunter, 2000), (Hunter, 2001).

A linguagem DDL é dividida em três partes, sendo as duas primeiras partes da especificação da XML Schema Language (Hunter, 2001), (Martínez, 2005), (Hunter, 2000), (XML Schema, 2001):

- Componentes estruturais de linguagem *XML Schema* (*XML Schema Part 1*)
- Componentes de tipos de dados da linguagem *XML Schema* (*XML Schema Part 2*)
- Extensões específicas MPEG-7

Os componentes estruturais de linguagem *XML Schema* possibilitam a descrição de restrições do conteúdo (Hunter, 2001), (Martínez, 2005), (Hunter, 2000), (XML Schema, 2001) de documentos XML associados a um determinado "*schema*". Estas restrições podem ser aplicadas a relações entre elementos, componentes que fazem partes do elemento, tipos de dados. Em suma, *XML Schema Part 1* provê uma maneira de aplicar restrições sintáticas, estruturais e de valores em documentos XML (XML Schema, 2001).

Os componentes de tipos de dados da linguagem *XML Schema Part 2* fornecem especificações de maneira que seja possível restringir os tipos de dados em documentos XML (XML Schema, 2001). A especificação provê um conjunto de tipo de dados primitivos (inteiros, *strings*), um conjunto de dados derivados (como inteiros longos, *strings* normalizadas, isto é conjunto de *strings* sem retorno de linha). Na especificação também são definidas formas de como o usuário pode construir seus próprios tipo de dados derivados (Hunter, 2001), (XML Schema, 2001), (Martínez, 2005).

A extensões para MPEG-7 DDL foram criadas pelo fato da *XML Schema* não implementar algumas características que eram necessárias para satisfazer os requerimentos de MPEG-7 DDL (Hunter, 2001), (Martínez, 2005). Para satisfazer estas características foram criados tipos de dados para vetores e matrizes e mecanismos para restringir suas dimensões. Também foi necessário criar

mecanismos para restringir o tipo de um elemento referenciado, de seus elemtentos e de alguns tipos de dados derivados: *mimeType*, *country Code*, *region Code* , *currency Code*, *characterSetCode*, *basicTimePoint* e *basicDuration* (Hunter, 2001), (Martínez, 2005).

DDL constitui o meio pelo qual se descreve o conteúdo multimídia. É a especificação desta linguagem que favorece o intercâmbio padronizado de informações entre diferentes "serviços multimídia". As sintaxes da linguagem DDL, como as sintaxes de outras partes do padrão MPEG-7 (*Audio*, *Visual* e MDS) são descritas através de *XML Schema*. Estes "*schemas*" podem ser "baixados" através do *site http://m7itb.nist.gov/M7Validation.html*.

3.2.2 MPEG-7 *Multimedia Description Schemes*

Os MPEG-7 *Multimedia Description Schemes* (MDS), também conhecidos por "*Multimedia Content Description Interface*" (ou Interface de Descrição de Conteúdo Multimídia), provêem, de acordo com os autores, "Um conjunto de tecnologias padronizado para descrever conteúdo multimídia" (ISO/IEC 15938-5, 2003).

O objetivo deste padrão é especificar as ferramentas genéricas de descrição multimídia (ISO/IEC 15938-5, 2003). Suas funcionalidades estão definidas de acordo com a figura 6.

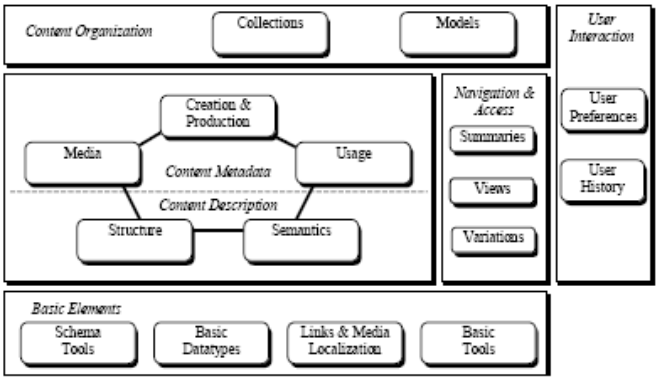

Figura 6 - Visão geral das ferramentas de descrição do *Multimedia Description Scheme*
Fonte: ISO/IEC 15938-5, 2003

Os elementos do fundo da figura 6, ou "*Basic Elements*", são os "blocos de construção" para as ferramentas de descrição de mais alto nível (ISO/IEC 15938-5,

2003), e são definidos como *Schema tools*, *Basic Datatypes*, *linking and media localization* e *basic description tools*.

No meio da figura 6 são exibidas as ferramentas de descrição de conteúdo que lidam com as características do conteúdo multimídia e os “metadados imutáveis” relacionados a este conteúdo (ISO/IEC 15938-5, 2003). Neste contexto são definidas as ferramentas de descrição de estrutura (como segmentos espaciais e temporais), descrição da mídia (formato de armazenamento e codificação), criação e produção (título, criador, classificação), descrição semântica (o que acontece no conteúdo) e uso (direitos de acesso).

No lado direito das ferramentas de descrição de conteúdo estão àquelas ferramentas utilizadas para navegação, sumarização e acesso do conteúdo (ISO/IEC 15938-5, 2003).

Localizado na figura 6, acima das ferramentas de navegação e acesso, encontram-se as ferramentas de organização de conteúdo. Nesta ferramenta são descritos modelos e estruturas de organização do conteúdo como coleções, modelos de probabilidade, modelos de classificação.

Bem a direita, na figura 6, encontra-se as ferramentas relacionadas a interação do usuário, que descrevem preferências do usuário com relação ao conteúdo multimídia e também a interação deste com o conteúdo multimídia.

Cada categoria da especificação MDS possui uma série de ferramentas e esquemas de descrição, somente a especificação MDS contém 738 páginas. Tornar-se-ia completamente inviável descrever a especificação completa neste trabalho. Sendo assim, serão descritos aqueles elementos que foram utilizados na implementação deste trabalho.

3.2.3 Tipos básicos MPEG-7

O diagrama da figura 7 representa uma visão geral de como é a organização de uma descrição no padrão MPEG-7. *Mpeg7BaseType* é tipo básico da hierarquia, desta “classe” estendem os tipos *HeaderType*, *DSType* e *DType.*

DSType provê a estrutura base para os outros *descripitons schemes* (DS), enquanto *DType* provê a estrutura base para outros *descriptors* (D).

HeaderType é uma estrutura base para cabeçalhos e representa a informação dos cabeçalhos dos DS. A idéia é que ele colete informações que possam ser

utilizadas por ferramentas de referência através do atributo *id* que este elemento possui (ISO/IEC 15938-5, 2003).

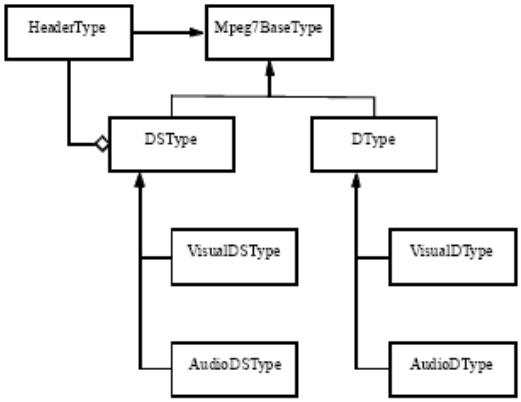

Figura 7 - Organização dos tipos básicos de descrição MPEG-7
Fonte: ISO/IEC 15938-5, 2003

O elemento raiz (*Mpeg7*) é utilizado como o maior elemento da descrição. Este elemento possibilita uma escolha entre elementos *Description* ou uma unidade de descrição (*Description Unit*) (ISO/IEC 15938-5, 2003).

Description: Descreve o conteúdo multimídia utilizando os tipos *CompleteDescriptionType* ou tipos derivados, por exemplo, *SummaryDescriptionType.*

Description Unit: Permite o uso de qualquer tipo de elemento *Mpeg7* (*Descriptor*, *DescriptionScheme*, *Header*), já que ele é um elemento do tipo *Mpeg7BaseType*. Assim, “pode ser utilizado para representar informação parcial de uma descrição completa” (ISO/IEC 15938-5, 2003). Obrigatoriamente, uma descrição MPEG-7 conterá um elemento do tipo *Mpeg7* e um dos dois elementos, *Description* ou *DescriptionUnit.*

Os tipos completos de descrição ou, *Complete Description Types*, são utilizados para “descrever o conteúdo multimídia, abstrações do conteúdo multimídia e metadados relacionados ao gerenciamento do conteúdo multimídia”, e também descrever relações entre outros esquemas de descrição (ISO/IEC 15938-5, 2003).

Complete Description Type é um tipo abstrato, sendo assim, quando o esquema requer o uso deste tipo de descrição é necessário explicitamente informar qual o tipo concreto que irá se utilizar. A figura 8 exibe a hierarquia dos tipos de descrição.

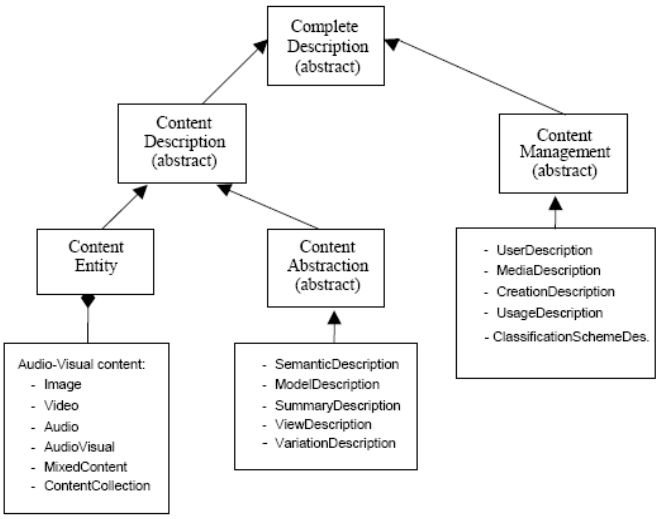

Figura 8 - Visão geral do *Complete Description Type*
Fonte: ISO/IEC 15938-5, 2003.

ContentDescriptionType: "Superclasse" dos tipos *Content Entity* e *ContentAbstraction*

Content Entity: Utilizado para descrever conteúdos de imagens, vídeo, áudio, coleções, etc.

Content Abstraction: "Descreve abstrações do conteúdo multimídia" (ISO/IEC 15938-5, 2003). Estas abstrações representam formas diferentes de organizar, apresentar, modelar este conteúdo multimídia. Estas abstrações são divididas nos seguintes tipos concretos:

SemanticDescriptionType: Descrição semântica do conteúdo, ou seja conceitos do mundo real, como eventos, objetos envolvidos, conceitos, etc.

ModelDescriptionType: Descrição dos modelos do conteúdo multimídia, e consiste na descrição do conteúdo multimídia de acordo com uma estrutura de organização, por exemplo coleções, classificações, etc.

SummaryDescriptionType: Descrição de sumarização de conteúdos multimídia. Define formas de apresentar o conteúdo multimídia de uma maneira compacta, e com facilidades de acesso.

ViewDescriptionType: Descrição de visualização e decomposições de sinais de áudio e vídeo. Esta descrição pode representar a partição de uma região em uma imagem, opções de visualização de acordo com a resolução de um sinal de áudio ou vídeo e também a organização destas decomposições e visualizações.

VariationDescriptionType: Descreve variações do conteúdo multimídia descrito, como por exemplo, variações de linguagem, fidelidade, codificações.

ContentManagementType é um tipo abstrato utilizado para descrever os metadados relacionados ao gerenciamento de conteúdo. Este tipo é especializado pelos seguintes tipos concretos:

- *UserDescriptionType*: Descrição de um usuário em um sistema multimídia
- *MediaDescriptionType*: Informações sobre o conteúdo multimídia, como formato físico, perfil do conteúdo e identificação.
- *CreationDescriptionType*: Descreve o processo de criação do conteúdo multimídia, fazem parte desta descrição informações de título, autor, público, forma de criação.
- *UsageDescriptionType*: Descrição da forma de uso do conteúdo multimídia, por exemplos, direitos de uso, edição, emissão e também, informações sobre custo de produção.
- *ClassificationSchemeDescriptionType*: É o esquema de classificação do conteúdo multimídia, propriamente dito. Por exemplo, gênero de um conteúdo (esporte, televisão, etc.) ou língua (português, inglês, etc.).

O trabalho desenvolvido utilizou *HierarchicalSummary DS* para descrição dos sumários hierárquicos, como também *ContentEntityType* e *Creation DS* para descrição dos dados de criação de itens multimídia e tempo - extraídos dos arquivos de áudio MP3 pela aplicação extratora. A seguir será descrito o que foi utilizado destes descritores para implementação do trabalho, como também uma revisão destes dois esquemas de descrição (*Creation DS* e *HierarchicalSummary DS*).

Antes serão comentadas algumas ferramentas genéricas utilizadas pelas aplicações, são elas: *MediaTimeType*, *MediaLocatorType* e *TextAnnotationType*.

3.2.4 MediaTimeType

MediaTimeType é uma ferramenta de descrição de tempo que pertence as ferramentas de ligação identificação e localização do padrão MPEG-7 (*Linking, identification e localization tools*). As ferramentas de descrição de tempo são baseadas no conjunto de expressão léxicas do padrão *ISO 8601*, que podem representar tanto tempo no formato que nós conhecemos (horas, minutos e segundos) como o tempo utilizado em dados de áudio e vídeo (ISO/IEC 15938-5, 2003).

O *MediaTimeType* especifica a codificação do tempo descrito em uma mídia e é representado por dois elementos de escolha que representam o ponto inicial e a duração da mídia. Diz-se dois elementos de escolha, pois a definição do esquema MDS especifica que as “instâncias” do tipo *MediaTime* devem escolher entre os elementos *MediaTimePoint, MediaRelTimePoint e MediaRelIncrTimePoint* para descrição do ponto inicial da mídia. E *MediaDuration* ou *MediaIncrDuration* para descrição da duração.

```
<MediaTime>
      <MediaTimePoint>T00:00:00</MediaTimePoint>
      <MediaDuration>PT2M43S</MediaDuration>
</MediaTime>
```

Quadro 1 – Exemplo de uso do *MediaTimeType*
Fonte: Autoria própria, 2008.

No exemplo do quadro 1, temos uma descrição de tempo de mídia que tem como ínicio o tempo de zero hora, zero minuto e zero segundos. O conteúdo do elemento *MediaTimePoint* é definido através da seguinte expressão *YYYY-MM-DDThh:mm:ss:FNNN*, onde *YYYY* representa o ano, *MM* representa o mês, *DD* representa o dia, *T* informa o início da descrição de tempo, *hh* representa hora, mm representa os minutos, *ss* representa os segundos, *F* representa o início da descrição da fração de tempo e *NNN* representa o número de frações de um segundo.

O conteúdo do elemento *MediaDuration* indica que o conteúdo de mídia possui dois minutos e quarenta e três segundos. Este conteúdo é definido pela expressão regular *(-)PnDTnHnMnSnNnF* e representa a duração de um período *P* da seguinte maneira *nD*(*n* dias), *T* (tempo) *nH* (*n* horas), ..., *nF* (*n* frações).

3.2.5 MediaLocatorType

O *MediaLocator* também é um componente pertencente as ferramentas de ligação identificação e localização do padrão MPEG-7 (*Linking, identification e localization tools*). Através desta ferramenta é possível referenciar mídias de duas formas: Pela referência a um identificador de recurso uniforme (*Uniform Resource Identifier,* URI), ou inserindo o conteúdo da mídia propriamente dito dentro desta descrição.

A primeira forma é descrita através do elemento *MediaUri.* URI é um padrão descrito pela *IETF (Internet Engineering Task Force)* e representa uma maneira de "identificar um recurso abstrato ou físico através de uma seqüência de caracteres."

```
<SourceLocator>
      <MediaUri>
      http://localhost/audios/pesquisa02.mp3
      </MediaUri>
</SourceLocator>
```

(URI, 1998). A segunda forma é descrita através do elemento *InLineMedia.*

Quadro 2 – Exemplo de uso do MediaLocatorType
Fonte: Autoria própria, 2008.

O exemplo do quadro 2 demonstra um exemplo do uso do *MediaLocatorType*, onde o identificador de recurso aponta para um arquivo que pode ser acessado via protocolo de transferência de hipertexto (HTTP). Neste caso a localização é descrita pelo elemento *SourceLocator* (localizador de fonte). Este elemento é do tipo *TemporalSegmentLocatorType* que é uma especialização de *MediaLocatorType.*

Através desta especialização é possível descrever o segmento que será acessado. Este segmento pode ser descrito através tipo *MediaTime* ou através do elemento *BytePosition* que requer dois atributos: *offset,* que é a posição do primeiro *byte* a ser acessado e *lenght*, que é o total de *bytes* acessados.

3.2.6 TextAnnotationType

A ferramenta *TextAnnotation* faz parte das ferramentas básicas de descrição do padrão MPEG-7 e são utilizadas na construção de outras ferramentas de descrição (ISO/IEC 15938-5, 2003).

A ferramenta *TextAnnotation* pode ser utilizada para criar anotações textuais estruturadas em várias línguas (que respondem aos seguintes atributos: Quem? O que? Onde? Como? Por quê?), informações não estruturadas (através da descrição de um texto livre), expressões que representam uma série de palavras chaves e até mesmo relações sintáticas entre sujeito e verbo.

O tipo TextAnnotation é composto de:

- Um ou mais elementos do tipo: FreeTextAnnotation (Anotação de texto livre), StructuredAnnotation (Anotação que responde os seguintes atributos: Quem? O que? Onde? Como? Por quê?), DependencyStructure (estrutura baseada em depedências sintáticas) ou KeyWordAnnotation (palavras-chave)
- da presença opcional destes atributos: *relevance* (escala de zero a um da relevância da anotação), *confidence* (semelhante a relevância indica a confidencialidade numa escala de zero a um) e,
- da presença obrigatório do atributo *xml:lang*, que indica a linguagem da anotação.

As aplicações criadas neste trabalho utilizaram os elementos *FreeTextAnnotation* e *StructuredAnnotation*, conforme o exemplo do quadro 3.

```
<FreeTextAnnotation xml:lang="pt">
      Segundo Voice Cast realizado pelo
professor Abraham       Lincoln Rabelo no
período letivo de 2007/2, com
      objetivo de elucidar os elementos de um
projeto
      de pesquisa.
</FreeTextAnnotation>
<StructuredAnnotation>
      <Who>
            <Name xml:lang="pt">
                  Professor Abraham Lincoln
Rabelo
            </Name>
      </Who>
      <WhatAction>
                        <Name xml:lang="pt">
                  Voice Cast
            </Name>
                  </WhatAction>
```

Quadro 3 – Exemplo de uso dos elementos FreeTextAnnotation e StructuredAnnotation
Fonte: Autoria própria, 2008.

3.2.7 Creation DS

O esquema de descrição de criação (*Creation* DS) especifica a forma que um conteúdo multimídia foi concebido. Através deste esquema são descritas as informações de local de criação, datas, instrumentos de criação, equipes e organizações envolvidas. Este DS pertence às ferramentas de descrição de criação e produção onde também se localizam os seguintes esquemas de descrição: *CreationInformation DS*, *Classification DS* e *RelatedMaterial DS*.

Embora o *Creation* DS faça parte do *CreationInformation* DS, este último não será detalhado já que nenhum elemento seu é referido nas aplicações construídas, sendo utilizado apenas por regra sintática.

O *Creation DS* é constituído pelos seguintes elementos:

- *Title*: É título do conteúdo descrito textualmente. Este título pode conter um dos seguintes valores no atributo *type*: *main*, *secondary*, *alternative*, *original*, *popular*, *opusNumber*, *songTitle*, *albumTitle*, *seriesTitle*, *episodeTitle*
- *TitleMedia*: Descreve o conteúdo descrito em um formato multimídia. Por exemplo, um áudio, uma imagem, ou um vídeo.
- *Abstract*: Resumo textual do conteúdo multimídia. Este elemento é do tipo *TextAnnotation*, descrito anteriormente.
- *Creator*: Pessoa, organização, grupo, etc, que criou o conteúdo multimídia.
- *CreationCoordinates*: Define a localização e data da criação do conteúdo multimídia.
- *Location*: Local onde o conteúdo foi criado
- *Date*: Data ou período que o conteúdo foi criado
- *CreationTool*: Descreve uma ferramenta que foi utilizada para criar o conteúdo multimídia
- *CopyrightString*: Representa a informação que deve ser apresentada ao usuário, esta informação “não representa uma declaração formal de direitos de uso” (ISO/IEC 15938-5, 2003).

A aplicação de extração criada utiliza os elementos *Title*, *Abstract* e *Creator*, conforme o exemplo do quadro 4. Este exemplo foi criado pela ferramenta de

extração de arquivos MP3 e descreve as informações a respeito da criação de uma canção. O elemento *Abstract* foi utilizado para descrever as informações da banda que gravou a canção e o ano de gravação. O elemento *Creator* descreve o agente criador do conteúdo e qual o seu papel na criação deste conteúdo.

```
<Creation>
      <Title type="songTitle">Listen</Title>
      <Abstract>
            <FreeTextAnnotation>
                  Holly Golightly – Listen
            </FreeTextAnnotation>
            <StructuredAnnotation>
                  <Who>
                        <Name>Holly Golightly</Name>
                  </Who>
                  <When>
                        <Name>2003</Name>
                  </When>
            </StructuredAnnotation>
      </Abstract>
      <Creator>
            <Role href="creatorCS">
                  <Name>Publisher</Name>
            </Role>
            <Agent xsi:type="PersonType">
                  <Name>
                        <GivenName>Damaged Goods</GivenName>
                  </Name>
            </Agent>
      </Creator>
</Creation>
```

Quadro 4 – Exemplo de descrição utilizando o *Creation* DS
Fonte: Autoria própria, 2008.

3.2.8 HierachicalSummary DS

O esquema de descrição de sumário hierárquico faz parte das ferramentas de navegação e acesso do padrão MPEG-7. Tais ferramentas, conforme descritas anteriormente dividem-se em: ferramentas de sumarização, partições, visualizações, decomposições e variações (ISO/IEC 15938-5, 2003).

Neste trabalho foram utilizadas ferramentas de sumarização, mais especificamente sumários hierárquicos. De acordo com a especificação ISO/IEC 15938-5, os “sumários facilitam a descoberta, navegação, visualização e sonorização de conteúdo multimídia” (ISO/IEC 15938-5, 2003). A idéia básica da sumarização é que esta seja uma maneira rápida e prática para capitular o conteúdo de áudio ou vídeo. O sumário é composto por três elementos chave:

- Conteúdo fonte, ou seja, o conteúdo que será sumarizado

- Conteúdo do sumário. São os componentes utilizados para criar o sumário, por exemplo, clipes de áudio, vídeo.
- Sumário, é a representação do sumário através da organização dos seus componentes e que contém a informação essencial do conteúdo fonte.

O padrão MPEG-7 especifica dois tipos de sumários: *Hierarchical Summary* e *Sequential Summary*. O primeiro descreve sumários hierárquicos de tempo variáveis compostos por áudios ou vídeos. O segundo representa sumários seqüenciais de imagens e vídeos, que podem ser sincronizados e exibidos de diferentes maneiras, por exemplo, um *slide show*, apresentação instantânea ou em modo rápido (ISO/IEC 15938-5, 2003).

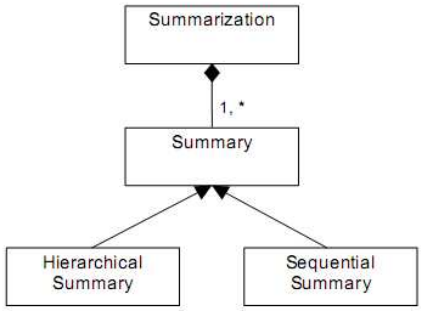

Figura 9 – Visão geral das ferramentas de sumarização
Fonte: ISO/IEC 15938-5, 2003.

A figura 9 exibe um diagrama demonstrando o comportamento geral das ferramentas de descrição de sumarização. O elemento *Summarization* é composto por um ou mais elementos do tipo *Summary*. Como *Summary* é um tipo abstrato, o desenvolvedor deverá escolher entre o tipo *HierarchicalSummary* ou *SequencialSummary*, conforme demonstrado no quadro 5.

```
<Summarization>
      <Summary xsi:type="HierarchicalSummaryType"
            components="keyAudioClips"
            hierarchy="independent">
            <!--Conteúdo do sumário-->
      </Summary>
</Summarization>
```

Quadro 5 – Exemplo de elemento *Summary* que utiliza o tipo *HierarchicalSummary*
Fonte: Autoria própria, 2008.

O *HierarhicalSummary* DS é utilizado para organizar grupos de sumários em uma forma hierárquica. Através deste esquema de descrição os sumários são organizados em uma estrutura de árvore onde cada nó representa um segmento

temporal de um conteúdo de áudio ou vídeo (por exemplo, na aplicação criada foram utilizados segmentos de um memorando de áudio). De acordo com a especificação MPEG-7 os sumários hierárquicos "descrevem o conteúdo multimídia em níveis de detalhes, de sumários brutos a mais detalhados" (ISO/IEC 15938-5, 2008), ou seja, o detalhamento é feito de uma maneira "*top-down*" (de cima para baixo), conforme o usuário desce os níveis do sumário, mais específico ele se torna.

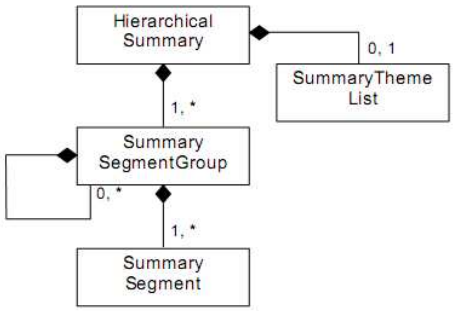

Figura 10 - Visão geral do *HierarchicalSummary* DS
Fonte: ISO/IEC 15938-5, 2008.

A sintaxe do *HierarchicalSummary* DS é representada pelo diagrama apresentado na figura 10. O elemento *HierarchicalSummary* é composto por um ou mais elementos *SummarySegmentoGroup* que representa uma “visualização alternativa do conteúdo” (ISO/IEC 15938-5, 2008) pelo agrupamento de segmentos de sumários e outros grupos de sumário. Cada *SummarySegmentGroup* deve possuir pelo menos um elemento *SummarySegment*.

Os elementos *SummarySegment* correspondem a um segmento temporal do conteúdo inserido no sumário, ou seja, representa um trecho do conteúdo. Estes segmentos são baseados por conteúdo de áudio ou vídeo descritos pelos elementos *KeyAudioVisualClip*, *KeyAudioClip* ou *KeyVisualClip*. Estes elementos podem tanto representar uma descrição da representação temporal de um trecho do conteúdo descrito pelo sumário (através do tipo *MediaTime*), ou por uma conteúdo previamente segmentado armazenado em outro arquivo (através do elemento *MediaUri*, que foi o caso da aplicação criada). No quadro 6 encontra-se um exemplo dos elementos *SummarySegmentGroup* e *SummarySegment*.

Retomando a explicação da figura 10, vê-se a representação do elemento *SummaryThemeList*, que pode ser utilizado para criar uma lista de categorias que são utilizadas pelo elemento *SummarySegmentGroup*.

```
<SummarySegmentGroup level="1">
      <SummarySegment>
            <Name>Estratégia de Avaliação</Name>
            <KeyAudioClip>
                  <MediaUri>
                      http://localhost/audios/pesquisa02.3.mp3
                  </MediaUri>
            </KeyAudioClip>
      </SummarySegment>
</SummarySegmentGroup>
```

Quadro 6 – Exemplo de uma descrição utilizando o *SummarySegmentGroup*
Fonte: Autoria própria, 2008.

3.3 XPath

XPath é um dos padrões utilizados pelo padrão MPEG-7 para referência entre descritores e esquemas de descrição (ISO/IEC 15938-5, 2003). De acordo com a *W3C*, "é uma linguagem para endereçamento de um documento XML" (XPath, 1999).

XPath possui as seguintes características:

- Possui uma sintaxe em um formato não XML
- Opera sobre a estrutura lógica de um documento XML, ao invés de sua sintaxe
- O documento XML é acessado como uma árvore de nodos – existem diferentes tipos de nodo, que podem ser elementos, atributos ou texto.

Para ilustrar o uso de XPath, tem-se o exemplo do quadro 7, que apresenta um documento XML descrevendo uma coleção de livros.

```
<livros>
      <livro paginas="394">
            <autor>Gabiel Garcia Marquez</autor>
            <titulo>Cem Anos de Solidão</titulo>
      </livro>
      <livro paginas="80">
            <autor>Aldous Huxley</autor>
            <titulo>As Portas da Percepção</titulo>
      </livro>
</livros>
```

Quadro 7 – Descrição de uma coleção de livros utilizando XML
Fonte: Autoria própria, 2008.

O uso da expressão XPath*, /child::livros/child::livro (number(attribute::paginas) < 100)* resulta em objeto do tipo *node-set*, ou conjunto de nodos, que representa a parte da árvore do documento que foi selecionada pela expressão (XPath, 1999). A

expressão representa os elementos “livro” cujo atributo “paginas” seja menor que 100 e sejam filhos de elementos do tipo “livros”, que por sua vez são filhos da raiz “/”. Se o conjunto de nodos for convertido em texto resultaria no código XML apresentado no quadro 8.

```
<livro paginas="80">
      <autor>Aldous Huxley</autor>
      <titulo>As Portas da Percepção</titulo>
</livro>
```

Quadro 8 – Resultado da expressão “*/child::livros/child::livro (number(attribute::paginas) < 100)*”
Fonte: Autoria própria, 2008.

Cada resultado de uma expressão XPath (contexto) podem ser: um nodo (*context node*), um par de inteiros positivos diferente de zero (*context position* e *context size*), um conjunto de variáveis vinculadas, uma função da biblioteca, ou conjunto das declarações de *namespace* no escopo da expressão (XPath, 1999).

Location Path é como são chamadas as expressões XPath que identificam um local no documento. O *location path* (passo de localização) seleciona conjunto de nodos relativos a um contexto e é o mais importante construto da linguagem XPath (XPath, 1999).

Existem dois tipos de *location path*, absoluto e relativo.

Relativo: É uma seqüência de "*location steps*" separados por '/'. Os passos são compostos juntos da esquerda para direita. Cada passo seleciona um conjunto de nodos, cada nodo no contexto é utilizado para selecionar os próximos passos (XPath, 1999).

Absoluto: Consiste do caractere '/' seguido de um caminho relativo. O caractere '/' seleciona o nodo raiz do documento (XPath, 1999).

Um passo de localização é dividido três partes:

- *Axis* (eixo) especifica a relação entre os nodos selecionados
- Teste de nodo, especifica o tipo de nodo selecionado pelo pelo *location step*
- Zero ou mais predicados utilizados para filtrar os nodos *axes* (eixos)

Considerando o passo de localização *child::livro(number(attribute::paginas) < 100), child* é o nome do eixo, *livro* é o teste de nodo e *(number(attribute::paginas) < 100)* é o predicado.

O padrão *XPath* considera os seguintes eixos (XPath, 1999):

child: Seleciona os filhos do contexto.

descendant: Seleciona os descendentes do contexto, por exemplo, filhos, netos ...

ancestor: Seleciona os ancestrais do contexto, por exemplo, pai, avô, ...

following-sibling: Seleciona todos os irmãos do contexto.

preceding-sibling: Seleciona todos os irmãos predecessores do contexto.

folowing: Seleciona todos os nodos que são posteriores ao contexto.

preceding: Seleciona todos os nodos anteriores ao contexto.

attribute: Atributos do contexto.

namespace: Namespace do contexto.

self: Seleciona o próprio nodo.

descendant-or-self: Seleciona o nodo e seus descendentes.

ancestor-or-self: Seleciona o nodo e seus ancestrais.

Não é necessário utilizar a formação completa dos *location paths.* A linguagem XPath possibilita algumas abreviações por exemplo:

* : Seleciona todos os elementos do contexto

@paginas: Seleciona o atributo paginas do contexto

..: Seleciona o pai do contexto

livro: Seleciona os elementos livro do contexto

//titulo: Seleciona os elementos “títulos” que estão em qualquer parte do documento

3.4 XQuery

XQuery é uma linguagem de consulta para dados armazenados em XML sendo este uma recomendação W3C desde 23 de janeiro de 2007. Esta linguagem foi criada devido aumento de informações armazenadas, trocadas e apresentadas em XML (XQuery, 2007) sendo baseada em linguagens como XPATH, SQL e outras linguagens de consulta de dados no formato XML.

Segundo a W3C, a linguagem XQuery foi projetada para estar de acordo com os requerimentos identificados pela *W3C XML Query Working Group*, ter consultas concisas e de fácil compreensão e ser flexível para consultar um grande espectro de informações XML, como por exemplo, banco de dados e documentos (XQuery, 2007).

O bloco de construção da linguagem XQuery são as expressões, as quais possuem as seguintes características:

- Diferem em palavras chaves, símbolos e operadores
- Suas expressões podem ser aninhadas
- São sensíveis ao caso

A gramática de uma expressão XQuery é representada no exemplo descrito no quadro 9.

```
Expr::= ExprSingle ("," ExprSingle)*
ExprSingle ::= FLWORExpr
            | QuantifiedExpr
            | TypeswitchExpr
            | IfExpr
            | OrExpr
```

Quadro 9 – Gramática de uma expressão XQuery
Fonte: XQuery, 2007.

Um expressão XQuery é constituída de pelo menos uma expressão formada pela gramática *ExprSingle* e zero ou mais seqüências da mesma gramática separadas por vírgula. A gramática *ExprSingle* pode ser do tipo *FLWORExpr* (*FLWOR Expressions*), *QuantifiedExpr* (*Quantified Expressions*), *TypeswitchExpr* (*Typeswitch Expressions*), *IfExpr* (*Conditional Expressions*) ou *OrExpr* (*Logical Expressions*) (XQuery, 2007).

FLWOR *Expressions* pronuncia-se "*flower*", são expressões que suportam iteração e vínculo com variáveis intermediárias ao resultado, e são úteis para juntar resultados entre dois ou mais documentos. O nome FLWOR deriva das a palavras chaves *for*, *let*, *where*, *order by* e *return* (XQuery, 2007).

As cláusulas *for* e *let* geram uma seqüência ordenadas de tuplas da variável vinculada, as quais é dado o nome de *tuple stream*. A cláusula *where* opcional serve para filtrar tuplas que não são desejadas. A cláusula *return* é o valor de saída para cada tupla da expressão selecionada (XQuery, 2007).

O exemplo do quadro 10 utiliza o mesmo documento XML utilizado no exemplo do quadro 7 da seção anterior. Esta expressão retorna um novo documento XML, mas ao invés de *paginas* ser um atributo do elemento *livro*, ele se torna um novo elemento filho de *livro*.

```
<livros>
{
      for $i in /livros/livro
            let $a := $i/autor/text()
            let $p := number($i/attribute::paginas)
            let $t := $i/titulo/text()
      return <livro>
                  <titulo>{$t}</titulo>
                  <autor>{$a}</autor>
                  <paginas>{number($p)}</paginas>
            </livro>
}
</livros>
```

Quadro 10 – Exemplo de uma expressão do tipo FLWOR
Fonte: Autoria própria, 2008.

Quantified Expression são expressões de retorno *boleano* (retornam o valor *true* ou *false*) que iniciam com um quantificador *some* ou *every*, seguido de uma ou mais cláusulas *in* (para vínculo com as variáveis), seguido da expressão *satisfies* e um teste *boleano* (ISO/IEC 15938-5, 2003). Por exemplo, a expressão *some $i in /livros/livro satisfies ($i(@paginas)>"100")* retornará *true*, pois existe um elemento *livro* em que seu atributo *paginas* é maior que 100.

Typeswitch Expressions são utilizadas para avaliar o tipo de uma expressão. Esta expressão consiste da palavra chave *typeswitch* seguida de uma expressão dentro de parênteses. Esta expressão terá o seu tipo testado para cada palavra chave *case*. Caso o tipo não seja identificado em um dos *cases* o valor da expressão será retornado através da palavra chave *default* (XQuery, 2007).

O exemplo do quadro 11 testa a ocorrência do primeiro elemento dentro do nodo *livros*, como este elemento é um nodo o retorno da expressão será a *string* “Nodo”. Note que neste exemplo foram utilizados apenas duas expressões de teste de tipo, o padrão XQuery apresenta ainda mais.

```
typeswitch(/livros(1])
      case $n as node() return "Nodo"
      case $t as text() return "Texto"
      default return "tipo não identificado"
```

Quadro 11 – Exemplo do uso da expreassão *typeswitch*
Fonte: Autoria própria, 2008.

Conditional Expressions, são expressões do tipo *if-then-else*. A expressão possui a seguinte sintaxe, a palavra chave *if*, uma expressão de teste dentro de parenteses, a palavra chave *then,* mais uma expressão, finalizando-se com a palavra chave *else* mais uma expressão (XQuery, 2007).

```
if (number(/livros/livro(1]/attribute::paginas) >= 100) then
      /livros/livro(1]/titulo/text()else
number(/livros/livro(1]/attribute::paginas)
```

Quadro 12 – Exemplo de uso da expressão *if-then-else*
Fonte: Autoria própria, 2008.

A expressão do quadro 12 acima retornará o título do livro caso este possua mais de 100 páginas, caso contrário retornará o valor armazenado em seu atributo *paginas*.

Logical Expressions são as expressões que retornam valores *boleanos* através de algum tipo de operação lógica como: *and*, *or*, <, >=, >, >=, *eq*, *idiv* (XQuery, 2007).

3.5 Revisão sobre a tecnologia

O presente capítulo apresentou um panorama sobre as tecnologias utilizadas na implementação do trabalho. Webservices foram utilizados para criação de componentes de software independentes de arquitetura.

Do padrão MPEG-7 foram utilizados os esquemas de descrição *HierachicalSummary* para descrição de sumários de conteúdo - utilizados na aplicação de busca e execução de sumários hierárquicos. E *Creator*, utilizado pela aplicação de extração, para converter os atributos de um arquivo MP3 em uma codificação MPEG-7.

A linguagem XPath foi utilizada para endereçar elementos de um conteúdo MPEG-7, ao passo que XQuery foi utilizado como maneira de efetuar consultas em um repositório de descrições MPEG-7.

4 MODELO ORIENTADO A SERVIÇOS MPEG-7

A figura 11 exibe o modelo de uma arquitetura orientada a serviços MPEG-7. Este modelo foi desenvolvido baseado nos textos do grupo MPEG a respeito de funções que as aplicações baseadas no padrão poderiam disponibilizar (Hunter, 2001), bem como no estudo de trabalhos que utilizam MPEG-7 (conforme apresentado na seção referencial teórico) e percepções próprias relacionadas com a maneira que os componentes de *software* deveriam ser publicados e categorizados.

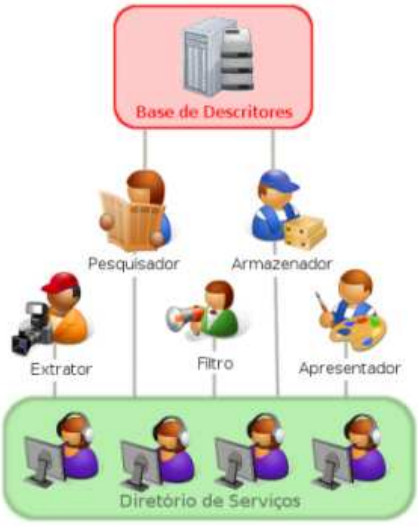

Figura 11 – Modelo Orientado a Serviços MPEG-7
Fonte: Autoria própria, 2008.

O modelo é representado pelos itens: base de descritores, serviços e um diretório de serviços. Neste trabalho não foi considerada a implementação do terceiro item, diretório de serviços, já que esta é uma aplicação conceitual, onde todos os serviços foram armazenados no mesmo servidor, e também por que apenas três aplicações fizeram uso deste serviço, sendo fácil de alterar seus endereços em caso de troca.

4.1 Base de descritores

A base de descritores é onde são armazenadas as descrições de conteúdo multimídia no formato MPEG-7. Estas descrições podem ser extraídas ou criadas por aplicações que implementem estas funcionalidades. A base de descritores é acessada somente pelos serviços de pesquisa e armazenamento, por serem eles os serviços que necessitam de um acesso direto a este recurso.

De forma geral a base de descritores pode ser implementada de diferentes maneiras, como por exemplo, uma estrutura de diretório, um banco de dados XML, um banco de dados relacional. É responsabilidade dos serviços de pesquisa e armazenamento conhecerem o formato desta estrutura para que os dados sejam armazenados ou consultados neste repositório.

Nesta implementação foi utilizado como base de descritores um banco de dados XML com suporte a linguagem XQuery. O uso de banco de dados XML é em razão de que as descrições MPEG-7 são declaradas no formato XML (Martínez, 2005), (Martínez, 2004), (Salembier e Smith, 2001), (ISO/IEC 15938-5, 2003), e também por que estes bancos são desenvolvidos especialmente para lidar com arquivos neste formato. O uso da linguagem XQuery é por ela ser uma padrão de consultas de dados XML recomendado pela W3C e também por possuir recursos específicos para lidar com este tipo de estrutura de dado (XQuery, 2007).

O banco de dados XML utilizado por este trabalho é o *eXistDB* (eXistDB, 2000). Este banco de dados é distribuído através de licença *LGPL*, possui suporte a XQuery e também uma suporte de desenvolvimento *Java* através de uma interface *XML:DB API*. *XML:DB API* é uma interface de desenvolvimento para acesso a bancos de dados XML criada pela *XML:DB initiative*. Esta organização tem por objetivo “discutir todos os itens gerais relacionados a banco de dados XML”, além de desenvolver tecnologias e especificações para gerenciamento de banco de dados XML e contribuir com implementação de suas referências. (XML:DB API, 2003).

Outros bancos de dados XML foram testados como *MonetDB* (*http://monetdb.cwi.nl/*) que também possui suporte a XQuery, embora não possua suporte a *XML:DB API,* e o *Apache Xindice* (*http://xml.apache.org/xindice/*) que possui suporte a *XML:DB API* mas que não possui suporte a XQuery.

4.2 Serviços

Esta camada representa os serviços que manipulam os dados de descritores MPEG-7 e que foram categorizados de acordo com as operações pelas quais eles são responsáveis. Foram definidas cinco categorias de serviços:

- Pesquisador
- Armazenador
- Extrator
- Filtro e
- Apresentador

Por categoria subentende-se que podem existir múltiplos serviços de uma mesma categoria, mas cada serviço (desta categoria) implementa características ou funcionalidades diferentes. Por exemplo, um serviço para extração de descritores de baixo nível de uma imagem (como cor predominante, formato de arquivo) e outro para extração de dados de criação de um arquivo de áudio MP3 (título, artista, etc). Ambos os serviços são da categoria de extratores e possuirão a mesma interface de operações só que o resultado e o seu desenvolvimento serão diferentes, de forma análoga a uma classe abstrata e suas implementações concretas.

As operações de cada serviço foram concebidas de acordo com os trabalhos estudados (descritos no capítulo 2), pelas aplicações descritas nos documentos do grupo MPEG e algumas percepções próprias – operações que não estão explicitamente descritas em algum trabalho ou documento, mas vê-se com clareza a funcionalidade dela.

Tanto estas operações, quanto as categorias de serviços e os serviços que foram criados serão descritos a seguir.

Todos os serviços desse trabalho foram desenvolvidos em linguagem *Java*, utilizando o ambiente de desenvolvimento integrado *NetBeans* 6.0, que propicia o fácil desenvolvimento de *webservices*. Os *webservices* foram instalados e são acessados a partir do servidor de aplicação *GlassFish* 2.0, que possui integração com o *NetBeans*. Uma visão geral do desenvolvimento dos serviços será descrita juntamente com a descrição dos serviços.

4.2.1 Serviço Pesquisador

Este serviço tem como finalidade consultar o repositório de documentos MPEG-7 para retornar um conjunto de esquemas de descrição que estão de acordo com os critérios de consulta fornecidos pelo usuário. Como se trata de um serviço, as aplicações não têm necessidade de conhecer o funcionamento do motor de busca, somente em como utilizar a sua interface de acesso.

Uma aplicação pode escolher o serviço de pesquisa que mais se adéqua as suas características, ou também utilizar vários serviços de pesquisa para aumentar a quantidade de resultados para uma determinada consulta.

O serviço de pesquisa possui duas operações:

- *search (_type, _query, _elementTypes, _page, _itensPerPage)*
- *help()*

A operação *search()* realiza a pesquisa na base de descritores MPEG-7 baseado nos parâmetros de entrada e retorna uma saída XML com a lista dos fragmentos MPEG-7 que são do tipo especificado pelo parâmetro *_elementTypes*. Caso não haja resultados destes tipos a lista retornará vazia.

Os parâmetros desta operação têm o seguinte significado:

_type (*String*): Indica o tipo de consulta que será realizada. A implementação criada neste trabalho aceita os valores *FREE_TEXT* ou *XQUERY*. *FREE_TEXT* realiza uma consulta com palavras livres. Por exemplo, se o texto “gol de placa entrar” como parâmetro, o serviço irá procurar pela ocorrência de todas estas palavras no conteúdo. Caso o valor passado seja *XQUERY*, o usuário deverá entrar no valor do parâmetro *_query* uma consulta no formato XQuery, por exemplo, */Mpeg7/Description/**

_query: É a consulta que será realiza. Nesta implementação deve ser uma consulta texto livre ou XQuery, que é indicada pelo parâmetro *_type*.

_elementTypes: Lista de tipos MPEG-7 separados por vírgula que são retornados pelo serviço de pesquisa. Por exemplo: “*MediaTimeType*,*HierarchicalSummaryType*, *SummaryDescriptionType*”

_page: Página que será exibida, no caso de existir um limite de registros por página.

_itensPerPage: Quantidade de registros exibidos por página. Através deste parâmetro e do parâmetro *_page* o usuário pode criar intervalos que limitem a

quantidade de registros retornados. São parâmetros úteis para aplicações de pesquisa quando se necessita limitar a quantidade de itens exibidos ao usuário. Caso os valores dos parâmetros *_page* e *_itensPerPage* sejam zero, a operação retorna todos os registros encontrados.

A operação *search()* retorna como resultado uma saída em XML no formato descrito no quadro 13.

```
<SearchCollection offset="1"
      limit="1"
      query="teste"
      itemCount="1">
      <SearchItem documentId="/mpeg-7/db/pesquisa01.xml"
                  xpath="/*:Mpeg7">
            <!--Fragmento de descrição MPEG-7-->
      </SearchItem>
</SearchCollection>
```

Quadro 13 – XML resultante da operação *search()*
Fonte: Autoria própria, 2008.

O elemento *SearchCollection* serve como preâmbulo para os itens encontrados pelo serviço de busca. O atributo *offset* indica a página que o serviço retornou, *limit* indica a quantidade máxima de itens de retorno, *query* é a consulta que foi executada e *itemCount* a quantidade de registros encontrados. O elemento *SearchItem*, por sua vez, é o preâmbulo dos fragmentos de descrição encontrados. O atributo *documentId* informa o identificador de acesso do documento e pode ser utilizado no Serviço Armazenador para recuperar, adicionar ou remover fragmentos do documento MPEG-7. O atributo *xpath* informa o caminho XPath que endereça o fragmento encontrado.

A operação *help()* retorna um texto de ajuda no uso do serviço pesquisador.

4.2.2 Implementação do Serviço Pesquisador

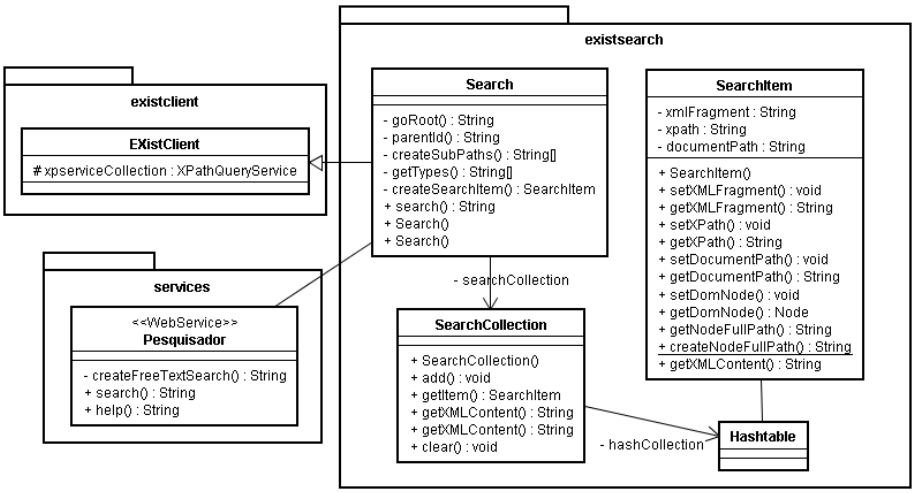

Figura 12 – Diagrama de classes do serviço pesquisador
Fonte: Autoria própria, 2008.

A figura 12 demonstra o diagrama de classes do serviço pesquisador. Este serviço é implementado pelas classes *Search*, *SearchCollection* e *SearchItem* que fazem parte do pacote *existsearch*.

Search: Faz pesquisas na coleção de arquivos MPEG-7. Ela possui a capacidade de encontrar o endereço XPath dos resultados encontrados, como também identificar o tipo de cada passo do endereço XPath resultante. Esta identificação de tipo serve para que a classe sempre retorne os tipos requisitados pelo usuário.

SearchCollection: Armazena os itens encontrados pela pesquisa. O método *getXMLContent()* retorna a coleção em seu formato XML.

SearchItem: Armazena os dados referentes ao item encontrado.

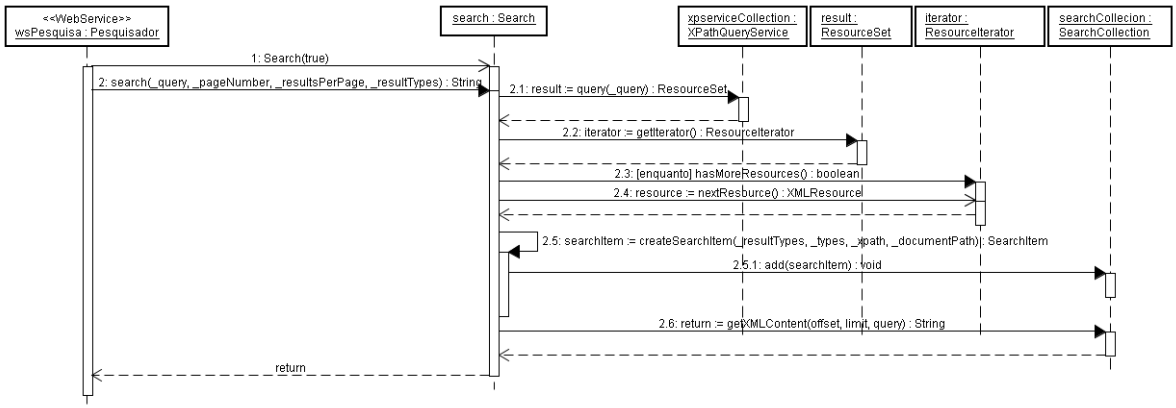

Figura 13 – Diagrama de seqüência da operação search
Fonte: Autoria própria, 2008.

A figura 13 representa a seqüência de execução simplificada da operação do serviço de pesquisa.

1 O objeto de pesquisa é inicializado

2 O método de pesquisa é invocado

2.1 A consulta é executada

2.2 Recupera-se o iterador de resultado da pesquisa

2.3 Enquanto existirem resultados faça

2.4 Recupera-se o próximo resultado

2.5 Cria-se um novo item de resultado de pesquisa com os dados de fragmento XML, caminho XPath e identificador do documento. Se o item encontrado não pertencer a lista de tipos informada pelo usuário, ele não é inserido na coleção

2.5.1 Adiciona-se o resultado de pesquisa criado na coleção de resultados

2.6 Recupera-se o conteúdo XML da pesquisa, que é então retornado ao serviço de pesquisa.

Como foi mencionado anteriormente, este serviço de busca reconhece dois tipos de pesquisa:

XQUERY, ou seja, nada mais que uma consulta no padrão XQuery que pode ser utilizada para que as aplicações de buscas tenham flexibilidade suficiente para montar suas consultas, assim podendo criar opções de consultas avançadas e parametrizações específicas para uma determinada aplicação.

FREE_TEXT, literalmente texto livre. Nesta opção o usuário digita uma lista de palavras que ele deseja encontrar e o serviço monta uma consulta XQuery especial com o uso de uma função própria do banco de dados XML *eXistDB* chamada *text:fuzzy-match-all()*. Esta função recebe como primeiro parâmetro um endereço XPath ou consulta XQuery que retorne uma série de nodos. Os parâmetros seguintes são as palavras que serão comparadas com os nodos resultantes do primeiro parâmetro. No quadro 14 se encontra a maneira como a função é utilizada pelo serviço.

```
for $i in
      (text:fuzzy-match-all(//*,
            'palavra_1', 'palavra_2', 'palavra_n')
return $i
```

Quadro 14 – Utilização da função *text:fuzzy-match-all()*.
Fonte: Autoria própria, 2008.

O serviço utiliza o caminho //* que indica que todas as palavras enviadas no parâmetro serão pesquisadas nos elementos armazenados na base de dados MPEG-7. A expressão no formato FLWOR indica que todos os elementos que forem reconhecidos pela função serão retornados.

4.2.3 Serviço Armazenador

Serviços armazenadores possuem a responsabilidade de gerir o repositório de documentos MPEG-7 através das seguintes funcionalidades:

- Adicionar novos documentos MPEG-7 ao seu repositório
- Recuperar um documento ou fragmento de um documento MPEG-7 baseado no identificador do documento e o endereço XPath do fragmento
- Inserir fragmentos MPEG-7 em documentos que já estão no repositório. Funcionalidade que pode ser aproveitada por ferramentas de anotação, e
- Remover fragmentos MPEG-7 de um documento

A partir destas funcionalidades foram definidas cinco operações para o serviço:

- *newMpeg7 (_mpeg7)*
- *get(_fileId, _xpath)*
- *add(_fileId, _xpath, _fragment)*
- *remove(_fileId, _xpath)*
- *help()*

A operação *newMpeg7()* insere um novo documento MPEG-7 dentro do repositório de descritores. O parâmetro *_mpeg7* (*String*) representa o conteúdo do documento que será inserido na base de descritores. Após a inserção a operação retorna o identificador de acesso ao documento, que é utilizado pelas outras operações do serviço.

A operação *get()* retorna um fragmento (ou o conteúdo completo) de um documento MPEG-7 armazenado na base de descritores. Para tanto a operação aguarda dois parâmetros:

_fileId (*String*): Identificador de acesso do documento pelo o qual se deseja recuperar informações

_xpath **(***String*): Endereço do fragmento que se deseja recuperar descrito na linguagem XPath

A operação *add()* é utilizada para inserir um fragmento de conteúdo MPEG-7 dentro de um documento armazenado na base de descritores MPEG-7. Caso o fragmento seja adicionado com sucesso, a operação retorna uma *string* com o valor "*true*". Caso contrário o valor "*false*" é retornado. Os parâmetros desta operação possuem os seguintes significados:

_fileId (*String*): Identifica o documento onde o fragmento será inserido

_xpath (*String*): Endereço onde o fragmento será inserido, ou seja, o fragmento será inserido como filho do nodo endereçado por este parâmetro.

_fragment (*String*): Fragmento de conteúdo MPEG-7 que será adicionado como filho do nodo endereçado pelo parâmetro *_xpath*

A operação *remove()* exclui o nodo endereçado pelo parâmetro *_xpath*. O parâmetro *_fileId* identifica o documento que terá o nodo excluído. Ao final da execução a operação retorna uma *String* com valor "*true*" caso o nodo tenha sido removido ou o valor "*false*" caso contrário.

A operação *help()* retorna um texto de ajuda no uso do serviço armazenador.

4.2.4 Implementação do Serviço Armazenador

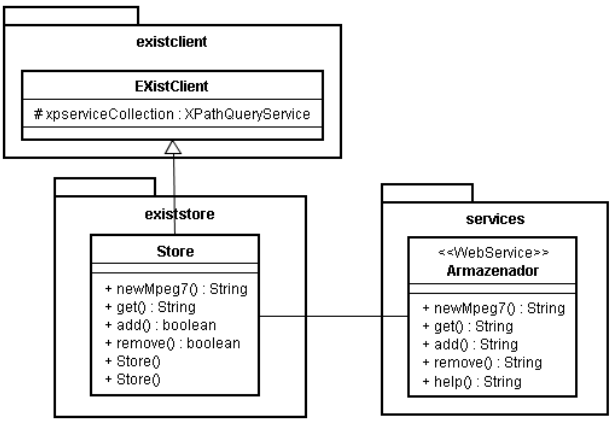

Figura 14 – Diagrama de classes do serviço armazenador
Fonte: Autoria própria, 2008.

Na figura 14 é exibido o diagrama das classes que permeiam o domínio do o serviço armazenador. O núcleo do serviço é constituído do pacote *existstore*, que possui a apenas a classe *Store*. Tanto a classes *Store*, quanto a classe *Search* (explicada na seção Serviço Pesquisador) estendem a classe *ExistClient*. O diagrama de classes demonstra também, a dependência do *webservice* Armazenador pela classe *Store*.

A classe abstrata *ExistClient* implementa as rotinas de inicialização de conexão com o banco de dados XML, a conexão com as coleções de dados MPEG-7 e *Schemas* MPEG-7 (que são armazenados para verificação de tipo) e também a inicialização serviço de consulta XQuery. Estas rotinas são comuns as classes *Store* e *Search* (documentada anteriormente).

A classe *Store* implementa as funcionalidades para inserir um novo documento MPEG-7, recuperar um fragmento ou documento MPEG-7, adicionar um fragmento MPEG-7 em um documento previamente armazenado ou, remover um fragmento de um documento armazenado. Estas funcionalidades foram desenvolvidas para o banco de dados *eXistDB* utilizando a sua interface *XML:DB API*.

O diagrama da figura 15 demonstra o fluxo de execução da operação *newMpeg7()* do *webservice*.

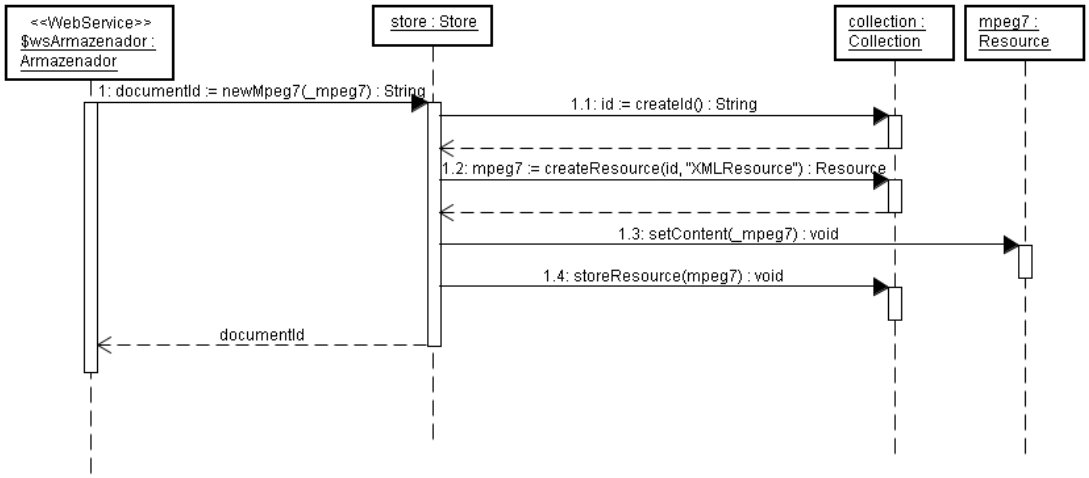

Figura 15 – Diagrama de sequência da operação *newMpeg7*
Fonte: Autoria própria, 2008.

1 O *webservice* invoca o método *newMpeg7()* da classe *Store*, passando o conteúdo do documento *MPEG-7* que será armazenado.

1.1 O objeto *store* invoca o método *createId()* do objeto *collection* para criar um identificador para o documento que será inserido

1.2 Um objeto da classe *Resource* é criado pela invocação do método *createResource()* do objeto *collection*

1.3 O conteúdo do documento que veio através do *webservice* é inserido dentro do objeto da classe *Resource*

1.4 O método *storeResource()* é invocado para armazenar o objeto do tipo *Resource* que foi criado

A classe *Store* retorna como identificador para o *webservice* o caminho da coleção no banco de dados XML mais o identificador criado dentro do método *newMpeg7()*. Esta construção é utilizada para facilitar o acesso direto do documento através do banco de dados XML.

O fluxo da execução da operação *get()* pelo *webservice* é demonstrado pelo diagrama da figura 16 e apresentado a seguir.

1 O *webservice* invoca o método *get()* da instância da classe *store*, passando os parâmetros de identificação do arquivo e endereço XPath

1.1 O método *query()* da instância de *XPathQueryService* é invocado. A classe *Store* cria uma consulta XPath do tipo *doc(_fileId)/_xpath* para recuperar o documento ou fragmento do documento

1.2 O objeto *store* recupera o primeiro elemento do conjunto de recursos retornados.

1.3 O conteúdo do recurso é retornado ao serviço.

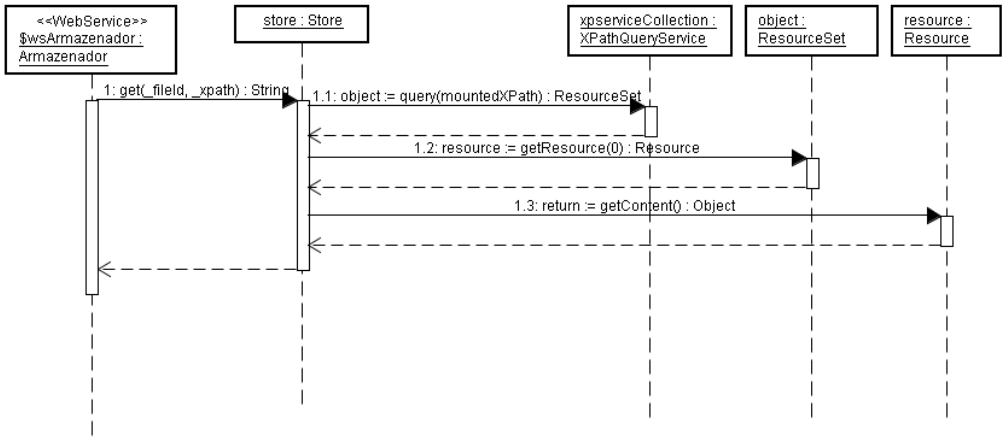

Figura 16 – Diagrama de seqüência da operação *get()*
Fonte: Autoria própria, 2008.

A operação *add()* é esquematizada através do diagrama de seqüência exibido na figura 17 e descrito a seguir.

1 O *webservice* invoca o método *add()* do objeto *store*, passando como parâmetro o identificador do documento, endereço XPath e o fragmento que deverá ser inserido.

1.1 O objeto *store* invoca o método *query()* da instância da classe *XPathQueryService*. Em caso de erro na execução o método retornará *false*, caso contrário *true*.

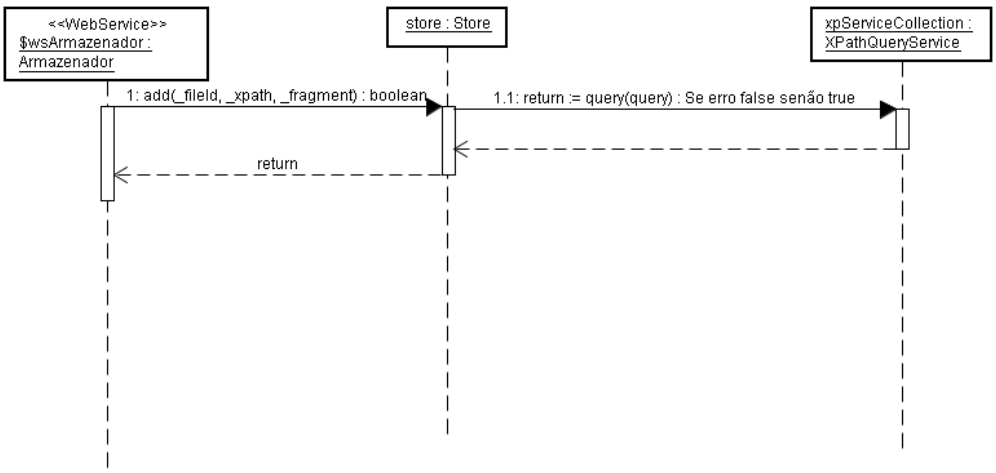

Figura 17 – Diagrama de seqüência da operação *add()*
Fonte: Autoria própria, 2008.

Para simplificar o desenvolvimento do método *add()*, foi utilizada uma funcionalidade do banco de dados *eXistDB* que permite o uso de instruções do tipo *XUpdate* dentro de consultas *XQuery* (eXistDB, 2000) para atualização de documentos. A instrução que foi utilizada é esquematizada no quadro 15, onde um elemento contendo os dados de *ISBN* é inserido dentro do nodo endereçado pelo caminho XPath */livros/livro(1)*.

```
update insert <isbn>85-01-01207-6</isbn> into /livros/livro(1]
```

Quadro 15 - Exemplo de inserção utilizando *XUpdate*
Fonte: Autoria própria, 2008.

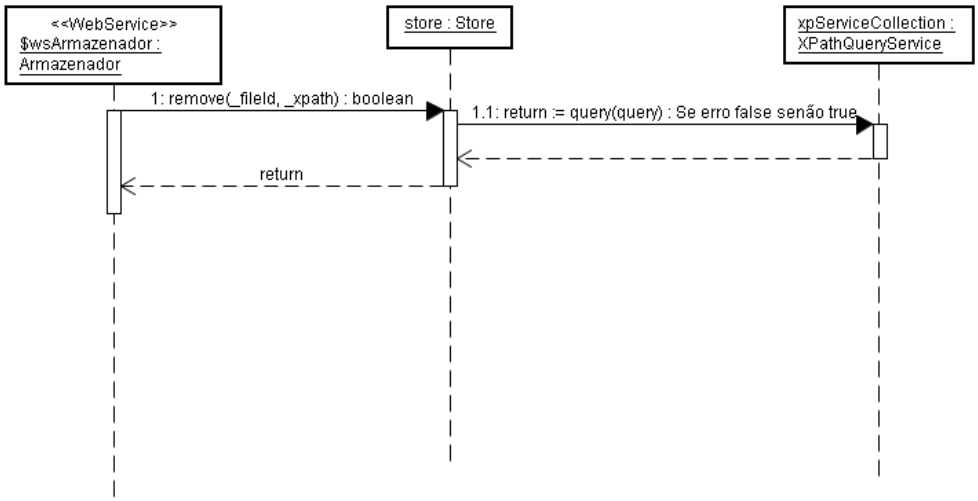

Figura 18 – Diagrama de seqüência da operação *remove()*
Fonte: Autoria própria, 2008.

A implementação da operação de remoção segue praticamente o mesmo esquema da operação de adição. A diferença é que o serviço armazenador passa apenas os parâmetros de identificação de documento e endereço XPath. Nesta operação também foi utilizada a instrução de *XUpdate* para efetuar a remoção do nodo. No exemplo do quadro 16 o elemento *isbn* é removido do nodo endereçado pelo caminho XPath */livros/livro(1)*.

```
update insert delete /livros/livro(1]/isbn
```

Quadro 16 – Exemplo de remoção utilizando *XUpdate*
Fonte: Autoria própria, 2008.

4.2.5 Serviço Extrator

Os serviços extratores são encarregados por criar um documento MPEG-7 a partir dos dados coletados automaticamente de um item multimídia. Assim, descrições que não precisam ser criadas com assistência humana ou são repetitivas podem ser extraídas através deste serviço (Mitrovic, Zeppelzauer e Eidenberger, 2006).

Este serviço possui duas operações *extract(_url)* e *help()*.

A operação *extract()* retorna um valor em *string* que codifica um documento MPEG-7. Esta codificação é gerada a partir da extração dos dados (e metadados) encontrados no arquivo apontado pela URL (*Uniform Resource Locator*) passada como parâmetro. Será critério do serviço abrir o conteúdo do arquivo, endereçado

pela URL, e extrair os dados que o serviço se compromete, gerando uma codificação MPEG-7 adequada.

A operação *help()*, padrão nos serviços do modelo, retorna um texto explicativo de uso do serviço.

É importante salientar que podem existir múltiplos serviços de extração. Por exemplo, podem existir serviços encarregados por extrair dados de criação de uma imagem como local, data, ferramenta de captura, etc. Também podem existir serviços que capturem os mesmos metadados de itens de áudio e vídeo.

4.2.6 Implementação do Serviço Extrator

A implementação do serviço extrator, criada por este trabalho, extrai informações de editor, comentário, artista, título e ano de arquivos MP3 e que estão codificadas no formato ID3. ID3 é um formato de identificação de metadados de arquivos de áudio. É utilizado tanto em softwares quanto hardwares (ID3, 1998) (aparelhos de som em geral) de maneira que possam ser exibidos (na execução dos conteúdos de áudio) dados como título, artista, entre outras informações. Para extrair estas informações utilizou-se a biblioteca *Java ID3 Tag Library* (*http://javamusictag.sourceforge.net/*).

O serviço também extrai a informação de duração do conteúdo, para isto foi utilizada a biblioteca *Java library MP3* (*http://www.vdheide.de/java_mp3/*).

A partir do conteúdo extraído é gerado um documento MPEG-7 que obedece ao formato apresentado no quadro 17.

```
<Mpeg7>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="AudioType">
      <Audio>
        <CreationInformation>
          <Creation>
            <Title type="songTitle">
              <!--Título do Arquivo-->
            </Title>
            <Abstract>
              <FreeTextAnnotation>
                <!--Comentário do arquivo-->
              </FreeTextAnnotation>
              <StructuredAnnotation>
                <Who>
                  <Name>
                    <!--Artista/Banda-->
                  </Name>
                </Who>
                <When>
                  <Name>
                    <!--Ano-->
                  </Name>
                </When>
              </StructuredAnnotation>
            </Abstract>
            <Creator>
              <Role href="creatorCS">
                <Name>Editor</Name>
              </Role>
              <Agent xsi:type="PersonType">
                <Name>
                  <GivenName>
                    <!--Nome do Editor-->
                  </GivenName>
                </Name>
              </Agent>
            </Creator>
          </Creation>
        </CreationInformation>
        <MediaTime>
          <MediaTimePoint>T00:00:00</MediaTimePoint>
          <MediaDuration>PT0M0S</MediaDuration>
        </MediaTime>
      </Audio>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

Quadro 17 – Formato do conteúdo extraído pelo Serviço Extrator
Fonte: Autoria própria, 2008.

Os elementos chaves do conteúdo extraído (*CompleteDescriptionType*, *MediaTimeType*, *MediaLocatorType*, *TextAnnotationType* e *Creation DS*) foram explicados anteriormente neste trabalho.

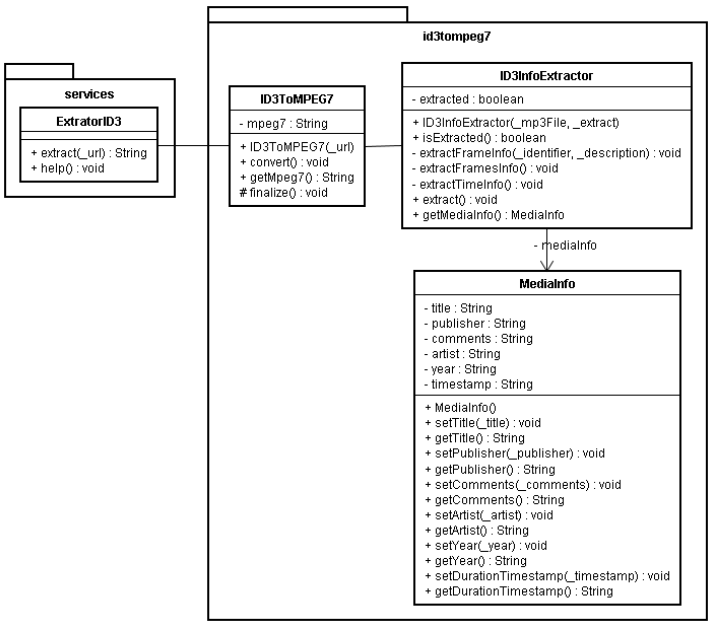

Figura 19 – Diagrama de classes do Serviço Extrator
Fonte: Autoria própria, 2008.

O comportamento do serviço *ExtratorID3* é implementado pelo pacote *id3tompeg*, que é representado pelo diagrama de classes acima. Este pacote possui três classes:

- *ID3ToMPEG7*, que é responsável por extrair os dados ID3, de um arquivo que possua este tipo de cabeçalho e transformar em uma saída codificada em MPEG-7. Os arquivos de entrada devem ser do formato MP3.
- *ID3InfoExtractor*, extrai as informações ID3 armazenadas em arquivos MP3 e as coloca dentro de um objeto do tipo *MediaInfo*.
- *MediaInfo*, armazena as informações de título, editor, comentários, artista, ano, e duração extraídas de uma mídia.

O diagrama de seqüência da figura 20 demonstra, simplificadamente, o comportamento das classes do pacote *id3tompeg.* A representação desta seqüencia é descrita a seguir.

1 O objeto *conversor* é inicializado. Neste momento o arquivo que é apontado pela URL é criado e armazenado temporariamente.

2 O método *convert()* é invocado, para que seja feita a extração e a conversão dos metadados do arquivo para o formato MPEG-7.

2.1 O método *extract()* é invocado. Neste momento serão utilizados as bibliotecas *Java ID3 Tag Library* e *Java library* MP3 para extração das

informações de título, editor, comentários, artista, ano, e duração. Estas informações são então armazenadas dentro de um objeto do tipo *MediaInfo*.

2.2 O conversor invoca o método *getMediaInfo()* do objeto extrator. Este método retorna um objeto do tipo *MediaInfo* que possui os metadados extraídos.

2.3 A informação de título é concatenada dentro do documento MPEG-7

2.4 A informação de comentário é concatenada dentro do documento MPEG-7

2.5 A informação do artista é concatenada dentro do documento MPEG-7

2.6 A informação de ano é concatenada dentro do documento MPEG-7

2.7 A informação de nome do editor é concatenada dentro do documento MPEG-7

2.8 A informação de duração é concatenada dentro do documento MPEG-7

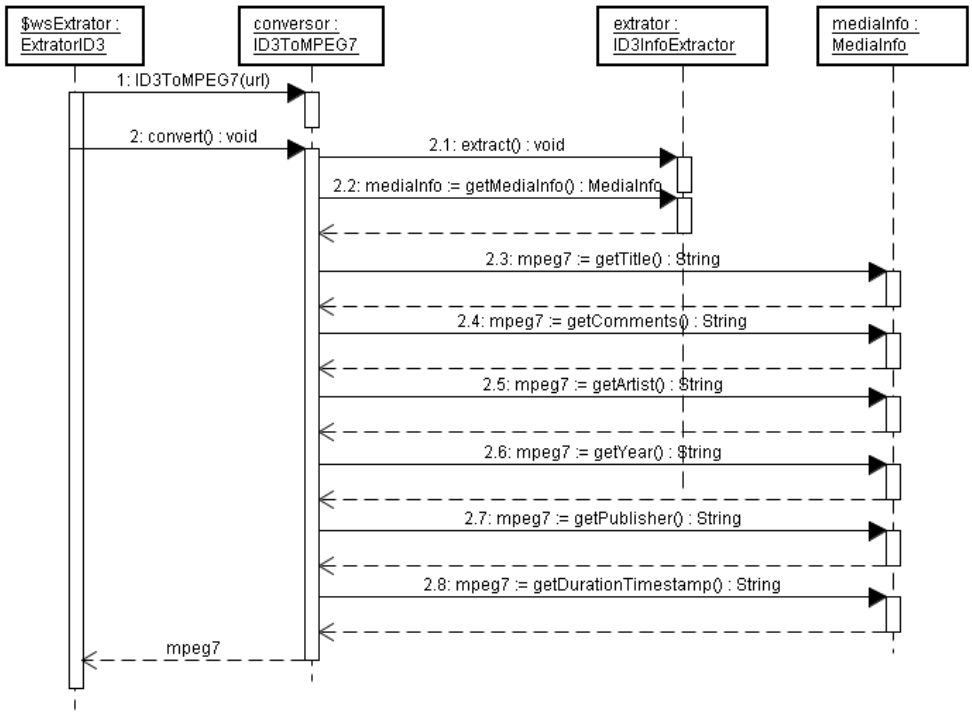

Figura 20 – Diagrama de seqüência da operação *extract()*
Fonte: Autoria própria, 2008.

4.2.7 Serviço de Filtro

O serviço de filtro recebe um documento ou fragmento MPEG-7 e retorna um ou mais fragmentos que estejam de acordo com o critério de filtro enviado pelo

usuário. Este serviço é útil para extrair elementos pertinentes a uma dada aplicação, facilitando as atividades de transformação ou processamento deste conteúdo.

O serviço de filtro possui duas operações: a operação padrão do modelo *help()* e a operação *filter().*

A operação *filter()* recebe dois parâmetros *_fragment*, que representa o fragmento ou documento de entrada que terá seus dados filtrados e o parâmetro *_xquery* que representa a expressão utilizada para filtrar os elementos do fragmento ou documento de entrada.

4.2.8 Implementação do Serviço de Filtro

Para demonstração do serviço de filtro foi construída uma única classe que executa uma expressão XQuery sobre os fragmento de entrada. Na figura 21 é exibido o diagrama de classes referente a implementação deste serviço. Esta implementação utiliza o banco de dados *eXist DB* como um meio para execução de expressões XQuery.

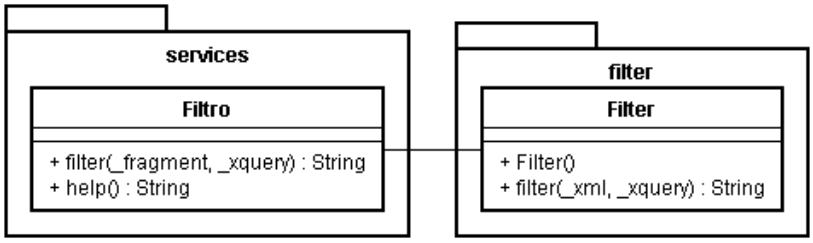

Figura 21 – Diagrama de classes da implementação do Serviço de Filtro
Fonte: Autoria própria, 2008.

Os fragmentos MPEG-7 são inseridos temporariamente na base de dados, após a execução do método de filtro o fragmento inserido é removido. É importante lembrar que os *namespaces* MPEG-7 devem ser informados no nodo raiz desse fragmento para que a consulta seja executada corretamente, caso contrário o banco de dados interpretará a entrada como um documento inválido.

O funcionamento do serviço é demonstrado pelo diagrama de seqüência da figura 22, que está acompanhado de sua explanação.

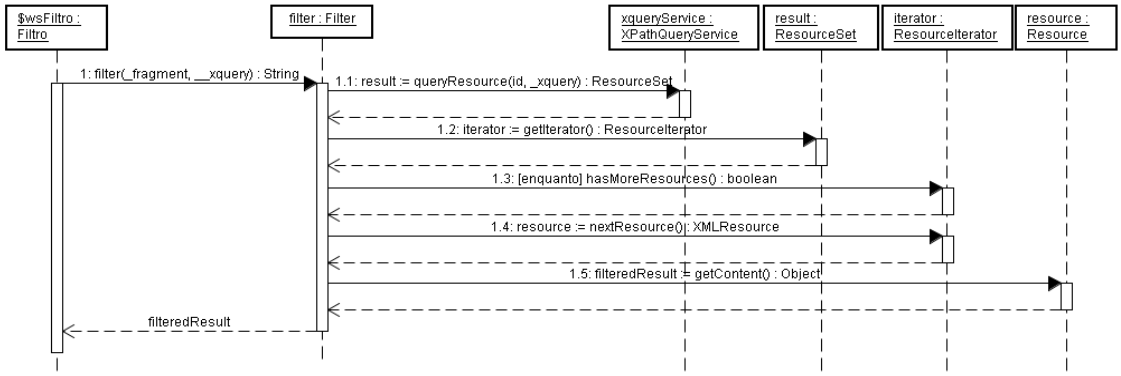

Figura 22 – Diagrama de seqüência da operação *filter()*
Fonte: Autoria própria, 2008.

1 O serviço invoca o método *filter()* do objeto *filter*.

1.1 Invoca o método *queryResource()*, que executa a expressão XQuery sobre o documento temporário criado. O parâmetro *id* representa a identificação do documento e que é retornada após a inserção do mesmo.

1.2 Recupera-se o iterador do resultado da consulta

1.3 Execução da iteração

1.4 Recupera-se o próximo resultado retornado pela expressão XQuery

1.5 Concatena o conteúdo de um recurso retornado como resultado ao restante do conteúdo já recuperado

4.2.9 Serviço Apresentador

A função dos serviços apresentadores é de transformar fragmentos ou documentos MPEG-7 em um conteúdo de outro formato. Para isto os apresentadores recebem como entrada um documento XML que possua um ou mais fragmentos MPEG-7*.*

Estes fragmentos são processados e transformados em um formato de conteúdo que facilite o consumo pelo usuário. Este conteúdo de saída pode ser um tipo dado intermediário (utilizado na execução de outras aplicações, como XAML), outro formato de metadado (como por exemplo, SCORM) ou um arquivo propriamente dito, como um *flash vídeo,* MP3*,* etc.

Também é possível existirem serviços de apresentação distintos, que recebam a mesma entrada de dados, mas que retornem saídas diferentes. Como por exemplo, uma página HTML ou uma apresentação SMIL, conforme apresentado no modelo da figura 23.

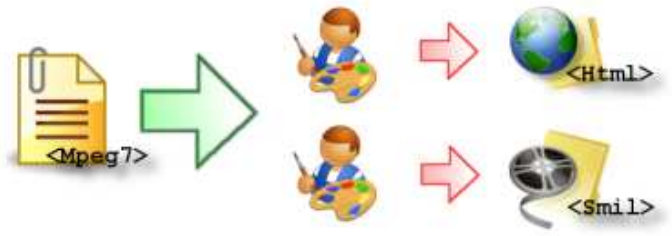

Figura 23 – Funcionamento do Serviço Extrator
Fonte: Autoria própria, 2008.

Um serviço de apresentação é composto por duas operações: *help()* e *present()*. A operação *help()*, operação padrão do ambiente, exibe um texto explicativo do uso do serviço. A operação *present()* recebe o parâmetro *_input* que possui um documento ou fragmento MPEG-7 que será processado. Como resultado, é retornado um valor no formato *string* que codifica o tipo de saída especificada pelo serviço.

4.2.10 Implementação do Serviço Apresentador

No desenvolvimento do trabalho foram criados dois serviços de apresentação que transformam fragmentos MPEG-7 distintos em uma saída no formato HTML. O primeiro serviço, chamado ApresentadorBuscaHTML transforma o resultado de um serviço de pesquisa em uma lista de itens em formato HTML. Esta lista possui ligações (*links*), que ligam a página de resultados com a apresentação do conteúdo. A implementação deste serviço restringe a listar itens do tipo *Summary.*

O outro serviço de apresentação, é chamado de ApresentadorSumarioHierarquico. Este serviço recebe como entrada um fragmento MPEG-7 do tipo *HierarchicalSummary* e retorna uma saída codificada em HTML que apresenta este sumário. Esta implementação está restrita a arquivos de áudio previamente segmentados (conforme a codificação MPEG-7 exibida no quadro 18), outra restrição é a necessidade da instalação do componente *Flowplayer*

(*http://flowplayer.org/player/index.html*) no lado do servidor, para a execução dos arquivos de áudio que fazem parte do sumário.

```
<SummarySegmentGroup level="1">
      <SummarySegment>
            <Name>Introdução</Name>
            <KeyAudioClip>
                  <MediaUri>
                  http://localhost/audios/pesquisa02.1.mp3
                  </MediaUri>
            </KeyAudioClip>
      </SummarySegment>
</SummarySegmentGroup>
<SummarySegmentGroup level="1">
      <SummarySegment>
            <Name>Fatores para Elaboração do Projeto</Name>
            <KeyAudioClip>
                  <MediaUri>
                  http://localhost/audios/pesquisa02.2.mp3
                  </MediaUri>
            </KeyAudioClip>
      </SummarySegment>
</SummarySegmentGroup>
```

Quadro 18 – Exemplo de segmentação utilizada pelo apresentador de sumários hierárquicos
Fonte: Autoria própria, 2008.

O diagrama de classes da figura 24, demonstra o relacionamento das classes que transformam a entrada MPEG-7 na saída codificada. Ambas as classes de apresentação (*HierarchicalSummaryHTMLPresenter* e *PresenterSearchHTML*) estendem a classe *Presenter*. Ambas as classes transformam o conteúdo de entrada em uma estrutura de árvore no formato DOM (*Document Object Model*) - a classe *Presenter* é responsável pela criação dessa estrutura. O método abstrato *createPresentation()* é implementado por suas subclasses para que o conteúdo de entrada seja transformado em uma codificação HTML.

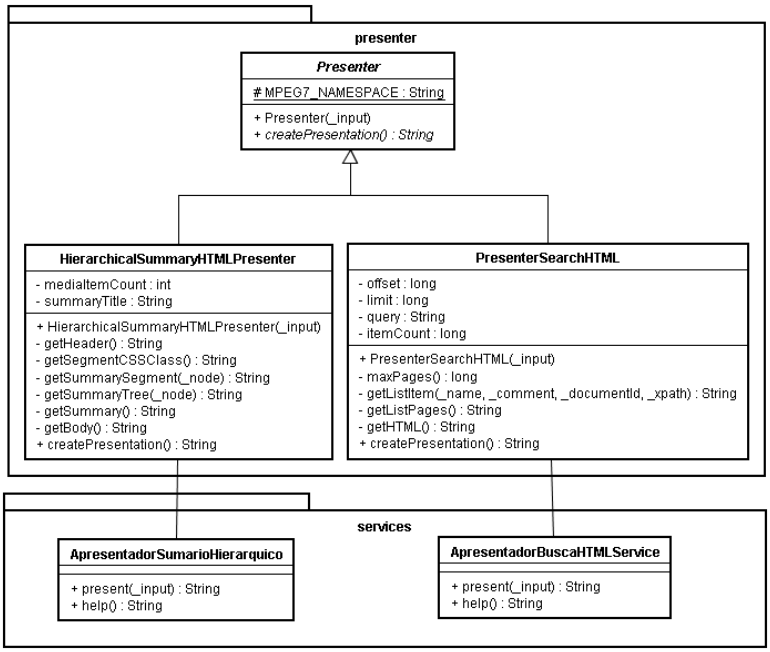

Figura 24 – Diagrama de classes dos serviços de apresentação
Fonte: Autoria própria, 2008.

De uma maneira geral o funcionamento dos serviços de apresentação implementados é o mesmo. Em ambos os casos, os serviços invocam o método *createPresentation()*, mas de objetos concretos diferentes. Um invoca o método de uma instância da classe HierarchicalSummaryHTMLPresenter e outro uma instância da classe PresenterSearchHTML. A figura 25 exibe os diagramas de seqüência de cada serviço.

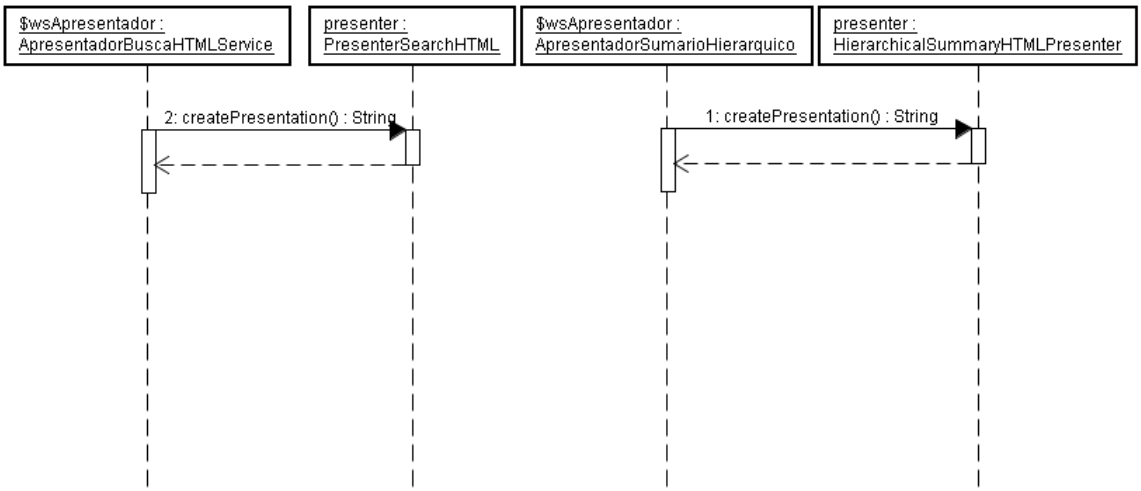

Figura 25 – Diagrama de seqüência dos serviços HierarchicalSummaryHTMLPresenter e PresenterSearchHTML
Fonte: Autoria própria, 2008.

4.3 Diretório de Serviços

Diretório de serviços ou serviço de registro, não chegaram a ser utilizados na implementação desse trabalho por razões já apresentadas.

Dentro desse componente seriam registrados e catalogados aqueles serviços que implementem os critérios definidos pelo modelo. A idéia básica do diretório de serviços é de registrar os serviços do modelo. Ao ser registrado, um serviço receberia uma identificação única desse diretório. Através dessa identificação seria possível que outras aplicações ou serviços invocarem o serviço independente dele ter sido trocado de localização ou fornecedor (Curbera et al., 2002).

Outra vantagem do uso de diretório serviços é a possibilidade de pesquisar por novos serviços através de categorias ou características de implementação (Curbera et al., 2002), (Endrei et al., 2004).

Embora não tenha sido utilizado neste trabalho, chegou-se a testar uma implementação de UDDI, como forma de solução de serviço de diretório, chamada *Apache jUDDI* (*http://ws.apache.org/juddi/*). *jUDDI* é uma implementação de UDDI versão 2.0, de código fonte aberto e que pode ser instalada dentro de um servidor de aplicação *Java*. Não chegou-se a registrar algum serviço do modelo nesse serviço, mas fica como registro a possibilidade de seu uso para este fim.

5 RESULTADOS OBTIDOS

Como forma de obter resultados concretos do uso do modelo desenvolvido, foi criada uma aplicação web para pesquisa e visualização de sumários hierárquicos.

Embora não seja a finalidade do trabalho, o qual se foca na construção ambiente de serviços para construção de aplicações que utilizam o padrão MPEG-7, foi executada medições de desempenho do serviço de extração e pesquisa. Ambas as medições de desempenho foram executadas para servir de valores de referências para trabalhos futuros, como também para identificar a necessidade de aperfeiçoamento de algum serviço.

A aplicação e as medições de desempenho dos serviços (incluindo uma análise dos seus resultados) serão descritas neste capítulo.

5.1 Pesquisador de Sumários Hierárquicos

O pesquisador de sumário hierárquico é uma aplicação que é executada a partir de um navegador de internet, semelhante às páginas de pesquisa existentes (*Yahoo*, *Google*, etc). A aplicação possui uma caixa de texto onde o usuário pode digitar o seu texto de pesquisa, um campo de escolha da opção de consulta (texto livre ou XQuery), e um botão para execução da consulta. Ao executar a consulta uma lista dos resultados é exibida, contendo um *link* em cada resultado para que o usuário utilize e visualize o sumário.

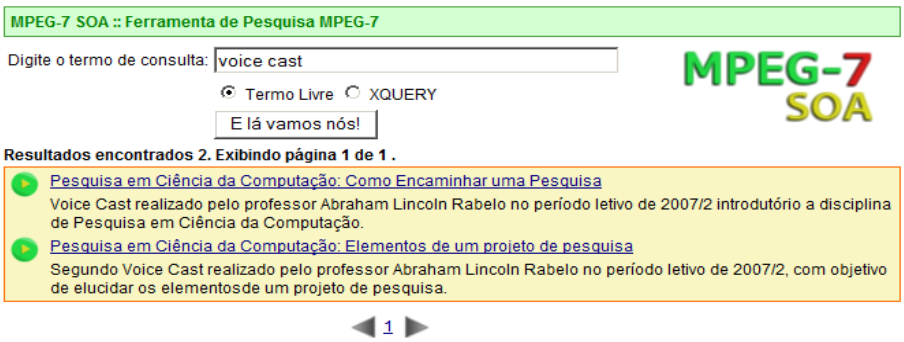

Figura 26 – Tela da aplicação de busca de sumários hierárquicos
Fonte: Autoria própria, 2008.

O funcionamento da aplicação quanto a execução dos serviços é feita desta maneira:

1. O usuário digita a sua consulta, que é enviada para o servidor
2. Um *script* PHP é executado no servidor, este por sua vez processa a requisição e envia a consulta para o serviço de pesquisa
3. O serviço pesquisador efetua a pesquisa e retorna o resultado ao *script* que está em execução no servidor
4. O *script* envia o resultado da pesquisa ao serviço ApresentadorBuscaHTML. O resultado da pesquisa é então processado e transformada em uma codificação HTML. Esta codificação é retornada para o *script*.
5. O *script* monta uma página contendo o resultado da pesquisa. Esta página é enviada para o usuário.

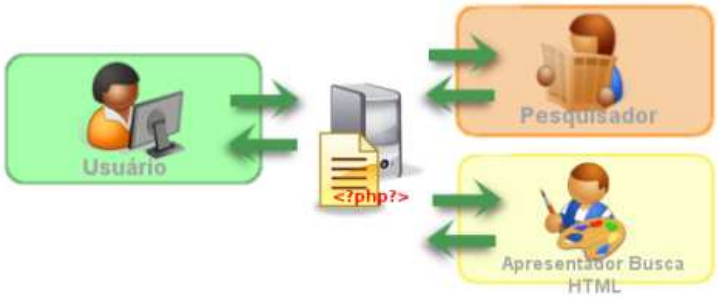

Figura 27 – Funcionamento da aplicação de busca de sumário hierárquico
Fonte: Autoria própria, 2008.

Quando o usuário “clica” em um *link* do resultado da pesquisa, é exibida uma página contendo uma apresentação daquele sumário que o usuário escolheu. O sumário é exibido de uma forma hierárquica, de forma que seja possível ao usuário navegar em sua árvore, escolher e executar aqueles itens do sumário que mais lhe interessam.

O funcionamento da apresentação do sumário quanto à execução dos serviços é feita da seguinte maneira:

1. O usuário “clica” em um item resultante da pesquisa. Ao clicar nesse item uma requisição é enviada ao servidor que executa um *script* PHP
2. A execução do *script* aciona o serviço de armazenamento, que recupera o fragmento do sumário que o usuário selecionou. Este fragmento é retornado ao *script* em execução
3. O *script* em execução aciona o serviço ApresentadorSumarioHierarquico. Este serviço recebe o fragmento de sumario e o transforma em uma codificação HTML
4. O servidor envia a apresentação no formato HTML do sumário ao usuário

Figura 28 – Tela de apresentação de um sumário hierárquico
Fonte: Autoria própria, 2008.

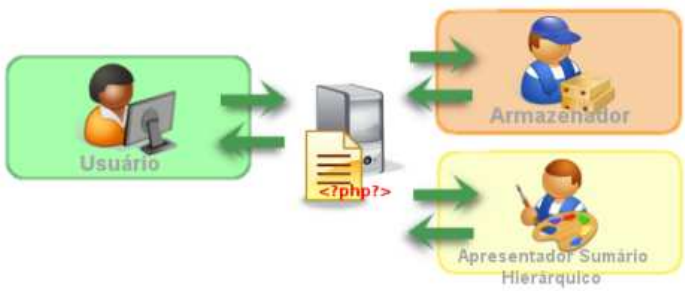

Figura 29 – Funcionamento da aplicação de sumários hierárquicos
Fonte: Autoria própria, 2008.

5.2 Medições de desempenho dos serviços de pesquisa e extração

Como forma de gerar valores de referências para trabalhos futuros sobre a situação de desempenho de alguns serviços do ambiente, foi medido (de forma simples) o desempenho do serviço de pesquisa e do serviço de extração. Ambos os testes foram feitos localmente, em um computador com as seguintes configurações: Processador *Intel Core 2 Duo T5500* (1.66 *GHz*), 1*GB Ram* e sistema operacional *Windows Vista*.

O serviço de extrator foi testado da seguinte maneira uma: uma lista de 70 arquivos MP3 foi disponibilizada em um diretório do sistema operacional. Um *script* PHP executou a operação *extract()* do serviço de extração para cada arquivo MP3 que encontrava-se no diretório. O tempo de execução da operação foi computado para cada arquivo. Como resultado tem-se a tabela exibida no apêndice A.

Não foi possível detectar uma relação clara entre o tempo de extração e duração da música. Por exemplo, a canção *Dance the Night Away,* que possui 216 segundos, foi extraída em 0,36 segundos, enquanto a canção *Sybil Green (Of the In Between),* que possui 163 segundos, teve um tempo de extração de 4,59 segundos. Este comportamento pode ter sido ocasionado por outro processo executado simultaneamente no sistema operacional, já que as mesmas canções em outro teste tiveram o tempo de extração de 2,47 segundos (*Dance the Night Away*) e 0,32 segundos (*Sybil Green (Of the In Between)*).

Dos trabalhos estudados, o artigo *Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching* disponibilizou seus resultados obtidos na extração de descritores MPEG-7 para a catalogação dos arquivos de áudio (Matushima et al., 2004). Embora existam resultados para esta operação, não é adequado comparar com este trabalho, pois o conteúdo extraído é diferente. No trabalho acima mencionado são extraídos descritores de baixo nível que necessitam analisar um trecho do arquivo de áudio, o que pode levar certo tempo. No presente trabalho foram extraídos dados do campo ID3 do arquivo de áudio. Estes dados não necessitam de análise apenas uma transformação para o formato MPEG-7.

Para o teste do serviço de pesquisa foram selecionadas 50 palavras (disponíveis nos apêndices B, C e D). Executou-se a operação *search()* utilizando o tipo de pesquisa “texto livre”, retornando elementos do tipo *AudioType* e sem limites

de resultado. Para cada palavra mediu-se o tempo despendido na pesquisa e a quantidade de resultados retornados. O tempo mínimo para um resultado foi de 0,42 segundos, e o tempo resultante para 20 resultados foi de 18 segundos.

Tabela 2 – Resultado da operação *search()* para as palavras *free* e *cream*

Palavra	**Resultados**	**Tempo (Segundos)**
free	1	0,42
cream	20	18

Fonte: Autoria própria, 2008.

Embora o desempenho do serviço de busca não seja o objetivo do trabalho, se comparado com os resultados de uma pesquisa do projeto MIRROR (Wong, Cheung e Po, 2005), 18 segundos é um tempo muito elevado. Isto se deve a duas operações que o serviço de pesquisa executa para cada resultado encontrado: Busca do caminho XPath do resultado e verificação do tipo de cada eixo do caminho XPath.

O algoritmo para busca do caminho XPath encontra-se no apêndice E. É um algoritmo que recursivamente vai buscando o nodo pai do elemento resultante, através de novas consultas XQuery. Por exemplo, se a consulta de entrada é "*//*:SummarySegment*", a próxima pesquisa será "*//*:SummarySegment/..*", depois "*//*:SummarySegment/../..*" e assim por diante. Como este algoritmo utiliza o método *query()* para executar a consulta, houve a necessidade de adicionar uma iteração do tipo *while* e a verificação do nome do documento antes de executar novamente o método. Para aperfeiçoar este resultado, poderiam ser utilizados o método *queryResource()* ou a função *util:node-xpath(node),* própria do banco eXistDB, que recupera o caminho XPath de um nodo.

A verificação de tipo é feita para cada eixo do caminho XPath resultante, de uma maneira *top-down* (de cima para baixo). O algoritmo considera que o primeiro eixo do caminho sempre será do tipo "Mpeg7". Em seguida o algoritmo irá verificar o tipo do próximo eixo à direita. Se o elemento possuir um atributo *type,* a pesquisa avança ao próximo eixo. Caso contrário, o algoritmo consulta os elementos permitidos pelo tipo do eixo à esquerda. Se o elemento for encontrado a pesquisa avança ao próximo eixo. Caso contrário, é feita uma pesquisa nos tipos superiores.

O serviço de busca foi testado sem a execução destas duas funcionalidades – Apêndice C e D. Sem a funcionalidade de verificação de tipo obteve-se os seguintes resultados para as palavras "*free*" e "*cream*", citadas no exemplo anterior:

Tabela 3 – Resultado da operação para as palavras free e cream sem a funcionalidade de verificação de tipo

Palavra	Resultados	Tempo (Segundos)
free	1	0,19
cream	20	9,19

Fonte: Autoria própria, 2008.

Neste caso, a quantidade de resultados pode aumentar para algumas palavras, pois sem a verificação de tipo alguns elementos são retornados mais de uma vez. Por exemplo, isoladamente ou dentro de outros elementos.

Sem a execução da funcionalidade de busca do caminho XPath, obteve-se o seguinte resultado para as mesmas palavras anteriores:

Tabela 4 - Resultado da operação *search* para as palavras *free* e *cream* sem a funcionalidade de busca do caminho XPath

Palavra	Resultados	Tempo (Segundos)
free	1	0,03
cream	20	0,54

Fonte: Autoria própria, 2008.

É extremamente perceptível o efeito que a execução destas operações possui sobre o tempo de resposta da operação de buscas. Sem a verificação de tipo e sem a busca do caminho XPath a operação de busca da palavra *cream* foi executada em 3% do seu tempo original. Embora o ganho de desempenho seja evidente, as duas funcionalidades são necessárias para as aplicações que utilizam os serviços do modelo.

A verificação de tipo é importante para que o serviço de pesquisa retorne em seus resultados apenas aqueles tipos que a aplicação tem interesse, enquanto que o caminho XPath é utilizado pelas aplicações que utilizam os serviços de armazenamento para buscar ou inserir informações de um arquivo de descrição MPEG-7.

5.3 Conclusão sobre os resultados

Embora o desempenho do serviço de pesquisa não tenha sido satisfatório, pode-se considerar que o objetivo do modelo foi alcançado. O foco do trabalho era o de criar um ambiente orientado a serviços que agregasse as operações comuns a aplicações MPEG-7 e que também pudesse facilitar a criação destas aplicações.

A aplicação descrita na seção 5.2 agrega três categorias de serviços (pesquisador, armazenador e apresentador). Para criar a aplicação não foi necessário escrever uma grande quantidade de linhas de código – 67 linhas para a página de pesquisa, contabilizando todo o código HTML e 16 linhas para a apresentação do sumário. Isto se deve ao fato do código ser baseado em chamadas aos serviços do modelo, beneficiando o desenvolvedor que terá mais trabalho escolhendo o serviço que mais está de acordo com as suas necessidades, do que com codificando a solução.

Os resultados das medições servem para que futuros trabalhos tenham uma referência do estado atual deste trabalho e possam dar prosseguimento no aperfeiçoamento daqueles pontos que ainda não atingiram sua plenitude.

6 CONCLUSÃO

O objetivo deste trabalho era de criar um modelo de ambiente que pudesse incorporar as operações comuns em aplicações que fazem uso do padrão MPEG-7, conforme abordado no capítulo 2. A tabela 5 recapitula a análise feita na seção 2.5, onde são comparados os trabalhos analisados pela implementação de cada uma das operações propostas. Nesta tabela é acrescentada uma comparação com o presente trabalho, que incorpora todas aquelas operações listadas.

Tabela 5 – Comparação entre os trabalhos estudados e o presente trabalho

	An Environment for Multimedia Indexing and Searching	**SOLO**	**MIRROR**	**Structuring Interactive TV Documents**	**Presente Trabalho**
Utiliza funções de anotação	NÃO	NÃO	NÃO	SIM	SIM
Utiliza funções de pesquisa	SIM	SIM	SIM	NÃO	SIM
Utiliza funções de extração de dados	SIM	NÃO	SIM	NÃO	SIM
Utiliza funções de apresentação	SIM	NÃO	SIM	SIM	SIM
Armazena novos documentos MPEG-7	SIM	NÃO	SIM	SIM	SIM
Acessa documentos externos	NÃO	NÃO	NÃO	SIM	SIM
Possui interface para acesso a seus serviços	SIM	SIM	NÃO	NÃO	SIM

Fonte: Autoria própria, 2008.

Todas as funções descritas estão incorporadas nos serviços do ambiente, o que possibilita que as aplicações mencionadas nos trabalhos estudados possam ser desenvolvidas utilizando o ambiente orientado a serviços descrito neste trabalho.

Outro benefício do modelo (abordado anteriormente na seção 5.3) é o tamanho dos códigos criados. No desenvolvimento dos aplicativos de pesquisa, apresentação de sumário, testes de desempenho do serviço de pesquisa e testes de desempenho do serviço de extração, pode-se observar que o tempo para o desenvolvimento destes aplicativos foi pequeno, como também a quantidade de código escrito.

Os programas gerados tiveram uma média de 60,25 linhas de código. Como são fontes de programas *web*, dentro dessa média inclui-se código HTML e também algumas classes e funcionalidades criadas rapidamente para implementação de alguma característica – como é o caso da classe *DirIterator* embutida do código fonte do teste de desempenho do serviço extrator.

Tabela 6 – Quantidade de linhas de código por *script*

Arquivo Fonte	**Função**	**Número de Linhas**
pesquisa.php	Página de pesquisa de sumários	67
show.php	Apresentador de sumário hierárquico	16
benchmark_extrator.php	Teste de desempenho do serviço extrator	102
benchmark_pesquisador.php	Teste de desempenho do serviço de pesquisa	56
	Média:	**60,25**

Fonte: Autoria própria, 2008.

Como o modelo incorpora as operações mais comuns em aplicações MPEG-7, fica fácil a construção de novos programas que utilizem este padrão.

De acordo com os resultados e conclusões demonstradas podem-se listar os seguintes candidatos a trabalhos futuros:

- Anotador de Arquivos Multimídia
- Aperfeiçoamento do Serviço de Pesquisa

6.1 Anotador de Arquivos Multimídia

A função do anotador seria a de criar as descrições de alto e baixo nível de arquivos multimídia. Por exemplo, um anotador de conteúdos de áudio poderia utilizar um serviço de extração para capturar o esquema de descrição de assinatura de áudio (*AudioSignature DS*). Este esquema de descrição funciona como uma identificação do arquivo de áudio e pode ser utilizada como elemento de comparação em consultas (Martínez, 2004).

Além disso, o anotador possibilitaria ao usuário, escutar o arquivo de áudio viabilizando a criação de sumários do conteúdo, definição de palavras chaves e comentários de um segmento. As descrições criadas seriam armazenadas através do serviço de armazenamento, e poderiam ser pesquisadas posteriormente pelo serviço de pesquisa.

6.2 Aperfeiçoamento do serviço de pesquisa

Embora o desempenho do serviço de pesquisa não seja o foco deste trabalho, observou-se através dos resultados da sua medição de desempenho a necessidade de aperfeiçoar a forma como a implementação do serviço recupera o caminho XPath dos elementos, como também a forma que a implementação verifica os tipos dos elementos MPEG-7. Embora retirando estas duas funcionalidades o serviço obtenha resultados muito satisfatórios, a ausência delas implica em duplicação dos resultados da pesquisa (alguns elementos são retornados mais de uma vez, por exemplo, às vezes como elementos raiz ou como filhos de outros elementos), como também sobrecarga do usuário do serviço, pois ficaria sob responsabilidade dele filtrar aqueles tipos que ele tem interesse.

6.3 Considerações finais

Como já foi abordado no início do capítulo, com o auxílio da composição dos serviços do ambiente, é possível criar as mais variadas aplicações multimídia que fazem uso do padrão MPEG-7. Além disso, como o ambiente foi criado através de uma arquitetura orientada a serviços fica fácil a adição e aquisição de serviços no ambiente, como também a criação de novas funcionalidades no modelo, como por

exemplo, um serviço de autenticação e autorização de acesso responsável restringir e autorizar o acesso de usuários as operações do ambiente.

REFERÊNCIAS BIBLIOGRÁFICAS

PAPAZOGLOU, Mike; GEORGAKOPOULOS. **Service Oriented Computing**. 2003. COMMUNICATIONS OF THE ACM Vol. 46, No. 10, p. 25-28

YANG, Jian. **Web Service Componentization**. 2003. COMMUNICATIONS OF THE ACM Vol. 46, No. 10, p. 35-40

CURBERA, Francisco; KHALAF, Rania; MUKHI, Nirmal; TAI, Stefan; WEERAWARNA, Sanjiva. **Web Service Componentization**. 2003. ACM Vol. 46, No. 10, p. 29-34

ELFATATRY, Ahmed. **DEALING WITH CHANGE: COMPONENTS VERSUS SERVICES**. 2007. COMMUNICATIONS OF THE ACM Vol. 50, No. 8, p. 35-39

MERSON, Paulo. **SOA O que se ganha e o que se perde.** 2006. Mundo Java, número 19, Ano IV . Editora Mundo.

NASCIMENTO, Harolfo. **Modelos de implementação para aplicações Web services clientes**. 2005. Mundo Java, número 11, Ano II. Editora Mundo.

ORT, Ed. **Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools**. 2005. Disponível em: <http://java.sun.com/developer/technicalArticles/WebServices/soa2/soa2.pdf>. Acesso em: out. 2007.

BARRY, Douglas. **Web Service and Service Oriented Architetures - The Savvy Manager's Guide**. 2003. Capítulos 1-6, 10 e 11. Editora Morgan Kaufmann.

CURBERA, Francisco; DUFTLER Matthew; KHALAF, Rania; NAGY, William; MUKHI, Nirmal; WEERAWARNA, Sanjiva. **Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI**. 2002. IEEE INTERNET COMPUTING – Março – Abril.

ENDREI, Mark; ANG, Jenny; ARSANJANI, Ali; CHUA, Sook; COMTE, Philippe; KROGDAHL, Pål; LUO, Min; NEWLING, Tony. **Patterns: Service-Oriented Architecture and Web Services**. 2004. IBM Redbooks. p. 18-20

LEWANDOWSKI, Scott. **Frameworks for Component-Based Client/Server Computing**. 1998. ACM Computing Surveys, Vol. 30, No 1.

ERL, Thomas. **Service-Oriented Architecture: Concepts, Technology, and Design**. 2005. Editora Prentice Hall.

YANG, Jian; PAPAZOGLOU, Mike. **Web Component: A Substrate for Web Service Reuse and Composition**. 2002.

HUNTER, Jane. **An Overview of the MPEG-7 Description Definition Language (DDL)**. 2001. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 11, NO. 6.

CHANG, Shih-Fu; SIKORA, Thomas; PURI, Atul. **Overview of the MPEG-7 Standard**. 2001. TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 11, NO. 6.

UDDI Version 2.04, 2002. **UDDI.org Working Group**. Disponível em: <http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>. Acessado em: out. 2007.

MARTÍNEZ, José. **Introducing MPEG-7 DDL – an Overview**. 2005. Disponível em: <http://www.chiariglione.org/mpeg/technologies/mp07-ddl/index.htm>. Acessado em: out. 2007.

HUNTER, Jane. **Working Draft ISO /IEC International Standard 15938-2 Information technology - Multimedia content description interface – Part 2 Description Definition Language**. 2000.

MARTÍNEZ, José. **MPEG-7 Overview**. 2004. Disponível em: <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm >. Acessado em: set. 2007.

REGO, Alex; BAPTISTA, Cláudio; DA SILVA, Elvis; SCHIEL, Ulrich; FIGUEIRÊDO, Hugo. **VideoLib: a Video Digital Library with Support to Spatial and Temporal Dimensions**. 2007. Proceedings of the 2007 ACM symposium on Applied computing, p. 1074 – 1078.

RYU, Jeewoong; SOHN, Yumi; KIM, Munchurl. **MPEG-7 Metadata Authoring Tool**. 2002. Proceedings of the tenth ACM international conference on Multimedia, p. 267 – 270.

PFEIFFER, Silvia; SRINIVASAN, Uma. **TV Anytime as an application scenario**. 2000. Proceedings of the 2000 ACM workshops on Multimedia, p. 89-92.

GOULARTE, Rudinei; MOREIRA, Edson; PIMENTEL, Maria. **Structuring Interactive TV Documents**. 2003. Proceedings of the 2003 ACM symposium on Document engineering, p. 42-51.

TRAN-THUONG, Tien; ROISIN, Cécile. **Multimedia Modeling Using MPEG-7 for Authoring Multimedia Integration.** 2003. Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval, p. 171-178.

AGIUS, Harry; AGELIDES, Marios. **MPEG-7 in Action: End User Experiences with COSMOS-7 Front End Systems**. 2006. Proceedings of the 2006 ACM symposium on Applied computing, p. 1348-1355.

XML Schema, 2001, **XML Schema**, Disponível em: <http://www.w3.org/XML/Schema>. Acessado em: out. 2007

LAY, Jose; GUAN, Ling. **SOLO: AN MPEG-7 OPTIMUM SEARCH TOOL**. 2000. ICME 2000 Volume 2 p. 777-780

WONG, Ka-Man; CHEUNG, Kwok-Wai; PO, Lai-Main. **MIRROR: An Interactive Content Based Image Retrieval System**. 2005. IEEE International Symposium on Circuits and Systems Vol. 2 p. 1541-1544

MATUSHIMA, Reinaldo; HIRAMATSU, Daniel; SILVEIRA, Regina; RUGGIERO, Wilson. **Integrating MPEG-7 Descriptors and Pattern recognition: An Environment for Multimedia Indexing and Searching**. 2004. WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia.

FATEMI, Nastaran; KHALED, Omar. **Indexing and Retrieval of TV News Programs Based on MPEG-7**. 2001. International Conference on Consumer Electronics, p. 360-361

SALEMBIER, Philippe; SMITH, John. **MPEG-7 Multimedia Description Schemes**. 2001. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 11, NO. 6, JUNE 2001, p. 748-759

MARTÍNEZ, Jose; KOENEN, Rob; PEREIRA, Fernando. **MPEG-7 The Generic Multimedia Content Description Standard, Part 1**. 2002. IEEE Multimedia Vol. 9 p. 78-87

XPath, 1999, **XML Path Language**. Disponível em: <http://www.w3.org/TR/xpath>. Acessado em: mar. 2008

XQuery, 2007, **XQuery 1.0: An XML Query Language**. Disponível em: <http://www.w3.org/TR/xquery/>. Acessado em: abr. 2008

ISO/IEC 15938-5, 2003, **Multimedia content description interface - Part 5: Multimedia description schemes**. Número de referência: ISO/IEC 15938-5:2003

XML:DB API, 2003, **XML:DB Initiative for XML Databases**. Disponível em: <http://xmldb-org.sourceforge.net/index.html>. Acessado em: abr. 2008

eXistDB, 2000, **Open Source Native XML Database**. Disponível em: <http://exist.sourceforge.net/>. Acessado em: mar. 2008

ID3, 1998, **ID3 tagging forma**t. Disponível em: <http://www.id3.org>. Acessado em: abr. 2008

MITROVIC, Dalibor; ZEPPELZAUER, Mathias; EIDENBERGER, Horst. **Analysis of the Data Quality of Audio Descriptions of Environmental Sounds**. 2006. Fourth Special Workshop Proceedings, p. 70-79

URI,1998, **Uniform Resource Identifiers (URI)**. Disponível em: <http://www.ietf.org/rfc/rfc2396.txt>. Acessado em: mai. 2008

APÊNDICE A – Resultado da medição de desempenho do serviço extrator

Autor	Título	Tamanho (MB)	Duração (Segundos)	Tempo de Extração (Segundos)
Blues Magoos	Albert Common Is Dead	2,65	111	0,2835
Blues Magoos	Baby, I Want You	3,87	164	1,8163
Blues Magoos	Chicken Wire Lady	5,76	247	2,3789
Blues Magoos	Gotta Get Away	3,89	165	1,6137
Blues Magoos	I Can Hear the Grass Grow	3,34	141	0,4219
Blues Magoos	I Can Move A Mountain	5,4	231	1,5657
Blues Magoos	I Wanna Be There	4,25	181	0,4183
Blues Magoos	Life Is Just Cher O' Bowlies	3,71	157	1,2801
Blues Magoos	Love Seems Doomed	4,29	183	2,0076
Blues Magoos	One By One	3,93	167	0,3129
Blues Magoos	Pipe Dream	3,48	147	0,7278
Blues Magoos	Queen Of My Night	4,36	186	0,9612
Blues Magoos	Rush Hour	3,71	157	1,8343
Blues Magoos	She's Coming Home	3,88	165	2,592
Blues Magoos	So I'm Wrong And You Are Right	3,48	148	0,378
Blues Magoos	Sometimes I Think About	5,91	254	0,5573
Blues Magoos	Summer Is the Man	4,25	181	0,8994
Blues Magoos	Sybil Green (Of the In Between)	3,84	163	4,5926
Blues Magoos	There's A Chance We Can Make It	3,21	136	0,2626
Blues Magoos	Tobacco Road	6,58	283	1,0098
Blues Magoos	(We Ain't Got) Nothing Yet	3,27	138	0,2817
Blues Magoos	Yellow Rose	3,59	152	0,4446
Cream	Blue Condition	4,83	210	0,6737
Cream	Dance the	4,44	216	0,3591

	Night Away			
Cream	Mother's Lament	1,65	107	0,5107
Cream/Cream	Outside Woman Blues	3,35	146	1,3792
Cream	Strange Brew	3,2	167	0,7809
Cream	Sunshine Of Your Love	3,87	253	1,3549
CREAM	Swlabr	2,35	153	0,2216
Cream	Take it Back	2,87	187	0,7278
Cream	Tales Of Brave Ulysses	2,57	168	0,504
Cream	We're Going Wrong	3,2	209	0,4206
Cream	World of Pain	4,21	183	0,3455
Cream	Cat's Squirrel	4,3	187	1,1385
Cream	Dreaming	1,85	121	0,2652
Cream	Four Until Late	2	130	0,7008
Cream	I Feel Free	2,66	173	0,6936
Cream	I'm So Glad	5,5	240	3,409
Cream	N,S,U,	1,91	166	0,5763
Cream	Rollin' and Tumblin'	6,5	283	0,5856
Cream	Sleepy Time Time	4,06	262	0,8007
Cream	Sweet Wine	3,06	200	0,2865
Cream	Toad	4,75	311	2,5833
Dinosaur Jr,	Feel the Pain	3,96	259	0,857
Dinosaur Jr,	I Don't Think So	3,31	216	1,0484
Dinosaur Jr,	Yeah Right	2,55	167	1,1803
Dinosaur Jr	Outta Hand	4,58	300	1,5499
Dinosaur Jr,	Grab It	3,26	212	1,1883
Dinosaur Jr	Even You	3,1	203	0,2651
Dinosaur Jr,	Mind Glow	4,65	243	1,9883
Dinosaur Jr,	Get Out of This	3,74	323	0,6625
Dinosaur Jr,	On the Brink	2,97	194	2,4113
Dinosaur Jr	Seemed Like The Thing To Do	5,32	348	0,4161
Dinosaur Jr	Over Your Shoulder	4,46	292	2,0239
Graforréia Xilarmônica	Eu	3,39	295	0,7184
Graforréia Xilarmônica	Eu Gostaria De matar os Dois	2,87	150	0,2412
Graforréia Xilarmônica	Iluminados Monstros do Amor	3,25	170	0,2571
Graforréia	Pensando Nela	3,49	182	0,2778

Xilarmônica				
Graforréia Xilarmônica	Benga Velha Companheira	2,68	117	0,6208
GraforréiaXilarmônica	Empregada	2,17	142	0,2295
Graforréia Xilarmônica	Hare	3,06	133	0,5775
Graforréia Xilarmônica	Minha Picardia	3,01	131	0,237
Graforréia Xilarmônica	Nunca diga	2,38	155	0,6325
Graforréia Xilarmônica	Patê	2,93	127	0,2749
Graforréia Xilarmônica	Se Arrependimento	4,12	179	0,3066
Graforréia Xilarmônica	Twist	2,21	96	0,1856
Graforréia Xilarmônica	Você foi embora	2,67	174	0,2363
Graforréia Xilarmônica	Denis	2,78	182	0,2063
Graforréia Xilarmônica	Hordas de demônio	2,3	150	0,1842
Graforréia Xilarmônica	O eunuco	2,35	153	1,8717
	Total:	**251,34**	**14094**	**65,6062**
	Média:	**3,590571429**	**201,3428571**	**0,937231429**

APÊNDICE B - Resultado da medição de desempenho do serviço de pesquisa

Palavra	Resultados	Tempo
about	1	1,19
amor	1	0,42
baby	1	0,6
back	1	0,57
blue	24	49,93
chance	1	0,42
coisa	0	0,03
cream	20	18
dance	1	0,43
dead	1	0,43
diga	1	0,42
dois	1	0,43
dream	1	0,42
embora	1	0,42
eu	2	0,89
feel	2	0,84
free	1	0,42
fresh	0	0,03
gear	0	0,04
green	1	0,43
hand	1	0,46
hear	1	0,43
hordas	1	0,44
just	1	0,44
lady	1	0,44
love	2	0,94
mind	1	0,49
mother	1	0,44
mountain	1	0,53
muito	0	0,02

night	2	1,39
nothing	1	0,62
nunca	1	0,43
one	1	0,62
out	1	0,43
pain	2	1,02
pensando	1	0,54
right	2	0,86
road	1	0,43
shoulder	1	0,46
strange	1	0,43
summer	1	0,42
sweet	1	0,43
there	1	0,44
thing	3	1,29
time	1	0,42
velha	1	0,42
woman	1	0,42
yellow	1	0,43
you	5	2,25
Total:	**100**	**94,74**
Média:	**2**	**1,8948**

APÊNDICE C- Resultado da medição de desempenho do serviço de pesquisa sem verificação de tipo

Palavra	Resultados	Tempo
about	1	0,19
amor	1	0,19
baby	1	0,21
back	1	0,2
blue	46	31,57
chance	1	0,21
coisa	0	0,02
cream	20	9,19
dance	1	0,23
dead	1	0,19
diga	1	0,19
dois	1	0,32
dream	1	0,2
embora	1	0,19
eu	2	0,36
feel	2	0,38
free	1	0,19
fresh	0	0,03
gear	0	0,02
green	1	0,19
hand	1	0,19
hear	1	0,19
hordas	1	0,19
just	1	0,2
lady	1	0,2
love	2	0,4
mind	1	0,19
mother	1	0,19
mountain	1	0,19

muito	0	0,03
night	2	0,38
nothing	1	0,22
nunca	1	0,19
one	1	0,19
out	1	0,19
pain	2	0,39
pensando	1	0,19
right	2	0,39
road	1	0,22
shoulder	1	0,43
strange	1	0,19
summer	1	0,19
sweet	1	0,19
there	1	0,22
thing	3	0,62
time	1	0,2
velha	1	0,19
woman	1	0,21
yellow	1	0,19
you	5	1,1
Total:	**122**	**52,33**
Média:	**2,44**	**1,0466**

APÊNDICE D - Resultado da medição de desempenho do serviço de pesquisa sem a funcionalidade de busca do caminho XPath

Palavra	Resultados	Tempo
about	1	1,33
amor	1	0,22
baby	1	0,06
back	1	0,03
blue	24	1,14
chance	1	0,28
coisa	0	0,03
cream	20	0,54
dance	1	0,03
dead	1	0,03
diga	1	0,13
dois	1	0,19
dream	1	0,04
embora	1	0,03
eu	2	0,23
feel	2	0,09
free	1	0,03
fresh	0	0,02
gear	0	0,02
green	1	0,03
hand	1	0,03
hear	1	0,03
hordas	1	0,03
just	1	0,03
lady	1	0,03
love	2	0,04
mind	1	0,03
mother	1	0,03
mountain	1	0,03

muito	0	0,16
night	2	0,04
nothing	1	0,03
nunca	1	0,03
one	1	0,03
out	1	0,03
pain	2	0,05
pensando	1	0,03
right	2	0,04
road	1	0,03
shoulder	1	0,17
strange	1	0,04
summer	1	0,03
sweet	1	0,1
there	1	0,03
thing	3	0,07
time	1	0,03
velha	1	0,04
woman	1	0,19
yellow	1	0,04
you	5	0,09
Total:	**100**	**6,08**
Média:	**2**	**0,1216**

APÊNDICE E – Código fonte do algoritmo para busca do caminho XPath

```
private String goRoot(String _query, String _document, String _id, String
_frag) {
  String xpath = "";
  try {
    ResourceSet result = xpserviceCollection.query(_query);
    if (result.getSize() > 0) {
      ResourceIterator iterator = result.getIterator();
      while (iterator.hasMoreResources()){
        XMLResource r =  (XMLResource) iterator.nextResource();
        if (!((String) r.getContent()).equals(_frag)
              && r.getDocumentId().equals(_document)
              && (r.getId().equals(_id)
                  || (r.getId().equals(_document)
                    && _id.split("\\_")(1).split("\\.").length == 1)
                )
          ) {
          return goRoot(_query + "/..", r.getDocumentId()
                ,this.parentId(r.getId())
                ,(String) r.getContent()) + "/*:" +
                 r.getContentAsDOM().getFirstChild().getNodeName();
        }
      }
    }
  } catch (XMLDBException ex) {
    Logger.getLogger(Search.class.getName()).log(Level.SEVERE, null, ex);
  } catch (Exception e) {
    System.out.println(e);
  }
  return xpath;
}
```

APÊNDICE F – Código fonte do algoritmo de verificação de tipo

```
private String() getTypes(String _collectionPath, String _xpath) {
  String() subPaths = this.createSubPaths(_xpath);
  String() types = new String(subPaths.length);
  types(0) = "Mpeg7";
  try {
    for (int i = 1;i < subPaths.length;i++) {
      String query = "doc('" + _collectionPath + "')" + subPaths(i);
      ResourceSet resultCollec = xpserviceCollection.query(query);
      Node node =
((XMLResource)resultCollec.getResource(0)).getContentAsDOM().getFirstChild(
);
      Node type = node.getAttributes().getNamedItem("xsi:type");
      if (type != null)
        types(i) = type.getFirstChild().getTextContent();
      else {
        boolean end = false;
        ResourceSet resultSchema = xpserviceSchema.query("//*(@name=\"" +
types(i-1)+
                                      "\")//*:element");
        ResourceIterator iterator = resultSchema.getIterator();
        while (!end) {
          if (!iterator.hasMoreResources()){
            resultSchema = xpserviceSchema.query("//*(@name=\"" + types(i-
1)+ "\")//*:extension");
            if (resultSchema.getResource(0)!= null) {
              XMLResource extensionSchema = (XMLResource)
resultSchema.getResource(0);
              Node nodeExtension =
extensionSchema.getContentAsDOM().getFirstChild();
              Node baseSchema =
nodeExtension.getAttributes().getNamedItem("base");
              String() tmpTypes =
baseSchema.getFirstChild().getTextContent().split("\\:");
              types(i-1) =  (tmpTypes.length > 1)?tmpTypes(1):tmpTypes(0);
              resultSchema = xpserviceSchema.query("//*(@name=\"" +
types(i-1)+ "\")//*:element");
              iterator = resultSchema.getIterator();
            } else
              end = true;
          } else {
            while(iterator.hasMoreResources()) {
              XMLResource resourceSchema = (XMLResource)
iterator.nextResource();
              Node nodeSchema =
resourceSchema.getContentAsDOM().getFirstChild();
              Node typeSchema =
nodeSchema.getAttributes().getNamedItem("type");
              Node elementSchema =
nodeSchema.getAttributes().getNamedItem("name");
              if
(elementSchema.getFirstChild().getTextContent().equals(node.getNodeName()))
{
                if (typeSchema != null) {
                  String() tmpTypes =
typeSchema.getFirstChild().getTextContent().split("\\:");
                  types(i) = (tmpTypes.length > 1)?tmpTypes(1):tmpTypes(0);
                } else
                   types(i) = "";
```

```
                end = true;
                break;
              }
            }
          }
        }
      }
    }
  } catch (XMLDBException ex) {
    Logger.getLogger(Search.class.getName()).log(Level.SEVERE, null, ex);
  } catch (Exception e) {
    System.out.println(e);
  }
  return types;
}
```