Faculdade de Engenharia da Universidade do Porto

DIP based MPEG-21 Player

Fernando André Gomes Silva

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientadores:
Prof.ª Dr.ª Maria Teresa Andrade (INESC Porto, FEUP)
Eng.º Pedro Miguel Carvalho (INESC Porto)

29/06/2009

Resumo

A norma MPEG-21 é uma norma que ainda se encontra em especificação e que tem como objectivo principal, permitir o acesso transparente a conteúdos multimédia, através de diferentes tipos de redes e dispositivos terminais, contribuindo assim para a implementação do Acesso Universal a Conteúdos Multimédia (UMA). A norma é baseada em dois conceitos essenciais: a definição da unidade fundamental de distribuição e transacção de informação multimédia, os *Digital Items*, e o conceito da interacção com os *Digital Items* por parte do utilizador. Actualmente existem poucas ferramentas capazes de interpretar *Digital Items* (DIs) de forma interoperável. Para além disso, o resultado dessa interpretação não permite extrair toda a essência do DI de forma transparente e simples para o utilizador final.

Para corresponder a estas dificuldades, os objectivos desta dissertação foi construir uma aplicação que processe *Digital Items* para mostrar o seu conteúdo, recorrendo à *Digital Item Processing* (DIP), a qual está especificada na parte 10 da norma MPEG-21, e também desenvolver alternativas que permitem mostrar o conteúdo do DI sem recurso à DIP, as quais fazem uso da criatividade do programador, respeitando contudo a estrutura atribuida pela norma para a especificação dos *Digital Items*. Após esta etapa mostrar que é possível visualizar o mesmo *Digital Item* (DI) de forma diferente, executando a mesma aplicação em duas máquinas diferentes. Numa primeira fase da dissertação é feito o levantamento do estado de arte, explicando todos os conceitos requeridos para perceber como se desenvolve uma aplicação deste tipo, e é descrito o funcionamento de algumas aplicações que se baseiam nesta norma; numa segunda fase é explicada de forma detalhada a capacidade da DIP para o processamento dos DIs, descrevendo a forma como ela interage com as diferentes partes da norma; numa terceira fase é descrita a aplicação desenvolvida, realçando a forma como ela utiliza a DIP para adicionar funcionalidades no que respeita à visualização do DI; e por fim, numa última fase é feita a conclusão de toda a dissertação e são dadas algumas sugestões de trabalho futuro para a DIP.

Abstract

The MPEG-21 standard is a standard that still meets under specification and that it has as main objective, to allow the transparent access to multimedia contents, through different types of nets and devices terminals, contributing like this to the implementation of the universal access to multimedia contents (UMA). The standard is based on two essential concepts: the definition of the fundamental unit of distribution and transaction of multimedia information, Digital Items, and the concept of the interaction with Digital Items on the part of the user. Actually exist few tools capable to interpret Digital Items (DIs) of interoperable form, as well as the fact of the result of that interpretation, don't get to extract all your essence in a transparent and simple way for the end user.

To correspond it these difficulties, the objective of this dissertation was to build an application that processes Digital Items to show your content, falling back upon the Digital Item Processing (DIP), which is specified in the part 10 of the MPEG-21 standard, and also to develop alternatives that allow to show the content of DI without resource to DIP, which make use of the programmer's creativity, respecting however the structure attributed by the standard for the specification of Digital Items. After this stage, to show that is possible to visualize the same Digital Item (DI) in a different way, executing the same application in two different machines. In a first phase of the dissertation it is made the rising of the art state, explaining all the concepts requested to understand as if develop an application of this type, and is described the functionalities of some applications that base on this standard; in a second phase it is explained the capacity of DIP in a detailed way for the processing of DIs, explaining like her interacts with the different parts of the standard; in a third phase the developed application will be explained, with emphasis for the form like her uses DIP to add functionalities in what it respects to the visualization of DI; and finally, in a last phase it is made the conclusion of all the dissertation and it is given some suggestions of future work for DIP.

Agradecimentos

Queria agradecer aos meus orientadores todas as suas sugestões, críticas e esclarecimentos de dúvidas que me forneceram ao longo da dissertação. E ao Eng.º Rui Filipe Santos Rocha pela sua disponibilidade em esclarecer dúvidas relacionadas com a sua dissertação de mestrado.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas

Lista de abreviaturas

API – *Application Program Interface*
DI – *Digital Item*
DIA – *Digital Item Adaptation*
DIBO – *Digital Item Base Operation*
DID – *Digital Item Declaration*
DIDL – *Digital Item Declaration Language*
DII – *Digital Item Identification*
DIM - *Digital Item Method*
DIME – *Digital Item Method Engine*
DIML - *Digital Item Method Language*
DIP – *Digital Item Processing*
DIXO - *Digital Item eXtension Operation*
DOM - *Document Object Model*
EC – *European Community*
ECMA – *European Computer Manufacturer Association*
ER – *Event Reporting*
FF –*File Format*
GUI – *Graphical User Interface*
HTML – *Hypertext Mark-up Language*
HTTP – *Hyper Text Transfer Protocol*
ID – *Identifier*
IEC – *International Electrotechnical Commission*
IEEE – *Institute of Electrical and Electronic Engineers*
IMS - *Integrated Management Supervisor*
IPMP – *Intellectual Property Management and Protection*
ISO – *International Organization for Standardization*

ITU – *International Telecommunication Union*

*M*IDI – *Musical Industry Digital Interface*

MPEG – *Moving Picture Experts Group*

PAT - *Evaluation Methods for Persistent Association Technologies*

PC – *Personal Computer*

PDA – *Personal Digital Assistant*

QoE – *Qualidade de Experiência*

QoS – *Quality of Service*

RDD – *Rights Data Dictionary*

RefSw – *Software* de referência

REL – *Rights Expression Language*

SMIL - *Synchronized Multimedia Integration Language*

SOAP - *Simple Object Access Protocol*

STP – *Step Top Box*

SVC - *Scalable Video Coding*

TR – *Technical Report*

TV – *Television*

UMA – *Universal Multiple Access*

URI - *Uniform Resource Identifier*

URL - *Uniform Resource Locator*

XML – *eXtensible Mark-up Language*

XSL - *eXtensible Stylesheet Language*

XSLT - *eXtensible Stylesheet Language Transformations*

Capítulo 1

Introdução

Hoje em dia as comunicações multimédia têm um peso enorme na nossa sociedade, existindo disponível no mercado um grande número de dispositivos terminais com capacidades variadas de apresentação de conteúdos multimédia. Em paralelo, o número de aplicações em que se combinam recursos como vídeo, texto, áudio e imagens cresce a um ritmo muito grande. Exemplos de dispositivos que têm sofrido grandes alterações evolutivas nos últimos anos são os telemóveis, em que assistimos constantemente à competição das operadoras de Telecomunicações pelos seus novos produtos tendo como pano de fundo não só o preço, mas essencialmente as características multimédia que estão associadas ao produto, nomeadamente no que diz respeito às aplicações multimédia que ele suporta.

Apesar de esta área continuar a avançar muito rapidamente, existem algumas limitações nesta grande área que é a multimédia, uma vez, que como todos sabem, alguns conteúdos são obtidos ilegalmente, como é o caso de músicas, vídeos, aplicações, etc., não respeitando os direitos de autor que lhe estão associados. Na maior parte dos casos, isto acontece porque as aplicações desenvolvidas para obter estes conteúdos não estão providas de mecanismos que permitam proteger esses direitos, de forma a impor regras a quem lhes acede. Outra das limitações existentes nesta área prende-se com o facto de existir problemas de compatibilidade entre diferentes dispositivos terminais para consumo de conteúdos multimédia. Face a estes problemas o grupo *Moving Experts Picture Group*, mais conhecido pela sua sigla, MPEG, está a desenvolver uma norma com o objectivo de colmatar este tipo de problemas, a norma MPEG-21.

A norma MPEG 21 tem como objectivos definir uma plataforma que suporte a transacção de conteúdos multimédia sob a forma de *Digital Items*, de uma forma interoperável e altamente automatizada, tendo em conta os requisitos de protecção de direitos desses conteúdos e sua distribuição usando uma larga gama de redes e terminais heterogéneos. A norma é baseada em dois conceitos fundamentais: os *Digital Items* (DIs) que são a unidade fundamental de distribuição e transacção de informação multimédia, e o conceito de interacção com os DIs por parte do utilizador. Face à norma ainda ser muito recente, existem poucas aplicações capazes de processar *Digital Items*, de modo a poder mostrar todo o seu

conteúdo de forma transparente e flexível para o utilizador final. Outros dos problemas destas aplicações é a questão da interoperabilidade em que duas aplicações distintas não conseguem abrir um mesmo *Digital Item* de modo a que possam visualizar o seu conteúdo. Isto deve-se ao facto de cada uma das aplicações ter definido as suas próprias regras no que respeita à interpretação do *Digital Item*, causando assim conflito.

O objectivo central desta dissertação consiste no desenvolvimento de uma aplicação para processar *Digital Items* de forma transparente e flexível para o utilizador. Tal deve ser alcançado recorrendo à especificação *Digital Item Processing* (DIP), a qual constitui a parte 10 da norma MPEG-21. Esta solução deverá resolver o problema da interoperabilidade mencionado acima, para além de oferecer formas alternativas à DIP para processar e apresentar os DIs, sempre com base nas regras especificadas pela norma MPEG-21. Por fim foi também proposto mostrar que é possível visualizar o mesmo DI de forma diferente, em duas máquinas que executam a mesma aplicação, consoante a pilha DIP que cada uma delas tem instalado. De salientar que a aplicação tem como cenário de aplicação o Ensino à Distância, e daí designá-la de MPEG21 DIP Teach.

Esta dissertação está divida em 9 capítulos. Os primeiros 3 capítulos englobam a explicação dos objectivos desta dissertação, o levantamento do estado de arte da norma e as aplicações MPEG-21 desenvolvidas ou que ainda estão em fase de desenvolvimento. O capítulo 4 é dedicado à DIP, explicando o seu conceito e as suas capacidades em termos daquilo que ela é capaz de fazer perante um DI. No capítulo 5 é descrita a aplicação desenvolvida, bem como questões de implementação. No capítulo 6 são apresentados testes da aplicação na forma como consome os *Digital Items*. No capítulo 7 é explicado um tipo particular de operações oferecidas pela DIP e que foram desenvolvidas nesta tese. No capítulo 8 são explicadas as outras interfaces desenvolvidas, no âmbito do cenário da aplicação. Por fim, no capítulo 9 é apresentada uma conclusão da dissertação, bem como sugestões de trabalho futuros para a DIP.

Capítulo 2

Norma MPEG-21

A ISO*(International Organization for Standardization)* e IEC*(International Electrotechnical Commission)* são duas organizações, que a par com outras organizações governamentais e não governamentais, trabalham em conjunto no desenvolvimento de normas internacionais, através de comités técnicos estabelecidos pelas mesmas. Através do grupo técnico SC29-WG11, conhecido sob a designação de MPEG *(Moving Pictures Expert Group)*, este organismo desenvolve especificações na área das aplicações multimédia. A norma MPEG-21 é um exemplo de uma especificação em desenvolvimento no seio deste grupo. MPEG-21 está dividido em várias partes, dentro das quais se destaca a parte 2 *(Digital Item Declaration)* e a parte 10 *(Digital Item Processing*) como relevantes no desenvolvimento da dissertação.

A norma MPEG-21 [14] tem como objectivos definir uma plataforma que suporte a transacção de conteúdos multimédia sob a forma de *Digital Items*, de uma forma interoperável e altamente automatizada, tendo em conta os requisitos de protecção de direitos desses conteúdos e sua distribuição usando uma larga gama de redes e dispositivos terminais heterogéneos. A norma é baseada em dois conceitos essenciais: os *Digital Items* e o conceito da interacção com os *Digital Items* por parte do utilizador.

2.1 - Estrutura da norma MPEG-21

A tecnologia multimédia permite aos criadores e consumidores, codificar, distribuir e aceder remotamente a conteúdos multimédia. A norma MPEG-21 pode ser vista como um conjunto de recomendações sobre a forma de utilizar ferramentas multimédia para a transacção de *Digital Items* entre diferentes comunidades e através de redes heterogéneas, de uma forma transparente. Para isso a norma é dividida em diferentes partes para facilitar o desenvolvimento de diferentes aplicações com diferentes requisitos e objectivos. Actualmente as 19 partes da norma MPEG-21 são (extraído de [5] e [44]):

- Parte 1: *Vision, Technology and Strategy*
- Parte 2: *Digital Item Declaration* (DID)

- Parte 3: *Digital Item Identification* (DII)
- Parte 4: *Intellectual Property Management and Protection Components* (IPMP)
- Parte 5: *Rights Expression Language* (REL)
- Parte 6: *Rights Data Dictionary* (RDD)
- Parte 7: *Digital Item Adaptation* (DIA)
- Parte 8: *Reference Software* (RefSw)
- Parte 9: *File Format* (FF)
- Parte 10: *Digital Item Processing* (DIP)
- Parte 11: *Evaluation Methods for Persistent Association Technologies* (PAT)
- Parte 12: *Test Bed for MPEG-21 Resource Delivery*
- Parte 13: *Scalable Video Coding* (SVC)
- Parte 14: *Conformance Testing*
- Parte 15: *Event Reporting* (ER)
- Parte 16: *Binary Format*
- Parte 17: *Fragment Identification of MPEG Resources*
- Parte 18: *Digital Item Streaming*
- Parte 19: *Media Value Chain Ontology*

Nesta dissertação será analisada a parte 2 *(Digital Item Processing)* e a parte 10 *(Digital Item Processing)*, uma vez que são as partes mais relevantes no desenvolvimento da aplicação, que faz parte dos objectivos desta dissertação. Se bem que ao longo desta dissertação as outras partes não são descuradas, uma vez que a DIP está relacionada de certa forma com elas.

2.2 - Digital Items e Utilizadores

Um *Digital Item* (DI) é definido como sendo a unidade fundamental de distribuição e transacção. É um objecto digital que se baseia numa combinação de recursos *media*, metadados e estrutura. Os recursos *media* constituem o próprio conteúdo a transaccionar, sendo os metadados a informação sobre os conteúdos ou recursos contidos no DI, e a estrutura é relação entre os recursos e os metadados. Os DIs são descritos na *Digital Item Declaration* (DID), parte 2 da norma que será explicada mais à frente neste capítulo.

Um exemplo de um DI pode ser uma compilação de música, incluindo a música mas também fotos, vídeos, animações gráficas, letras de músicas, ficheiros MIDI, entrevistas com os artistas, noticias relacionadas com os artistas, entre muitas outras coisas como ilustra a figura 2.1.

Figura 2.1 – Exemplo da estrutura de um *Digital Item* com conteúdos em forma de diagrama.

Na norma MPEG-21 existem vários tipos de utilizadores: consumidores, comunidades, organizações, corporações e até fornecedores de conteúdo. Contudo um utilizador pode assumir direitos específicos e responsabilidades de acordo com a interacção dele com outros utilizadores.

2.3 - Digital Item Declaration

Digital Item Declaration (DID) consiste num documento que descreve a estrutura de um *Digital Item* e as relações entre os seus componentes. É expresso através da linguagem *Digital Item Declaration Language* (DIDL), especificada na parte 2 [1] da norma MPEG-21 e é baseada em XML [49]. A DID estabelece um modelo uniforme e flexível, e uma representação de um esquema interoperável para definir os *Digital Items* numa forma estática.

A DID é estruturada através de entidades, para entender as entidades que estão referenciadas na declaração DID e a forma como elas estão relacionadas umas com as outras, é usado um modelo gráfico baseado em blocos para representação da DID de uma forma mais simples do que a declaração DID baseada na linguagem XML [49], como ilustra a figura 2.4 e 2.3 respectivamente. Cada uma das entidades vai ser descrita a seguir e o modo como elas se relaciona para melhor percepção da estrutura interna da declaração DID e do modelo gráfico usado como representação da declaração DID. No entanto, no capítulo 4 serão ilustrados exemplos de aplicação destas entidades com realce para o relacionamento entre elas.

Container – Um *Container* (contentor) é uma estrutura que permite agrupar *Items* e/ou *Containers*. Os *Containers* são identificados através de *Descriptors* (descritores).

Item – Um *Item* é um grupo de *sub-Items* ou *Components* (componentes) e também podem conter *Descriptors*, os quais contém informações sobre o *Item*.

Component – Um *Component* pode conter um ou mais *Resources* (recursos) e *Descriptors*. Os *Descriptors* têm informações sobre os *Resources* tais como *bit rate*, conjunto de caracteres, informações de encriptação, mas não informação descrevendo o conteúdo.

Anchor – Uma *Anchor* (Ancora) liga *Descriptors* a um *Fragment* (fragmento), que corresponde para uma específica localização ou uma gama dentro de um *Resource*.

Descriptor – Um *Descriptor* contem informação dentro de um elemento fechado *(*exemplo:*Containers*, *Items*, *Components)*. Esta informação pode ser um *Component* (tais como um *thumbnail* de uma imagem, ou um *Component* de texto), ou uma declaração textual.

Condition – Uma *Condition* (condição) descreve o elemento fechado como sendo opcional, e liga isto à *Selection* (selecção) ou *Selections* (selecções) que afectam a sua inclusão.

Choice – Uma *Choice* (escolha) descreve um conjunto de *Selections* relacionadas que podem afectar a configuração de um *Item.*

Selection – Uma *Selection* descreve uma decisão específica que pode afectar uma ou mais *Conditions* algures dentro do *Item*.

Annotation – Uma *Annotation* (anotação) descreve um conjunto de informações sobre outro elemento identificado no modelo sem alterar ou adicionar esse elemento. A informação pode tomar, a forma de *Assertions* (afirmações), *Descriptors* ou *Anchors*.

Assertion – Uma *Assertion* define um estado configurado de uma *Choice* completa ou parcial afirmando se ela é *true*, *false* ou *undecided* para algum número de *Predicates* associados com as *Selections* para essa *Choice.*

Resource – O *Resource* é um *Asset* (conteúdo com direitos) identificado individualmente tais com vídeo ou clipe de áudio, uma imagem, ou um *Asset* textual. Um *Resource* pode também ser potencialmente um objecto físico.

Fragment – Um *Fragment* é uma designação ambígua de um ponto específico ou uma gama dentro de um *Resource*. *Fragment* pode ser um tipo específico de *Resource*.

Statement – Um *Statement* (declaração) é um valor textual literal que contém informação, mas não é um *Asset*.

Predicate – Um *Predicate* é uma ambígua identificação de declaração que pode ser *true*, *false* ou *undecided*.

Como podemos observar na figura 2.2 existe um relacionamento entre as diversas entidades descritas anteriormente, as quais o criador da DID deve respeitar, isto é, se a

estrutura da DID não estiver segundo as regras da norma (não validação dos *Schemas*) pode não ser possivel visualizar o *Digital Item*, correctamente.

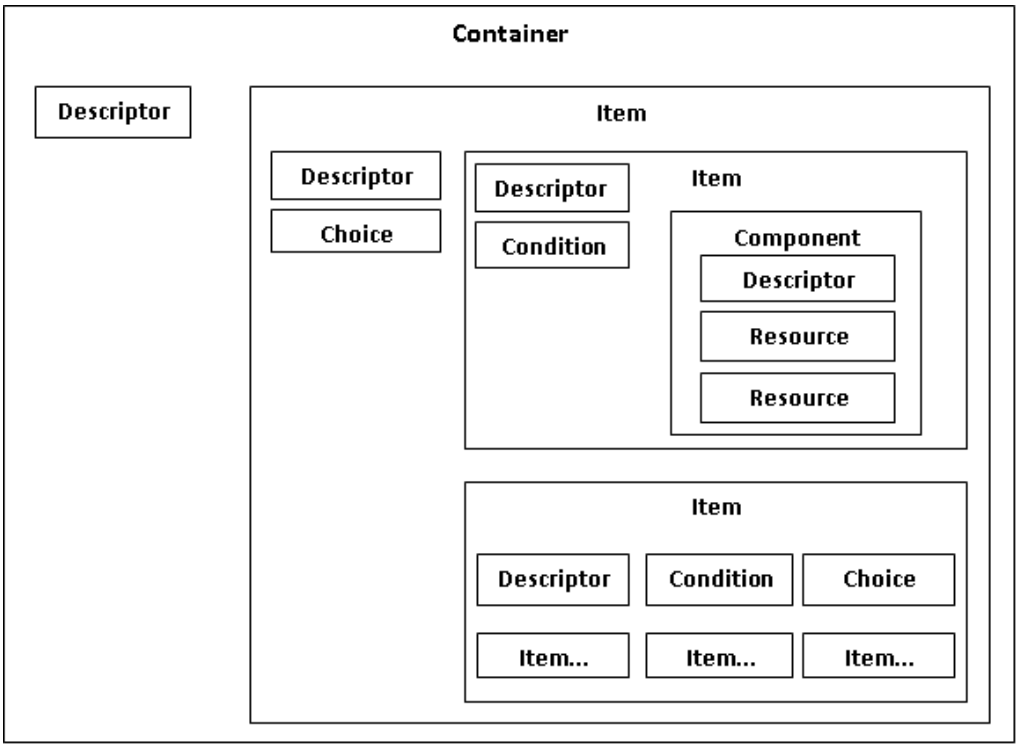


Figura 2.2 – Exemplo da relação das diferentes entidades com base numa representação gráfica da DID.

Exemplo de uma declaração de uma DID (representação XML) está representado na figura 2.3 usando recursos e metadados, e em que podemos visualizar cinco entidades, a *Descriptor*, a *Component*, a *Item*, a *Resource* e a *Container*. Enquanto na figura 2.4 podemos ver o modelo gráfico para representar as entidades da declaração DID mencionada na figura 2.3. Na representação gráfica da esquerda da figura 2.4, temos o modelo gráfico para as entidades que constituem a estrutura da DID da figura 2.3, enquanto na representação gráfica da direita da figura 2.4, temos a substituição dessas entidades pela respectiva informação que lhe está associada de acordo com a DID da figura 2.3.

```
<? Xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02:DIDL-NS">
        <Container>
                <Descriptor>
                        <Statement mimeType="text/plain">
                        As duas partes mais importantes desta dissertação, Digital Item Declaration e
Digital Item Processing.
                        </Statement>
                </Descriptor>
                <Item>
                        <Component>
                                <Resource mimeType="application/pdf" ref="did.pdf"/>
                        </Component>
                </Item>
                <Item>
                        <Component>
                                <Resource mimeType=" application/pdf" ref="dip.pdf"/>
                        </Component>
                </Item>
        </Container>
</DIDL>
```

Figura 2.3 – Exemplo de uma declaração de documentos da norma MPEG-21 usando a DID (representação XML).

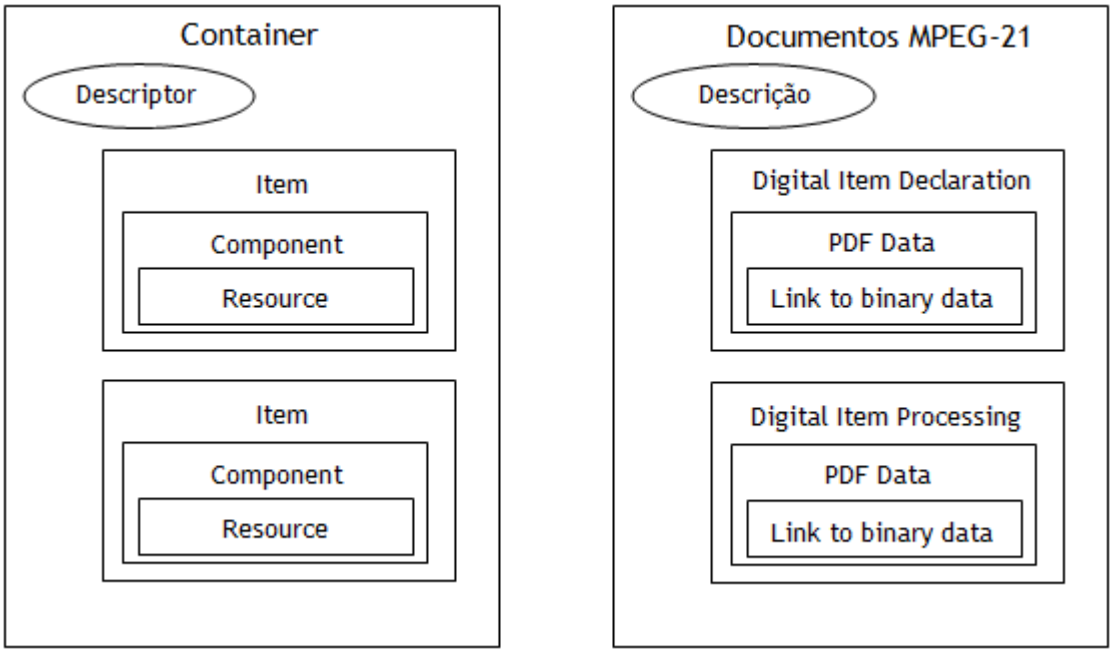


Figura 2.4 – Uma declaração de documentos da norma MPEG-21 usando a DID (representação gráfica).

2.4 - Digital Item Processing

Digital Item Processing (DIP) corresponde à parte 10 [2] da norma MPEG-21. Do ponto de vista da interoperabilidade e do consumo controlado de recursos *media*, esta parte é uma das mais importantes porque permite ao utilizador interagir com o DI, através das sugestões de interacção *(Digital Item Methods)* que o autor do DI fornece. Exemplos dessas interacções podem ser adicionar ou remover músicas de um álbum de música digital, pedir adaptação para os recursos, configurar os DIs de acordo com as preferências do utilizador. Em sistemas que utilizem DIP, é inserida na DID na altura de criação de cada DI, a indicação de um conjunto de operações ou métodos que pode ser aplicado ao DI. Essa lista de métodos, designada de *Digital Item Methods* (DIMs), pode ser apresentada ao utilizador na altura em que recebe o DI. O utilizador pode então escolher um método, o qual é executado automaticamente por um motor *software* designado de DIP Engine. As DIMs são descritas através de uma linguagem baseada em ECMAScript [31], e que no contexto da norma MPEG-21 é designada de *Digital Item Method Language* (DIML).

Quando um utilizador interage com um DI, existindo na respectiva DID uma lista de DIMs, o utilizador poderá escolher uma delas para ser executada. Sendo assim um DI poderá ter várias DIMs, sendo que cada uma delas poderá ter ou não argumentos associados a elas. Um aspecto importante, é o facto de poderem existir DIMs que estão definidas para arranque automático, ou seja, sempre que é feito o carregamento do DI, se existirem essas DIMs automáticas, então elas são executadas e resultado do processamento poderá ser uma interacção com o utilizador. De salientar o facto de existirem DIMs automáticas que são executadas quando do carregamento do DI, não implica que elas possam ser executadas de novo por o utilizador quando bem pretender.

As DIMs podem conter chamadas para *Digital Item Base Operations* (DIBOs) [2] e *Digital Item eXtension Operations* (DIXOs) [2], que são *Application Program Interfaces* (APIs) disponibilizadas pela norma MPEG-21, e que podem contribuir para permitir a interacção do utilizador com o DI. A principal diferença entre as DIBOs e DIXOs, reside no facto de que as DIBOs descrevem operações a efectuar sobre o DI e que estão normalizadas, e que por isso se aplicam de uma forma genérica a qualquer aplicação, ou seja, analisando a sua vertente prática, existe uma interface já criada para elas, a qual não pode ser modificada, podendo no entanto ser alterada a sua implementação. Pelo contrário, as DIXOs, não estão especificadas na norma, e descrevem operações específicas a cada aplicação, ou seja, analisando a sua vertente prática, o programador tem mais liberdade para programar em termos de definir a sua interface e a sua implementação, devendo no entanto seguir algumas restrições impostas pela norma.

Algumas DIBOs existentes actualmente na norma MPEG-21 e que são analisadas em detalhe no capitulo 4, recorrendo a exemplos (extraído de [2]):

DIA *(Digital Item Adaptation)* – *adapt*;

DID – *areConditionsSatisfied*, *configureChoice*, *setSelection*;

DII *(Digital Item Identification)* – *getElementsByIdentifier*, *getElementsByRelatedIdentifier*, *getElementsByType*;

DIP – *alert*, *execute*, *getExternalData*, *getObjectMap*, *getObjects*, *getValues*, *play*, *print*, *release*, *runDIM*, *wait*;

REL *(Rights Expression Language)* – *getLicense*, *queryLicenseAuthorization*;

DIP Error – *getDIPErrorCode*;

ObjectMap – *getArgumentList*, *getArgumentListCount*, *getMethodCount*, *getMethodWhithArgs*, *getObjectOfType*, *getObjectsOfType*, *getObjectsOfTypeCount*, *getObjectTypeCount*, *getObjectTypeName*;

PlaySatus – *getStatus*;

Outro aspecto não menos importante a ter em conta nesta parte da norma é o *ObjectMap* [2]. A DIP especifica a forma como DIMs e DIXOs são declarados na DIDL. Isto inclui como um elemento DIDL pode ser associado com um tipo de objecto *(Object Type)* e como os argumentos para uma DIM podem ser associados com um tipo de argumento. Ligando tipos de objectos e de argumentos fornecemos um mapa entre os elementos da DIDL e os argumentos da DIM. Este mapa é chamado de *Object Map*.

Mais concretamente um *Object Type* é um tipo de argumento passado para a DIM. Isto permite que o argumento seja processado dentro da DIM, de acordo com a sua semântica. Um *Object Map* fornece a relação entre os elementos da DID para os *Object Types*. Isto permite eles serem usados como um argumento de uma DIM.

O *Object Type* e o *Object Map* foram criados para permitir elementos da DID interagir com os argumentos da DIMs. A informação do *Object Type* é usada pelo DIP *Engine* para criar o *Object Map*. O DIP *Engine* é a unidade de processamento responsável para gerar um *Object Map*, e executar as DIMs.

A figura 2.5 ilustra a aplicação destes conceitos. Um *Object Map* pode ser comparado a uma tabela ligando o tipo de semântica a um conjunto de elementos DID. Antes de executar *PlayAdaptedMovie* da DIM incluído na DID, o *Object Map* pode ser usado para criar a lista de argumentos para esta DIM. Como só existe só um elemento DID de *Object Type* "urn:foo:movie", o único argumento válido para esta DIM é o primeiro *Component* da DID. Quando este *Component* é seleccionado da DID, usando o *Object Map* como mecanismo de procura, é convertido em *Object* DIM e feita acessibilidade em ambiente ECMAScript.

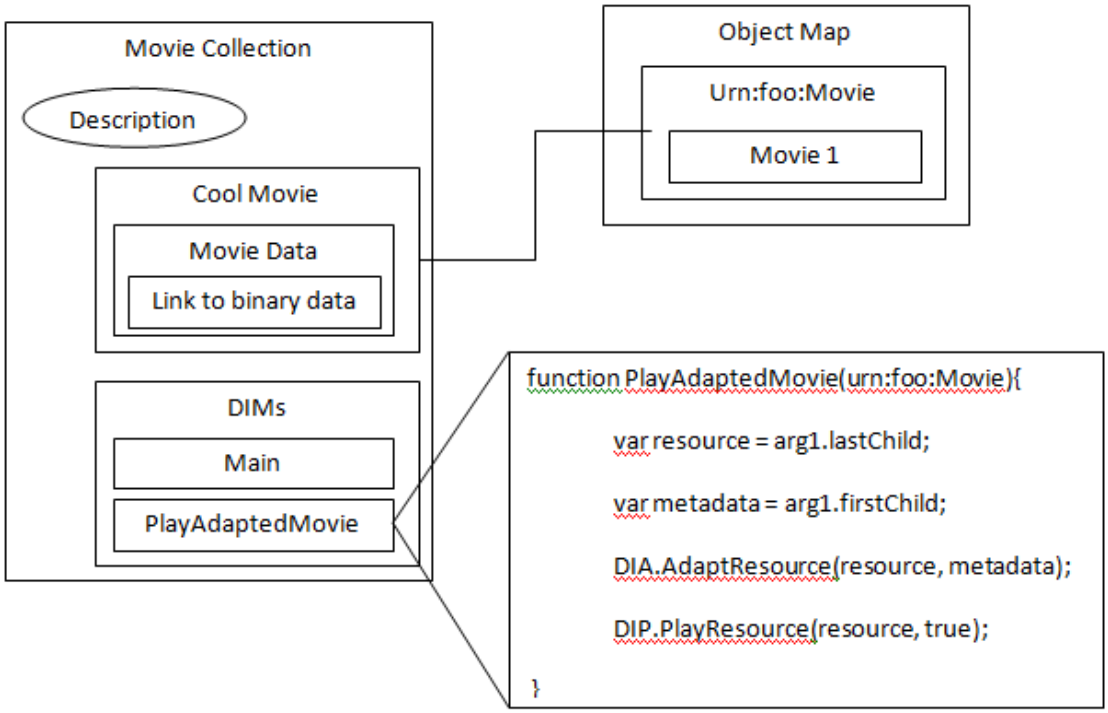


Figura 2.5 - Relação entre a DID, *Object Map* e a DIP[10].

2.5 - Conclusão

Um dos principais objectivos da norma é manipular os DIs de forma eficiente, transparente e de modo interoperável através das tecnologias adequadas para o seu acesso, consumo ou até mesmo comércio. O MPEG-21 visa a utilização de recursos multimédia de forma transparente e crescente, através de uma larga gama de redes e dispositivos, usados por diferentes comunidades. Para manipular estes DIs a parte 10 da norma (DIP), disponibiliza um conjunto de métodos designados por DIMs que permitem a interacção do utilizador com DI. Desta forma permitem estender a definição estática do DI fornecendo mecanismos que permitem manipular o DI de uma forma dinâmica. Esses métodos são declarados juntamente com os recursos e metadados na estrutura DID.

No próximo capítulo serão descritas aplicações MPEG-21 desenvolvidas por várias entidades, e que põe em prática algumas dos conceitos que foram referidas ao longo deste capítulo.

Capítulo 3

Aplicações MPEG-21

Como foi visto no capítulo anterior, a norma é baseada em dois conceitos essenciais, a definição da unidade fundamental de distribuição e transacção *(Digital Item),* e o conceito de interactividade com o *Digital Item*. *Digital Item* pode ser por exemplo uma colecção de vídeo, um álbum de música ou outros tipos de conteúdos multimédia. Sendo este um dos principais objectivos da norma: transaccionar os DIs de forma eficiente, transparente e de modo interoperável através das tecnologias adequadas.

Este capítulo aborda aplicações MPEG-21 do estado da arte desenvolvidas por diversas entidades. Algumas aplicações foram ou estão a ser desenvolvidas, mas existe pouca informação acerca delas, nomeadamente quanto às suas funcionalidades, daí que a essas apenas seja feita uma breve referência.

3.1 – INESC Porto

O INESC Porto é uma instituição que desenvolve trabalhos em diversas áreas, sendo a multimédia uma delas. A norma MPEG-21 é uma das normas que é alvo de investigação e desenvolvimento por parte desta instituição. Duas aplicações MPEG-21 desenvolvidas pelo INESC Porto são descritas de seguida:

DDIBrowser:

O DDIBrowser [5] é uma aplicação destinada especialmente para a *Web*, apresentando uma arquitectura baseada em serviços *Web* com as seguintes características:

- O Processamento de *Digital Items* é realizado no lado do servidor, sendo a apresentação na aplicação cliente;
- A troca de mensagens entre o cliente e o servidor são ficheiros baseados na linguagem XML (DIDs);
- É possível configurar a arquitectura do DDI Browser.

A aplicação apresenta várias funcionalidades dentro das quais se destacam as seguintes:

- Permite o download de *Digital Items*, bem como a sua validação;
- *Navegação* de conteúdos remotos de DIs em estilo de página *Web*;
- Apresentação de *Assets* (conteúdos com direitos) contidos nos DIs;
- Processamento das preferências e selecções do utilizador;

Serviços *Web* no servidor:

- Aplicação IDIP *server* é a unidade central. Esta fornece à aplicação processamento específico e controlo da navegação;
- Implementa um mecanismo de cache para *backward* dos DIs;
- Estrutura Modular;
- API dos serviços *Web* desenvolvida;

Cliente *Web*:

É chamado de WDI Browser, usa XSLT [50] para geração de apresentações HTML dos DIs, se bem que podem ser mapeados para outros formatos; apresenta uma estrutura modular; permite fácil adição de novas interfaces *Web* para diferentes terminais, e tem a vantagem de ser acessível através dos *Web Browsers comuns*.

Object Model:

Um modelo de objectos foi criado para aumentar a performance da aplicação e que pode ser reutilizado em outras aplicações MPEG-21.

Na figura 3.1 podemos ver arquitectura do DDIBrowser e relação entre os diferentes módulos.

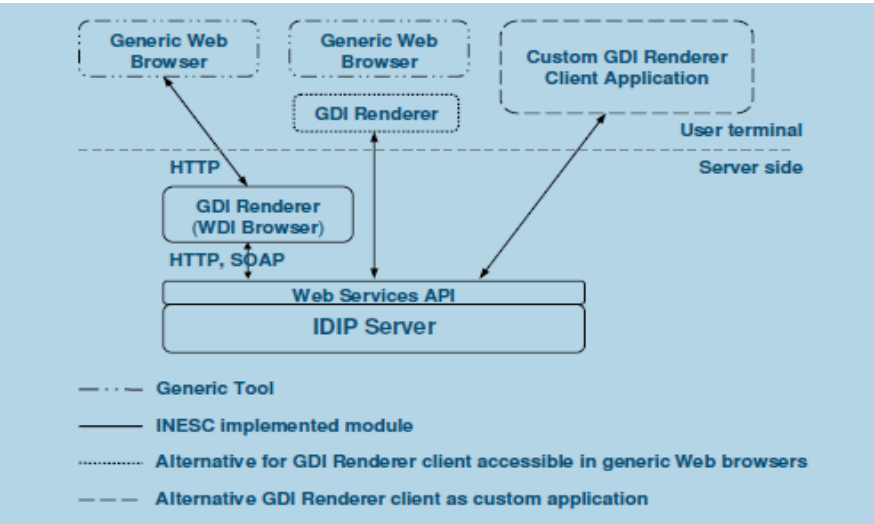


Figura 3.1 - Arquitectura do DDIBrowser [11].

O terminal cliente ilustrado na figura 3.2 é chamado de WDI Browser. O princípio de funcionamento da *framework* começa com o pedido do utilizador para acesso ao DI; após isto ele pode visualizar o conteúdo do elemento de topo do DI, devido ao facto de os DIs serem processados e visualizados de maneira progressiva, com a navegação nos DIs feita do topo do elemento da DIDL para baixo, ou seja, o utilizador pode navegar no DI carregando nos elementos que pretende aceder, de acordo com sua a estrutura hierárquica.

Um exemplo de estrutura hierárquica de um DI está representado na figura 3.3. Observando a estrutura hierárquica e seguindo a filosofia de navegação desta aplicação, se um utilizador aceder ao elemento *Container*, é visualizada e apresentada uma página *Web* ao utilizador com a informação do elemento, e são listados no ***Content Overview*** do WDI Browser os filhos do elemento *Container*, que o utilizador escolheu, que no caso da figura 3.3, lista os elementos: *Item 1*; *Item 2* e *Item 3*. Se o utilizador pretender aceder a um desses *Items*, o procedimento é o mesmo, e assim sucessivamente. De salientar que o utilizador pode retroceder para elementos anteriores, da mesma forma que retrocede quando navega em páginas *Web*. Por fim, os *Resources* do DI podem ser vistos no centro dessa interface, no ***Resource Viewer***; os *Descriptors* no painel ***Descriptions*** e as DIMs no painel ***DIMs***.

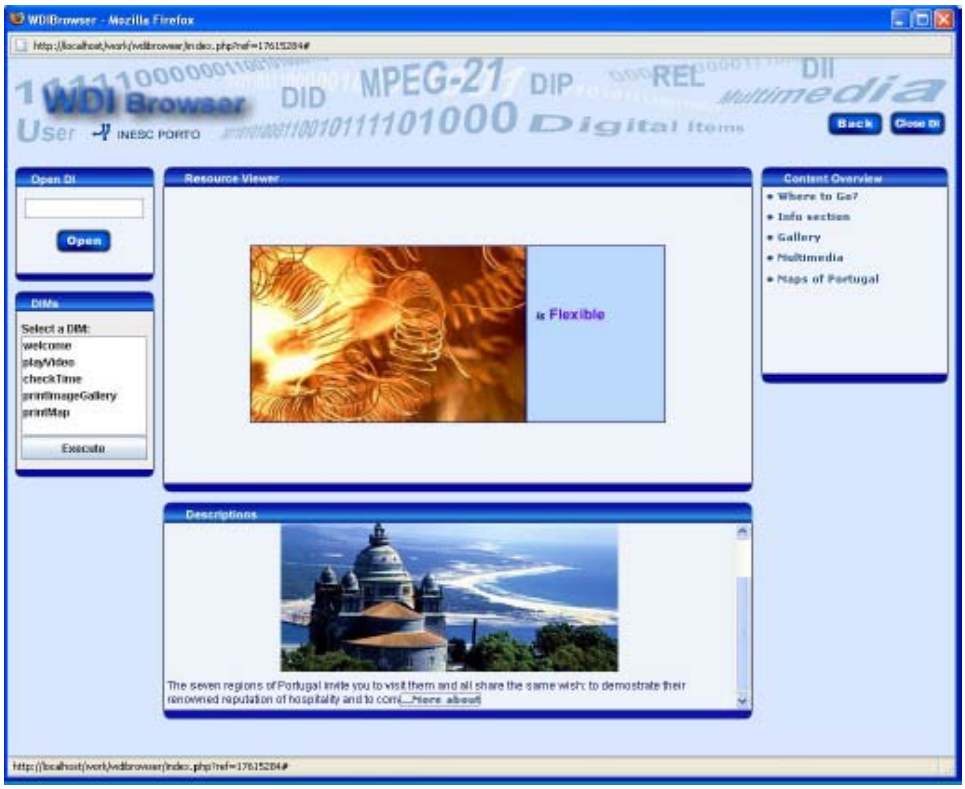


Figura 3.2 - Interface da *framework,* WDI Browser[11].

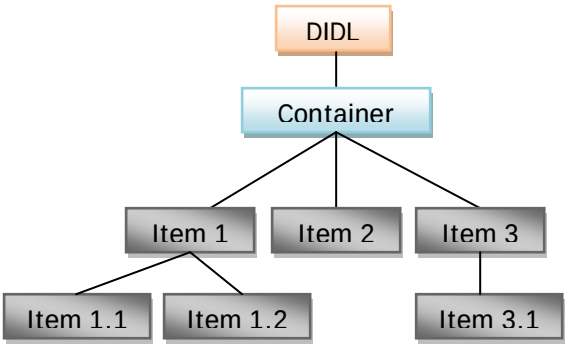


Figura 3.3 – Exemplo de estrutura hierárquica de um *Digital Item*.

Limitações:

Esta aplicação tem algumas falhas no que respeita à questão de interoperabilidade, uma vez que os DIs criados para esta aplicação foram testados para vereficar se eram interpretados na aplicação do AXMEDIS (descrita na secção 3.2) e vice-versa. O que aconteceu foi que os DIs desta aplicação não eram interpretados na aplicação do AXMEDIS, nem os DIs do AXMEDIS eram interpretados por o DDIBrowser. O que leva a concluir que estas aplicações definiram as suas próprias regras na definição dos DIs, gerando assim conflitos.

DIP:

A DIP [4] é uma aplicação que apresenta uma interface desenvolvida para software de referencia da norma MPEG-21, visto que ele não dispõe de uma, e que tem a particularidade de filtrar as DIMs com base nos *Object Types*. A figura 3.4 ilustra a interface da aplicação.

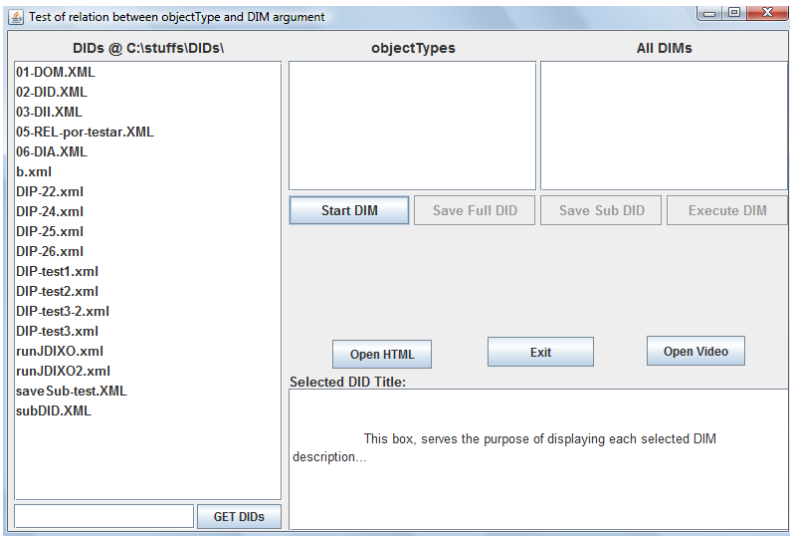


Figura 3.4 – Interface da aplicação DIP.

Ao arrancar a aplicação, ela acede a um directório específico e lista todos os ficheiros XML contidos nesse directório na sua interface, como podemos ver no lado esquerdo da interface da figura 3.4. Debaixo da lista encontra-se um botão (***GET DIDs***) que não funciona

na versão testada. No lado direito inferior encontrámos um painel que serve para apresentar informação sobre cada DID seleccionada, mais concretamente daquilo que ela trata (um *Descriptor*).

Quando seleccionamos uma DID é mostrado uma janela *popup* que diz o que a DID contém em forma textual, sendo que essa informação fica também no painel direito inferior como mostra a figura 3.5, mantendo o resto da interface como estava inicialmente à excepção do botão ***Save Full DID*** que fica activo.

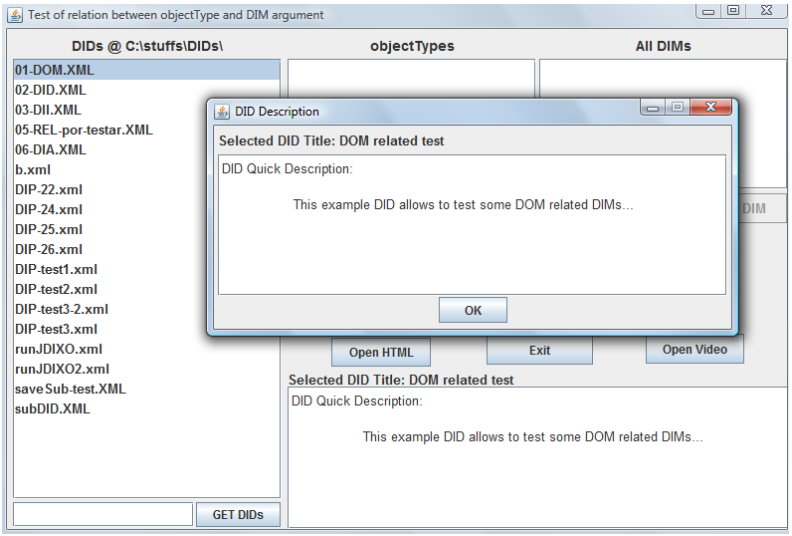


Figura 3.5 – Figura da aplicação DIP que mostra o que acontece quando selecciona-se uma DID.

Para testarmos a DIP na DID que seleccionamos, carregamos no botão ***Start DIM*** que vai procurar na DID seleccionada, todos os *Object Types* e DIMs que estão presentes nela, listando-os na respectiva parte correspondente da interface, e também activa o botão ***Save Sub DID***, como ilustra a figura 3.6. Para executar uma DIM, selecciona-se a DIM desejada e carrega-se no botão ***Execute DIM***, o qual fica activo após a selecção. Alternativamente podemos seleccionar um *Object Type* da lista, o que faz com que liste todas as DIMs que tem esse tipo de *Object Type* como tipo de Argumento, mantendo-se o mesmo procedimento para execução de uma DIM.

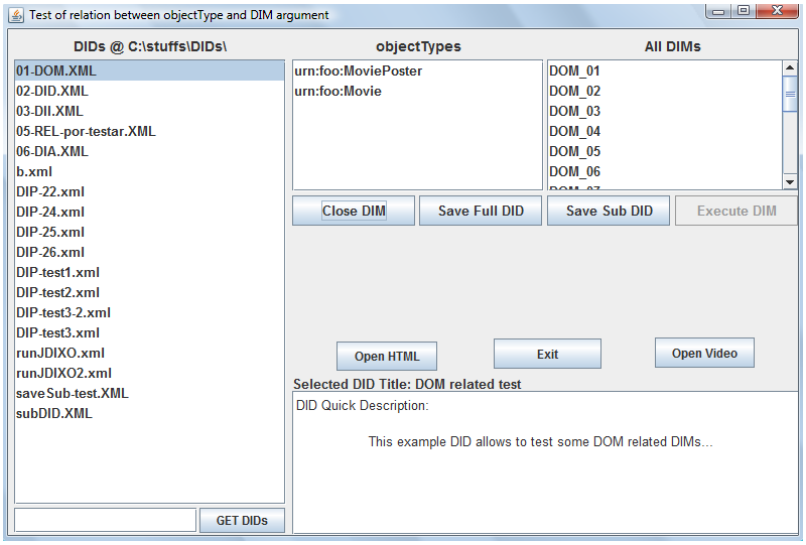


Figura 3.6 – Listagem de todos os *Object Types* e DIMs de uma DID.

Por fim para concluir as funcionalidades desta interface o botão ***Save Full DID*** permite guardar a DID num directório do computador escolhido pelo utilizador. O botão ***Exit*** permite sair da aplicação. O botão ***Save Sub DID*** permite guardar uma parte da DID com base num *Item* escolhido pelo utilizador através do id associado. O botão ***Open Vídeo*** e ***Open HTML*** não estão a funcionar, nem fazem parte da versão final desta interface.

Limitações:

- Não valida DIs;
- Não tem em conta os *mimeTypes* na reprodução de recursos, reproduzindo todos os recursos num *player* de áudio e vídeo;
- Aplicação pouco transparente a nível de utilizador, porque o utilizador não tem noção da forma como estão estruturados os *Digital Items*;
- Não reproduz DIMs definidas como arranque automático;
- DIs restringidos a um directório específico;

3.2 - AXMEDIS

Uma das aplicações mais populares baseadas na norma MPEG21 é o AXMEDIS [17]. É um projecto fundado pela Comunidade Europeia e que teve início em Setembro de 2004. Apresenta várias funcionalidades dentro das quais se destaca: ver os recursos existentes dentro de um objecto, visualizar os metadados do objecto, visualizar o SMIL [51], HTML, e/ou apresentação de construção MPEG-4. Uma das aplicações do AXMEDIS está baseada em páginas HTML optimizada para o *Mozila Firefox* e para o *Internet Explorer*, como podemos observar na figura 3.7.

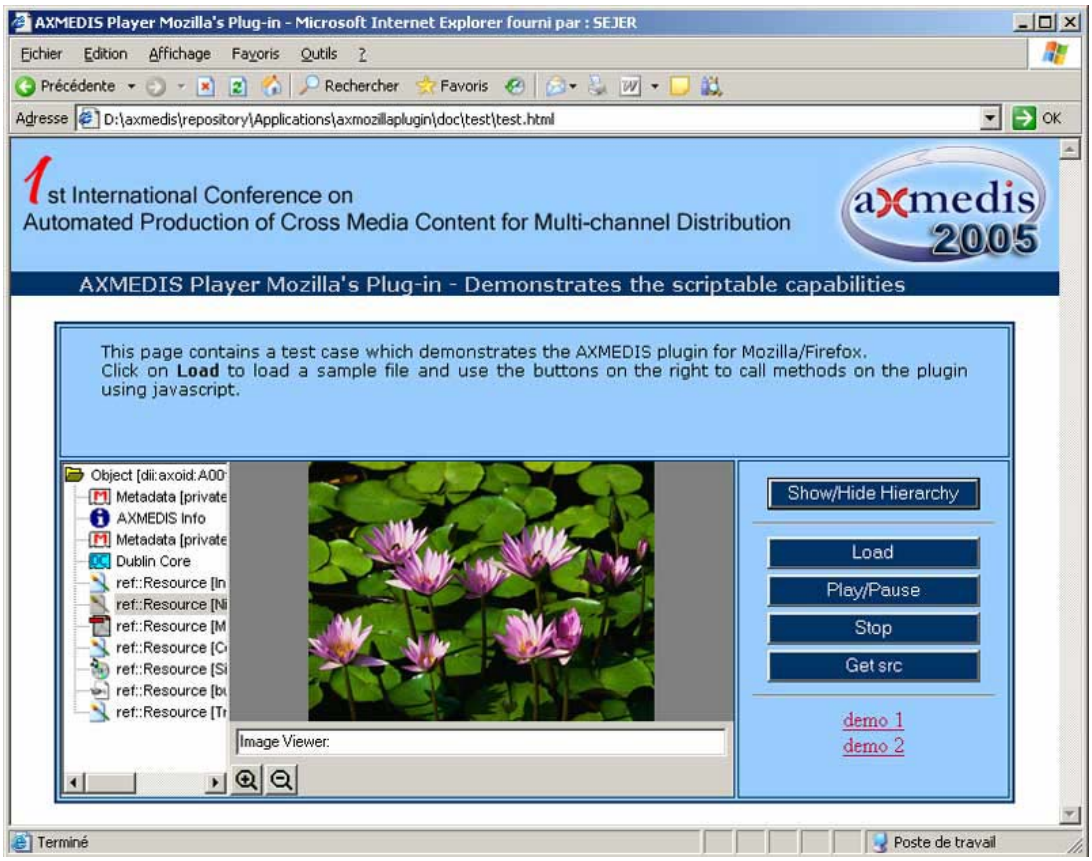


Figura 3.7 – Interface do Axmedis Player baseada em páginas HTML [18].

Outra das aplicações deste grupo é o AXMEDIS PLAYER Sejer, Bordas and Nathan, que se baseia nos formatos HTML, SMIL e MPEG4, cujo objectivo é melhorar o layout da aplicação. Um exemplo de um *screenshot* da aplicação está representado na figura 3.8, com diferentes tipos de recursos, tais como: imagens, vídeos, etc.

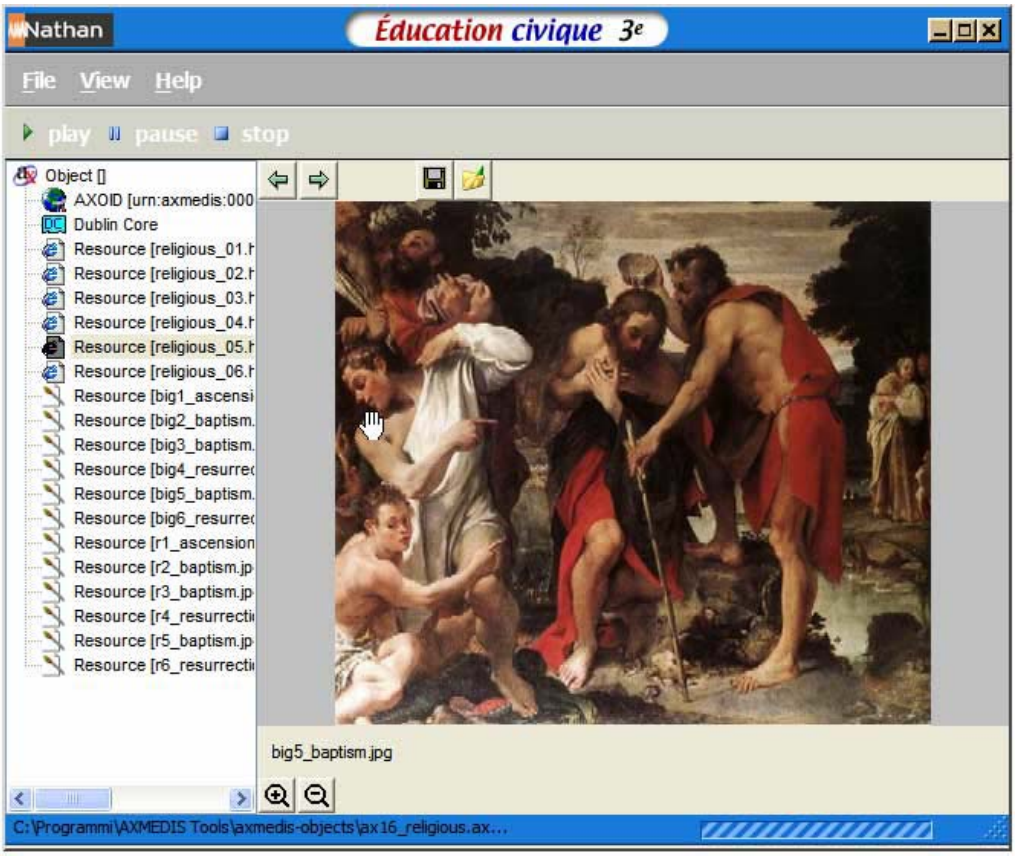


Figura 3.8 – Interface do AXMEDIS Player Sejer, Bordas and Nathan [18].

Neste momento o grupo AXMEDIS está a desenvolver uma aplicação para a sua própria *Step Top Box* (STB) [18]. A STB do AXMEDIS Player consiste num dispositivo para receber o conteúdo AXMEDIS e fazer *play* dos recursos incluídos no conjunto de TV. Dois tipos de STB estão a ser desenvolvidos. Esta STB pode incluir capacidades de gravação HD *(High Definition)* e/ou podem ser descodificadores simples para receber em *stream* ou em *download* do conteúdo AXMEDIS.

O AXMEDIS Player para a STB é de dois tipos:

- Linux baseado numa STB desenvolvida pelo parceiro MBI (criador de descodificadores *Open Sky*) e apresenta características que satisfazem objectos AXMEDIS contendo conteúdos MPEG-2 e MPEG-4 recebidos em *download*;

- Baseado na Kreatel STB devenvolvida pelo parceiro TEO *(Telecom Lithuania)*, para fazer o *play* dos objectos AXMEDIS contendo conteúdos MPEG-2 e MPEG-4 em *stream* e *download* em progresso;

Os *players* AXMEDIS MBI para *Step Top Box* apresentam as seguintes funcionalidades:

- Receber em *stream/download* objectos AXMEDIS;

- *Play* de objectos AXMEDIS protegidos e não protegidos;
- *Play* de recursos com tempo associado como é o caso de vídeo e áudio com funcionalidades para *stop, pause, play, fast forward, fast backward*, etc.
- Suporta DRM *(Digital Rights Management)* e protecção AXMEDIS, incluindo relatórios e logs de acções;
- Mostra metadados;

Para concluir as *frameworks* dos produtos AXMEDIS é mostrado na figura 3.9 um *Player* desenvolvido para PDA, baseado nos formatos referenciados nas outras aplicações AXMEDIS.

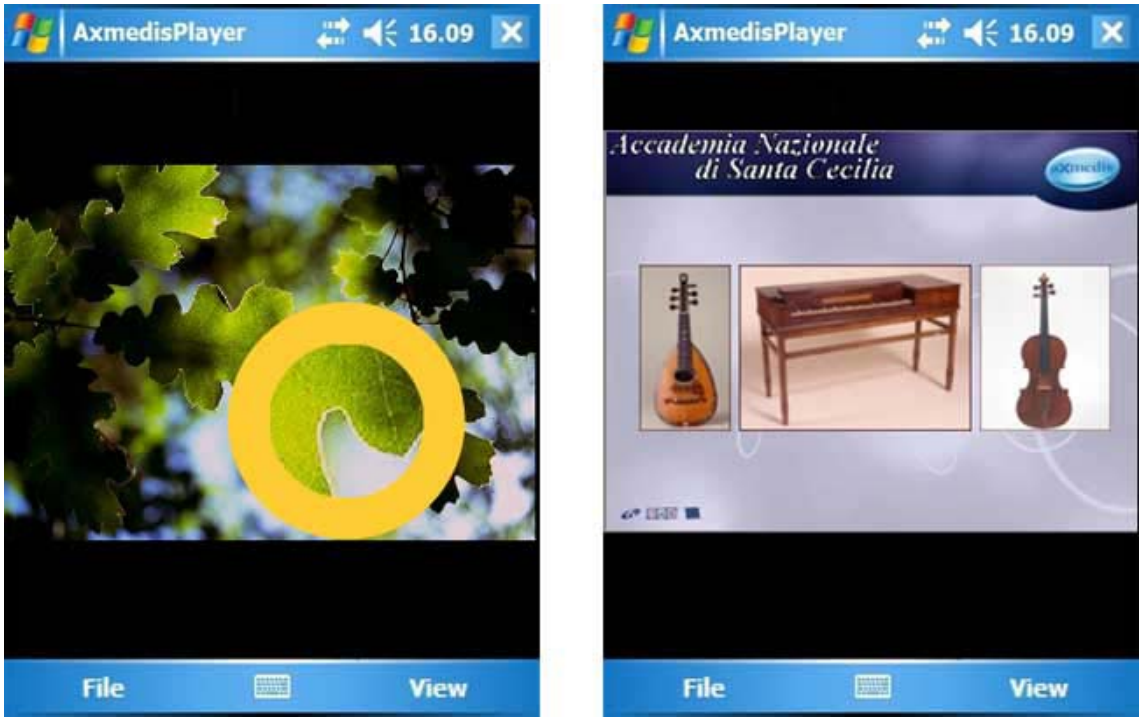


Figura 3.9 – Interface do Player AXMEDIS para PDAs [18].

Limitações:

Além das limitações referidas no DDIBrowser, esta aplicação apresenta outra limitação encontrada. Essa limitação prende-se também com o facto da interoperabilidade, porque tal como o AXMEDIS testou na sua aplicação os DIs do DDIBrowser, também testou os DIs do Enthrone 1. Alguns DIs do Enthrone tinham uma particularidade que causou conflito no AXMEDIS, que era o facto de ter em diferentes *Components*, filhos do mesmo pai, em que cada um deles tinha o mesmo recurso só que com qualidades diferentes. Como o AXMEDIS reproduz todos os *Components* automaticamente, no caso dos DIs do Enthrone reproduzia os mesmos recursos com qualidades diferentes, o que não faz muito sentido. Sendo assim isto levou à questão: "Porque não é o autor do DI a decidir como os *Components* são reproduzidos ?".

3.3 – Ghent's MPEG-21 Applications

O grupo MULTIMEDIA LAB [22] fundado em 2001 pela universidade Ghent da Bélgica, fez a primeira aplicação *online* baseada na norma MPEG-21. Este grupo desenvolveu várias aplicações baseadas na norma MPEG-21 com diferentes propósitos, que se encontram descritas de seguida:

MPEG-21 DIP Terminal Demo

O MPEG-21 DIP Terminal Demo produz uma apresentação multimédia sincronizada contendo vários recursos (texto, imagens, vídeo e áudio), sendo essa sincronização temporal realizada dentro das *Digital Item Declarations*. Ao executar o *demo* pela primeira vez somos confrontados com uma escolha de 3 DIs (figura 3.10.1). O primeiro deles dá acesso à escolha de um *trailer* de um total de 4, como é ilustrado na figura (figura 3.10.3), sendo que cada um dos DIs apresenta vários métodos para sua manipulação (DIMS), como mostra a figura 3.10.5. O Segundo DI tem um método DIM para *play* de uma apresentação *demo* do gladiador, com recursos tais como texto, vídeo e imagens sincronizados temporalmente, tal como pode ser visto na figura 3.10.4. Por fim o último DI apresenta uma lista de músicas que o utilizador pode ouvir como ilustra a figura 3.10.2.

Figura 3.10.1 – Interface principal onde se pode escolher um DI de 3 DIs.

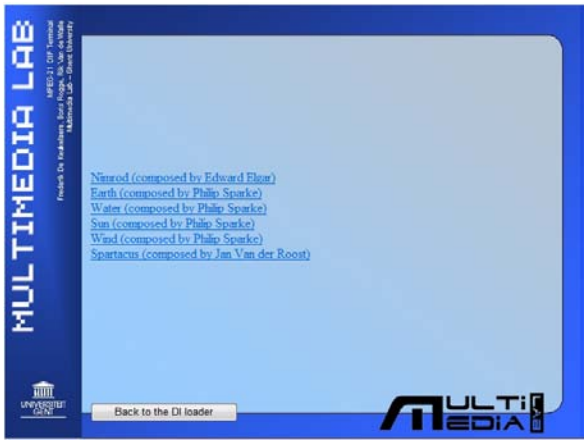


Figura 3.10.2 – Lista de músicas.

Figura 3.10.3 – Escolha um de 4 trailers.

Figura 3.10.4 – Demo do Gladiador.

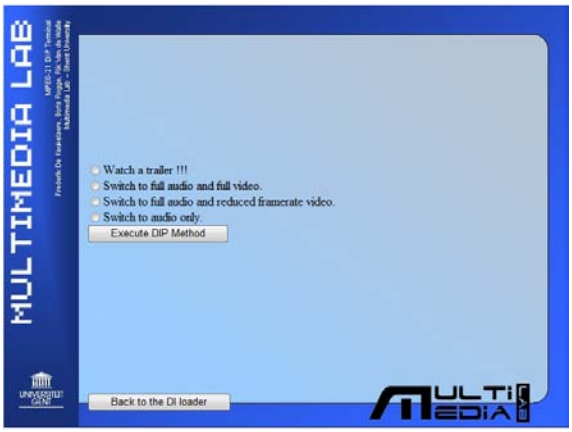


Figura 3.10.5 – Lista de métodos (DIMs) para visualização de um trailer.

Figura 3.10 – Diversas interfaces do demo da Multimedia Lab que são apresentadas ao utilizador.

MPEG-21 Session Mobility

Esta aplicação permite a transferência transparente de sessões de multimédia de um dispositivo para outro dispositivo. Para realização destas sessões entre dois clientes precisa-se de correr o software em dois dispositivos diferentes (por exemplo, dois PC ou um PC e um PDA), uma vez que ele foi implementado em duas plataformas diferentes, para Windows e Windows Mobile 2003.

MPEG-21 Scalable Video-On-Demand

Esta aplicação é uma aplicação de VOD *(Video-On-Demand)* que usa MPEG-21 e metadados *Time-Dependent* para entregar vídeos de MPEG-4 ao cliente. Quando o cliente inicia a aplicação, recebe um DI dum servidor de *streaming*, contendo a lista de vídeos disponíveis no servidor *streaming*. O cliente pode escolher entre dois vídeos que são disponibilizados actualmente, após a escolha o cliente recebe um novo DI que contém informação extra sobre o vídeo que escolheu. Quando o vídeo começa, um *stream* de áudio e vídeo MPEG-4 é enviado para computador do cliente, sendo que ao mesmo tempo o cliente pode ajustar os parâmetros de forma a melhorar a qualidade do *stream*. O objectivo dos metadados *Time-Dependent* é permitir ao cliente essas modificações sem ter que reiniciar o *stream* de vídeo, porque se não haveria um atraso fazendo com que não fosse possível visualizar o vídeo no seu ritmo natural.

Temporal synchronization of media within MPEG-21 Digital Item Declarations

Existem ainda 4 *demos* relativos a este tema em que se pode ver diferentes tipos de recursos (textos, imagens e vídeos) sincronizados temporalmente de diferentes formas, cada uma delas ilustrada em cada *demo*. Estas combinações de sincronização de apresentações multimédia são conseguidas através da combinação da tecnologia HTML+TIME e SMIL.

MPEG-21 wireless application

Esta aplicação tem como objectivo mostrar como pode ser usada a tecnologia de WAP/WML em combinação com tecnologia de MPEG-21. O propósito é criar um dispositivo de interface específica para aplicações de WAP.

3.4 - ADACTUS

Adactus [12] é uma empresa de software que tem como base a norma MPEG-21 para oferecer soluções para entrega e apresentação de conteúdos multimédia adaptados para terminais móveis com Qualidade de Serviço (QoS) e Qualidade de Experiência (QoE). Um dos seus produtos é o Mobilize, que apresenta uma plataforma escalável de um sistema de entrega de conteúdos que possibilita aos fornecedores de conteúdos e *broadcasters*, entregar os seus conteúdos a todos os terminais capazes de processar e apresentar multimédia. Mobilize, suporta vários cenários de consumo de conteúdo para o utilizador final, dos quais constam os seguintes:

- *Live streaming* de video
- *Streaming of on demand*- clips de video
- *Download* de videos
- Subscrição para um *podcast/blog*
- *Download* de conteúdo de vídeo para normas- conjunto de terminais multimédia

Mobilize é projectada para integrar diferentes *workflows* de conteúdos usando uma interface XML flexível para preparar e publicar alguns *workflows* de conteúdo em *"easy way"*. O modo de funcionamento começa por passar o XML para o URL do servidor do Mobilize, que então faz a validação e obtém os recursos. Após esta etapa o servidor notifica os consumidores registados sobre a disponibilidade de um *Digital Item*, que quando é acedido pelos consumidores, os recursos nele presente podem ser adaptados especificamente para o tipo de terminal que o utilizador está a usar, tendo em conta requisitos do terminal como codificação de vídeo, tipo e qualidade de som e tamanho de imagem.

3.5 - ENIKOS

Enikos [16] é mais uma companhia que desenvolve e fornece serviços de *software* para aplicações de multimédia da próxima geração com base na norma MPEG-21. Fundada em Sydney, Australia, Enikos tem desenvolvido a EMP *(Enhanced Media Platform)* que está baseada em patentes únicas de propriedade intelectual (IP) e que permite aos utilizadores

combinar recursos tais como áudio, vídeo, paginas *Web* e todas as formas de *media* digital para criar e personalizar experiências multimédia.

Enikos lançou o DICreator, que permite aos utilizadores explorar o potencial do *Digital Item* como contentor para o conteúdo deles, permitindo criar facilmente, editar e ver DIs construídos de conteúdo actual e de metadados. Esta *framework* MPEG-21 suporta Direitos, Adaptação e mecanismos de DRM *(Digital Rights Management)*. O DI Creator consiste em uma GUI *(Graphical user Interface)* completa baseada em editor e um *browser* de *Digital Item*, os quais se designam de DI Editor e DI Browser respectivamente. As características de cada um deles estão ilustradas a abaixo:

DI Editor:

- É fácil de usar, e tem vista hierárquica de *Digital Items*;
- Menus Inteligentes para adicionar elementos do *Digital Item* na posição hierárquica desejada;
- Mostra e renderiza os recursos multimédia (incluindo lançamento de externos *players* se necessário).
- Insere recursos multimédia através de *browsing* do sistema de ficheiros ou então editar directamente a ref (referencia) do atributo do *Resource*.
- Validação de DIs no momento do *carregamento* e também quando se guarda.
- Usa menus *pull-down* para assistir a entrada de valores de atributos;

DI Browser:

- *Split pane windows* para vista de navegação, vista de conteúdos, e uma área de mensagem.
- Mostra a descrição de *Items* selecionados incluindo texto e outro tipo de *Descriptors*.
- Apresentação de *Choices* como imagens *thumbnails*, *radio buttons*, *check boxes* ou menus.
- Interface Simples com a história dos DIs procurados.

3.6 – Klagenfurt University

Tem, como outras instituições e grupos referidos acima, contribuído para o esforço de normalização do MPEG21. Eles desenvolveram aplicações, na maioria relacionada com a parte 7 *(Digital Item Adaptation)* da norma MPEG-21. Três outras aplicações desenvolvidas são o DI Builder e o DI Consumer, as quais se encontram descritas a seguir [52]:

DI Builder:

É uma aplicação Java *on-line*, a qual permite a qualquer utilizador que lhe aceda, criar um *Digital Item* simples de acordo com a norma MPEG21. Os DIs são constituídos por um ou mais recursos de vídeo, sendo que estes podem ter *Descriptors* associados, *Choices*, Licenças e *Components*. Para criar um DI o utilizador tem de seguir alguns passos. Quando a aplicação *online* começa, é apresentado ao utilizador a descrição da *applet* e informação de como pode usá-la (Figura 3.11.1). O próximo passo consiste em escolher quais os recursos de vídeo para o DI como mostra a figura 3.11.2; de seguida o utilizador pode criar descrição MPEG-7, *Descriptors* DIA, Licenças e *Choices* (sendo que estes últimos 4 passos são opcionais para o utilizador) como está ilustrado nas figuras 3.11.3 e 3.11.4. No final destes passos o utilizador só precisa de criar o *Digital Item* adicionando uma descrição global e definir o local onde ele será guardado (Figura 3.11.5).

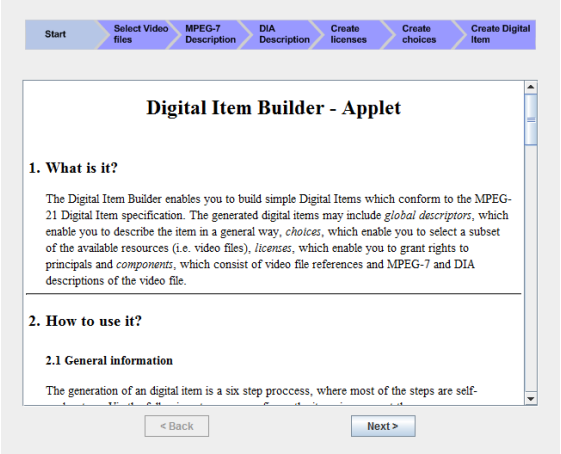


Figura 3.11.1 – Descrição do funcionamento da *applet*.

Figura 3.11.2 – Recursos para o *Digital Item*.

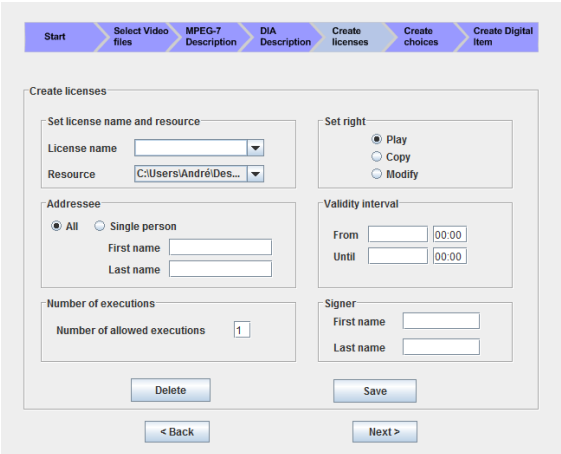


Figura 3.11.3 – Criação de licenças.

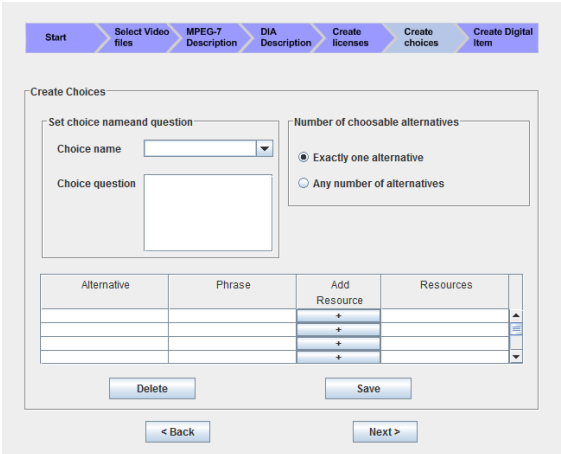


Figura 3.11.4 – Criação de *Choices*.

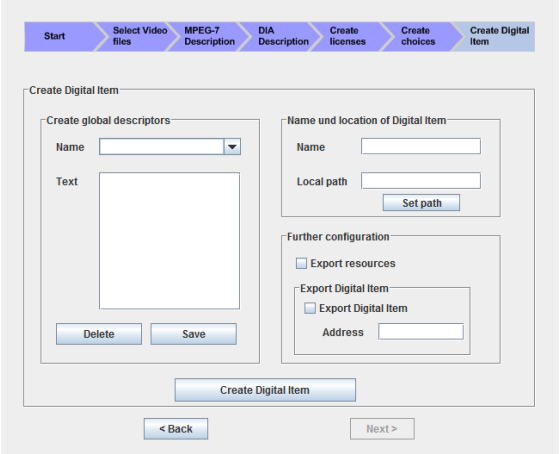


Figura 3.11.5 – Criação do *Digital Item*.

Figura 3.11 – Diferentes interfaces do DI Builder que permitem a criação de um *Digital Item*.

DI Consumer:

Tal como a anterior, esta aplicação também pode ser testada por qualquer utilizador que o assim o pretenda. A primeira coisa que é apresentada ao utilizador é a descrição do funcionamento da *applet*. Além deste passo, a aplicação, tem mais 3 passos para se dar o consumo do *Digital Item*. O primeiro é a escolha do DI para consumo (figura 3.12); o 2º passo é escolher os recursos a serem consumidos com base em *Choices;* e por fim o último passo é consumir o DI de acordo com as escolhas feitas no passo anterior.

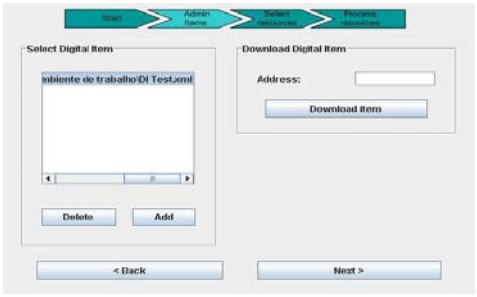


Figura 3.12 – Interface do DI Consumer que permite escolher o *Digital Item* para que o utilizador possa visualizar o seu conteúdo.

3.7 – ENTHRONE 1

ENTHRONE [24] é um projecto fundado pela Comunidade Europeia. A visão do projecto foi fornecer acesso de forma transparente aos recursos multimédia assegurando QoS e eficiência, usando tecnologias distribuídas suportadas por todas as camadas *(middleware)*, para construir serviços interoperáveis. ENTHRONE desenvolveu um componente integrante designado de IMS *(Integrated Management Supervisor),* e que é responsável pelas decisões que são tomadas e acções aprovadas na cadeia de entrega. O IMS teve um papel principal na manipulação de toda a cadeia de distribuição de conteúdo. Isto suporta a cooperação entre várias entidades na cadeia de distribuição de informação digital, da geração de conteúdo e criação de serviços para o terminal do utilizador. O IMS é constituído por 3 sub-sistemas, o *IMS dispatcher* que permite realizar uma pesquisa de conteúdos noutros fornecedores de serviços, o *IMS content*

Manager que contém os metadados usados para pesquisa e para adaptação, e o *IMS Terminal Device Manager* que envia a informação de contexto descrevendo as capacidades do terminal e as características da qual o terminal reside.

Outras das funcionalidades do projecto são a integração do projecto com DID Browser, que é uma aplicação terminal usada para pesquisa, visualização de DIDs e também permite fazer *apresentação* de recursos.

3.8 – ENTHRONE 2

Este projecto visa desenvolver uma solução de gestão integrada que cobre toda a cadeia de distribuição audiovisual, incluindo geração e protecção de conteúdo, distribuição através de redes heterogéneas e recepção nos terminais dos utilizadores, garantindo qualidade de serviço. Os principais clientes do ENTHRONE serão os fornecedores de serviços tais como *broadcasters*, operadores de telecomunicações e empresas agregadoras de conteúdos, entre outros [25].

3.9 – DANAE

Danae *(**D**ynamic and distributed **A**daptation of scalable multimedia co**N**tent in a context-**A**ware **E**nvironment)* [20] é um projecto de 30 meses fundado pela European IST. Os objectivos deste projecto são especificar, desenvolver, integrar e validar uma *framework* completa para teste, capaz de fornecer um serviço de qualidade multimédia *end-to-end* para o utilizador final. O trabalho pode ter em conta a definição de formatos multimédia escaláveis com metadados associados a eles; a adaptação deles para o contexto de sessão para fazer a caracterização completa de adaptação de cenas multimédia distribuída através de uma optimização global do áudio, vídeo, gráficos 2D, e o transporte e entrega de conteúdo multimédia para o utilizador final. A DANAE também pretende analisar modelos potenciais de novos negócios que permitirão avaliar a viabilidade comercial dos novos serviços conduzidos por DANAE. Na figura 3.13 está ilustrado a arquitectura do projecto DANAE, baseada nas normas MPEG-21 e MPEG-4.

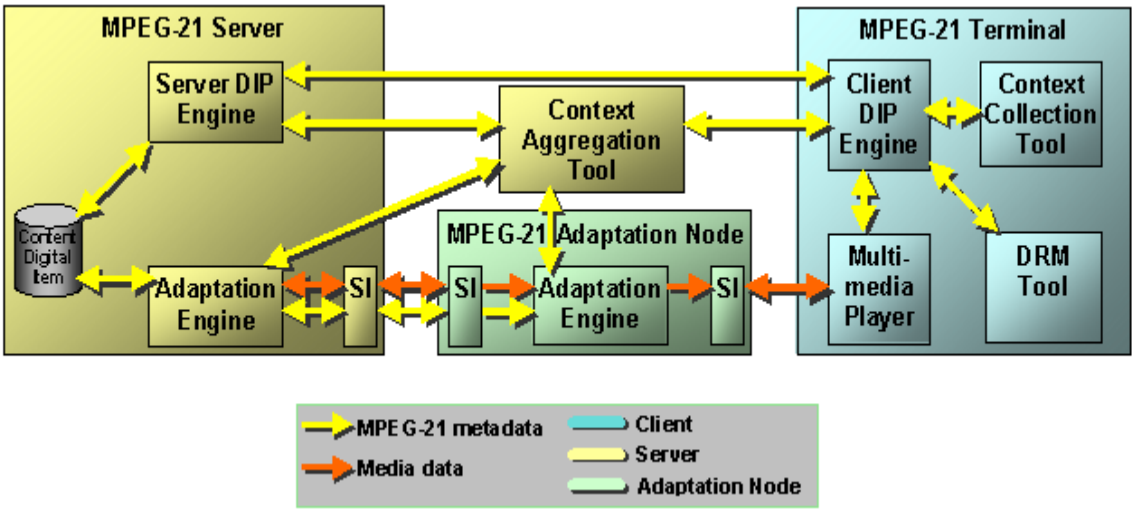


Figura 3.13 - Arquitectura do projecto DANAE [19].

No diagrama da arquitectura DANAE podemos visualizar vários módulos interligados uns com os outros, os objectivos desses módulos são descritos de seguida:

- O módulo CAT (*Context Aggregation Tool)*, que como o próprio nome do módulo indica, é a unidade responsável por armazenar e agregar informação de contexto que vem de todos os terminais que estão a consumir conteúdos do servidor. Como podemos observar, contexto é enviado ao CAT pelo *DIP Engine* do Cliente como uma entrada para o *server adaptation engine*. No módulo *Adaptation Node* o CAT permite controlar os parâmetros de adaptação manualmente.

- O módulo *Adaptation Engine* é responsável por adaptar cada conteúdo multimédia de acordo com o contexto de cada terminal fornecido pelo CAT. O *Adaptation Engine* é activado pela *Interface de Streaming* (SI), em *Multimedia Player*.

- O módulo SI *(Streaming Interface)* é responsável por obter conteúdo multimédia da *Adaptation Engine*, enviando para o respectivo terminal.

- O módulo *Server DIP Engine* é o ponto de comunicação com o *DIP Engine* Cliente. Administra sessões, cria e transfere o diferente *Digital Item* que é necessário para *a DIP Engine* Cliente.

- A *Context Collection Tool* é responsável por coleccionar vários dados de sensores localizados dentro de cada terminal, sendo estes dados processados e enviados ao servidor para dirigir a sua adaptação.

- A DRM *Tool* é responsável por administrar a decifração de códigos, quando o terminal tiver que lidar com conteúdo protegido.

- O *Multimedia Player* é responsável por reproduzir conteúdo de multimédia.

- Por fim o módulo *Client DIP Engine* é responsável por criar e dirigir todos os outros módulos no terminal. Este módulo envia informação de contexto coleccionada ao CAT, além de obter e processar a DID, e dirigir o módulo *Multimedia Player*.

3.10 - MUFFINS

MUFFINS *(Multimedia Framework For Interoperability in Secure)* [13] é um projecto fundado pela Comunidade Europeia e finalizado em Dezembro de 2003. Este projecto focou-se em resolver o problema da entrega de recursos multimédia de forma transparente através de uma vasta gama de redes e dispositivos, para diversos tipos de utilizadores em múltiplos domínios de aplicação. Um dos objectivos do projecto Muffins foi contribuir para o desenvolvimento da norma MPEG-21 e investigar o problema da descrição, entrega e protecção de conteúdos multimédia. A aplicação terminal desenvolvida pelo Muffins é destinada aos utilizadores, para o consumo de conteúdos multimédia. A aplicação inclui funcionalidades como pesquisa, *upload* e *download* de conteúdo, podendo também ser usada para comprar ou vender licenças; além de ter a inclusão de um *player* multimédia que é armazenado localmente. O processamento de selecção, aquisição e consumo de um *Digital Item* é apresentado no diagrama da figura 3.14.

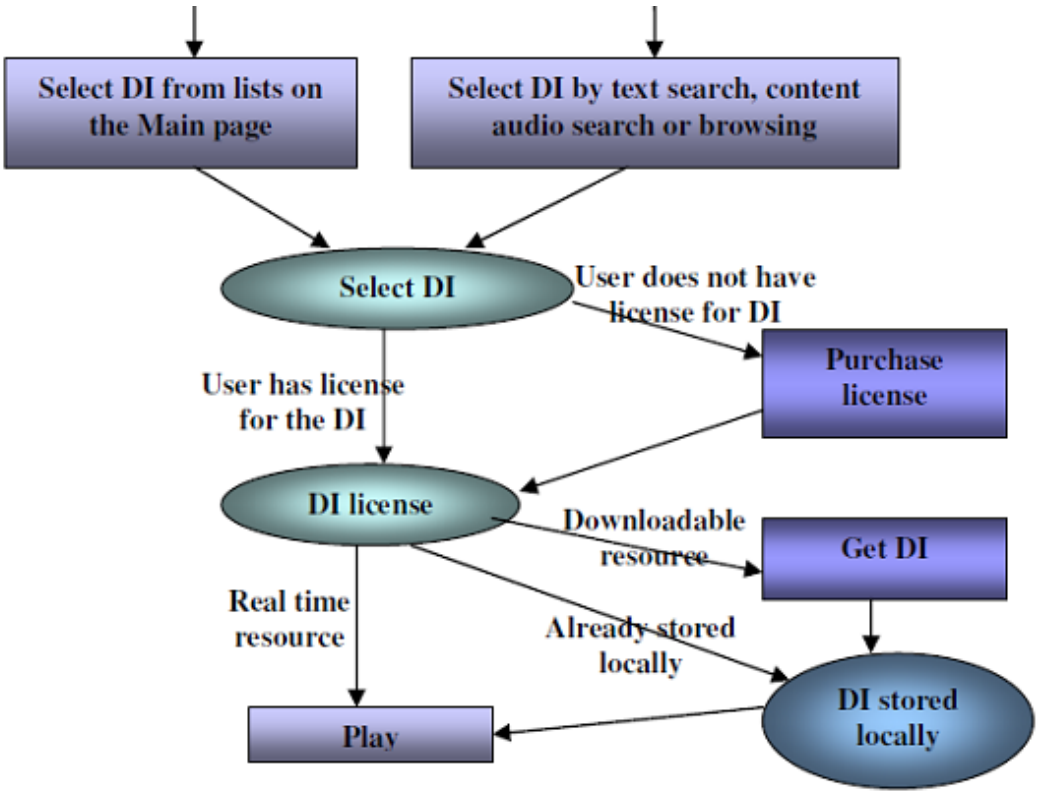


Figura 3.14 - Processamento de selecção, aquisição e consumo de um *Digital Item* [5].

3.11 - Conclusão

Como foi abordado neste capítulo existem diversas aplicações MPEG-21 com diferentes requisitos e diferentes funcionalidades, sendo que cada uma delas tem uma finalidade para a qual foi ou está a ser desenvolvida. Algumas delas recorrem à DIP para processamento de *Digital Items*, outras utilizam outra forma de processamento para interpretá-los, respeitando contudo o que está referido na norma. No entanto a maior parte destas aplicações apresenta algumas limitações, de entre as quais se salienta a falta de interoperabilidade entre elas, ou seja, um mesmo *Digital Item* não é processado de forma conveniente quando utilizado pelas aplicações.

O próximo capítulo aborda a *Digital Item Processing* de uma forma mais detalhada, bem como as operações que ela disponibiliza, DIBOs e DIXOs.

Capítulo 4

DIP

Este capítulo explica o *software* de referência da norma MPEG-21 para DIP, em termos de funcionalidades, limitações e arquitectura, uma vez que serviu de base para a construção da aplicação MPEG21 DIP Teach, e dá ênfase à *Digital Item Processing*, explicando de forma detalhada as capacidades que ela apresenta, através dos métodos (DIMs), que permitem adicionar interactividade a um *Digital Item*.

4.1 – Software de Referência MPEG-21 para DIP

O *software* de referência (SwRef) [44] tem como objectivo mostrar o funcionamento da norma MPEG-21 a nível prático, com base na informação referida ao longo das 19 partes dela, além de ilustrar bem o funcionamento da DID (DIDEngine) e da DIP (DIPEngine) a nível de código para que possíveis programadores da norma possam contribuir para o desenvolvimento da *framework*.

4.1.1– Funcionalidades

A nível prático (quando se executa), o *software* ilustra o funcionamento da DIP através de janelas *popup* que são apresentadas ao utilizador após ter passado a DID como argumento do módulo do DIDEngine e ter executado a classe DIP Demo. O software possibilita a execução de métodos contidos na DID quer eles sejam constituídos por DIBOs, quer por DIXOs ou mesmo por ambos. Quando um utilizador estabelece o caminho de uma DID como parâmetro de entrada da classe DIDEngine e executa a classe DIP Demo, a primeira coisa que ele vai fazer é verificar se a DID é válida, ou seja, se está de acordo com a norma. Quer ela seja válida ou não, ele vai continuar a processar a DID mesmo que tenha erros. O motivo disto acontecer deve-se ao facto de ser uma limitação da linguagem DOM *(Document Object Model)* [37], que neste *software* é usada para fazer o *parsing* do documento, bem como a sua análise para o devido processamento por parte do *software*. Além desta limitação da linguagem DOM foi encontrada outra, e que está relacionada com os endereços passados ao DIDEngine, isto é se

os endereços não contiverem caracteres especiais como o caso de acentos e de cedilhas, a DID é processada normalmente, se contiverem então a DID não é processada.

Após ter feito o *parsing* de uma DID, o *software* vai ver se existem DIMs para execução na DID através do módulo DIP Engine. Se não existirem é retornado na consola da aplicação que não existem DIMs terminando a execução do *software*, caso contrário ele vai ver primeiro se existe DIMs automáticas; se sim então executa a primeira e termina a execução do *software*; se não mostra numa janela *popup* as DIMs que existem na DID para o utilizador escolher; após a escolha de uma DIM, se ela tiver argumentos associados é apresentado ao utilizador os argumentos para ele escolher um, se não tiver argumentos é executada a DIM e no final é terminada a execução do *software* de referência. De salientar que se for uma DIM com argumentos o utilizador escolhe um (figura 4.1) e com base no código da DIM o argumento é tratado.

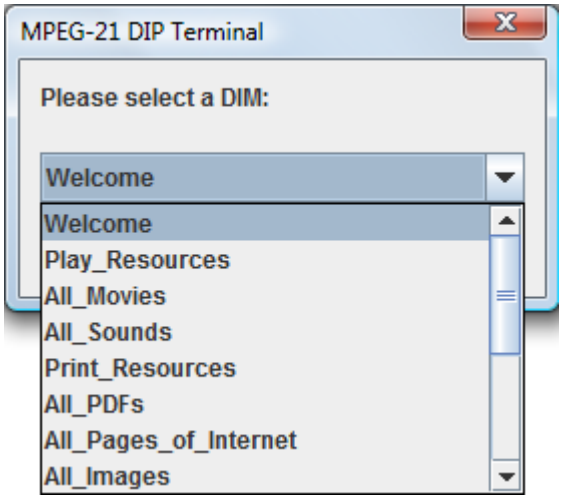


Figura 4.1.1 – Lista de DIMs de uma DID.

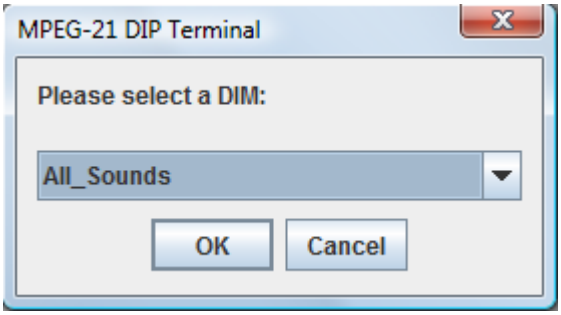


Figura 4.1.2 – Escolha por parte do utilizador da DIM All_Sounds.

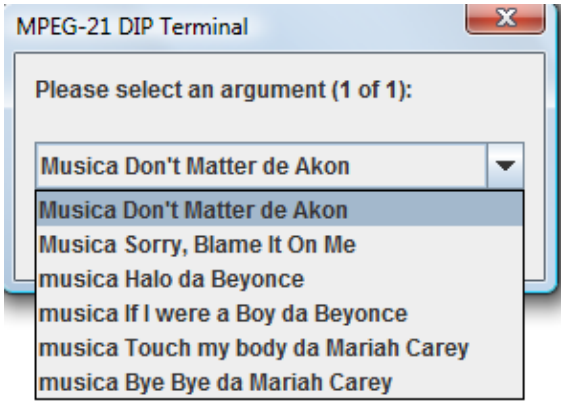


Figura 4.1.3 – Lista de argumentos da DIM All_Sounds.

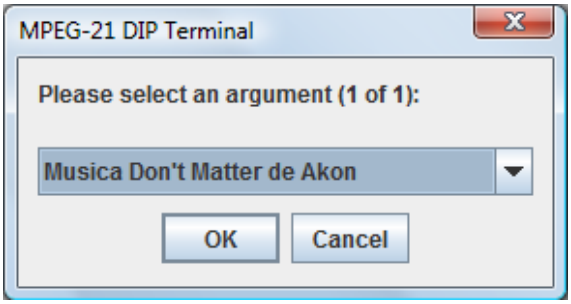


Figura 4.1.4 – Escolha do argumento Musica Don´t Matter de Akon.

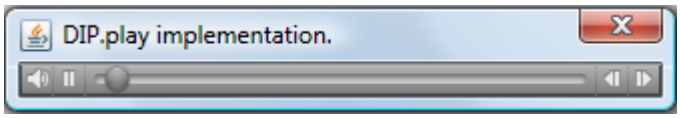


Figura 4.1.5 – Reprodução da música escolhida em 4.1.4.

Figura 4.1 – Exemplo das interfaces que são apresentadas antes, durante e depois da execução de uma DIM com argumentos.

O Diagrama seguinte (figura 4.2) mostra a sequência de passos que um utilizador é confrontado quando passa uma DID como argumento do DIDEngine e executa o *software* de referência.

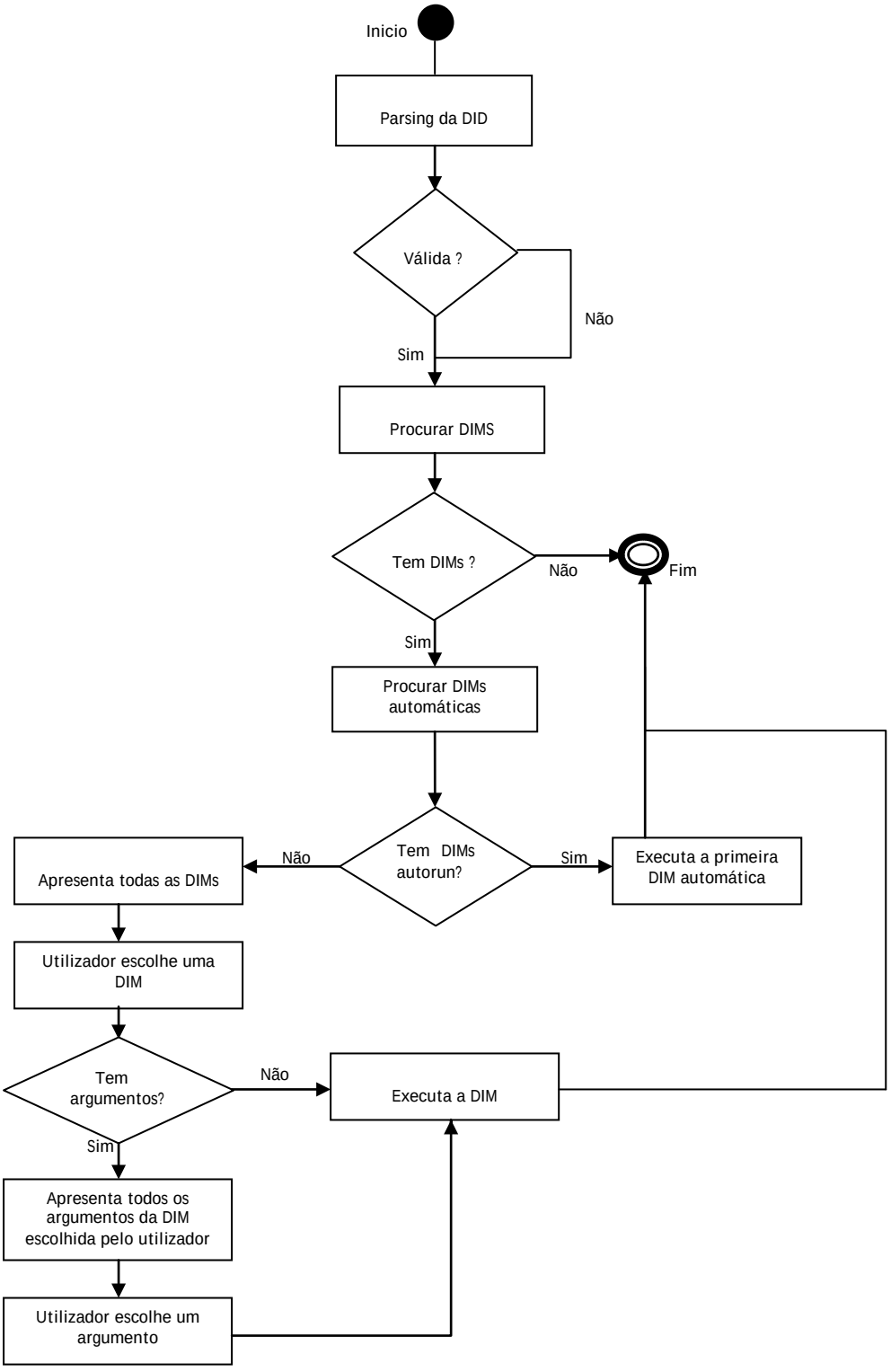


Figura 4.2 – Diagrama que mostra a sequência de passos que um utilizador é confrontado quando executa o *software* de referência.

4.1.2– Implementação e Arquitectura

A nível de arquitectura o *software* é composto por dois módulos principais, os quais já foram referidos e que são o DIDEngine e o DIPEngine. O Módulo DIDEngine é responsável por

fazer o carregamento e o *parsing* da DID, bem como verificar o estado das *Selections*, processar *Selections* definidas como *default* atribuindo o estado *true*, dos três estados possíveis *(true, false, undecided)* e processar *Assertions*. O Módulo DIPEngine recebe como parâmetro de entrada o documento (DID) retornado pela DIDEngine. O DIPEngine é responsável por procurar todas as DIMs existentes na DID de acordo com o identificador que as identifica, para ver se se trata de uma DIM. Além disso o módulo permitir mandar executar as DIMs e permite processar os argumentos de cada DIM. De salientar que quando se refere a mandar executar, significa que não é este módulo que executa, mas sim o módulo DIMEngine, ou seja, a DIPEngine prepara as DIMs com todos os parâmetros necessários de acordo com o seu código na DID, para o DIMEngine executar.

Outra coisa a salientar, e que neste ponto é importante referir, é que as DIMs são declaradas dentro de um *Component,* tendo um identificador que as identifica, permitindo assim distinguir dos outros *Components* que não tem DIMs. A figura 4.3 mostra um exemplo de declaração de uma DIM, como podemos observar o identificador que identifica que este *Component* tem uma DIM para DIPEngine, é o conteúdo textual que está entre a *tag* <dip:Label>. Continuando a analisar a figura, dentro da *tag* <dip:Methodinfo/> podemos definir se a DIM é para executar de forma automática ou não, ou seja, quando do carregamento da DID ou do DI. Relativamente à DIM podemos visualizar que ela é declarada sempre dentro de um *Resource* com o *mimeType* application/mp21-method. A DIM designa-se então por *Welcome* e tem na sua função uma DIBO, designada de *alert*, a qual pertence à parte 10 da norma e que será analisada mais à frente em conjunto com outras. O resultado da execução da DIM da figura 4.3 está ilustrado na figura 4.4.

```
<Component>
        <Descriptor>
          <Statement mimeType="text/xml">
                <dip:MethodInfo autoRun="false" />
          </Statement>
        </Descriptor>
        <Descriptor>
          <Statement mimeType="text/xml">
             <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
          </Statement>
        </Descriptor>
        <Resource mimeType="application/mp21-method"><![CDATA[
                function Welcome()
                {
                 DIP.alert("I'am mad.",MSG_INFO);
        }]]>
        </Resource>
</Component>
```

Figura 4.3 – Exemplo de uma declaração de uma DIM num *Component* de uma DID.

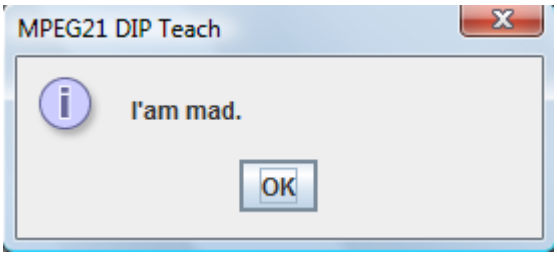


Figura 4.4 – Resultado da execução da DIM da figura 4.3.

Outro módulo a qual o DIPEngine recorre é o módulo DIPUtils e o módulo DIIEngine. O DIPUtils que além de apresentar as DIMs e os argumentos para o utilizador, auxilia o DIPEngine a procurar os nomes das DIMs, os nomes dos argumentos, os textos dos *Descriptors* e a obter os filhos de um determinado elemento, enquanto que o DIIEngine auxilia na identificação de certos elementos na DID, para possível processamento por parte de algum módulo.

O módulo DIMEngine, como já foi visto é a unidade responsável pela execução das DIMs com os argumentos associados a elas. O que este módulo faz é interpretar as DIMs com o auxílio doutros módulos existentes no software, os chamados Wrappers, sendo que estes últimos tem por missão verificar se o código de uma DIM, mais propriamente as DIBOs que podem estar associadas a ela, estão correctas em termos da sua chamada e dos parâmetros que lhe são passados como argumentos, para a execução delas em ambiente ECMACscript [31]. Em baixo encontram-se listados os diferentes módulos Wrappers, em que cada um é responsável por tratar as DIBOs que lhe estão associadas:

- DIAWrapper
- DIDWrapper
- DIIWrapper
- DIPWrapper
- DOMWrapper
- RELWrapper

Se existir algum erro é lançado um erro através do módulo DIPErrors, terminando tudo. Se não, isto é, se não houver nenhum erro no código escrito dentro da DIM, os Wrappers mandam executar as DIBOs e as DIXOs. E aí entram por fim os últimos módulos que estão relacionados com as DIBOs e com as DIXOs. Sendo que estas últimas são chamadas através de uma DIBO, designada de RunJDIXO, daí considerar no capítulo 2 da dissertação que as DIXOs têm algumas restrições impostas pela norma, pois têm que seguir algumas regras para que a sua execução seja efectuada com sucesso.

Módulos que contém as declarações das DIBOs para que o autor do DI só precise de chamar pelo nome da DIBO (ex: DIP.wait(9000)) na DID:

- DIA
  DIBOs: *adapt*;

- DID
  DIBOs: *areConditionsSatisfied*, *configureChoice*, *setSelection*;

- DII
  DIBOs: *getElementsByIdentifier*, *getElementsByRelatedIdentifier*, *getElementsByType*;

- DIP
  DIBOs: *alert*, *execute*, *getExternalData*, *getObjectMap*, *getObjects*, *getValues*, *play*, *print*, *release*, *runDIM*, *wait*;

- DIPError
  DIBOs: *getDIPErrorCode*;

- JDIBOFactory
  DIBOs: getJDIBOObject;

- ObjectMap
  DIBOs: *getArgumentList*, *getArgumentListCount*, *getMethodCount*, *getMethodWhithArgs*, *getObjectOfType*, *getObjectsOfType*, *getObjectsOfTypeCount*, *getObjectTypeCount*, *getObjectTypeName*;

- PlayStatus
  DIBOs: *getStatus*;

- REL
  DIBOs: *getLicense*, *queryLicenseAuthorization*;

Módulos que contém a implementação das DIBOs referenciadas nos módulos acima:

- DIAImp - contém a implementação das DIBOs do módulo DIA;
- DIDImp - contém a implementação das DIBOs do módulo DID;
- DIIImp - contém a implementação das DIBOs do módulo DII;
- DIPImp - contém a implementação das DIBOs do módulo DIP;
- GlobalEnvImp
- JDIBOFactoryImp - contém a implementação das DIBOs do módulo JDIBOFactory;
- ObjectMapImp - contém a implementação das DIBOs do módulo ObjectMap;
- PlayStatusImp - contém a implementação das DIBOs do módulo PlayStatus;
- RelImp – contém a implementação das DIBOs do módulo REL;
- URIClassLoader

Se houver algum erro na execução das DIBOs ou na execução das DIXOs, é enviado outra vez para o módulo DIPError, o erro em questão, terminando a execução, se não é apresentado o resultado da DIM de acordo com o propósito com que ela foi realizada.

O *software* ainda tem um módulo mediaPlayer, para apresentação de vídeo e áudio nos formatos aceites pelo *Quicktime*. E o módulo CheckBoxDialog que permite mostrar as opções das *Choices* declaradas na DID, para o utilizador escolher. Por fim existe o módulo JDIXO que permite interpretar as DIXOs mandadas executar.

Na figura 4.5 é mostrada a arquitectura do software de referência da norma MPEG-21 para DIP, com base nos principais módulos e nas relações entre eles.

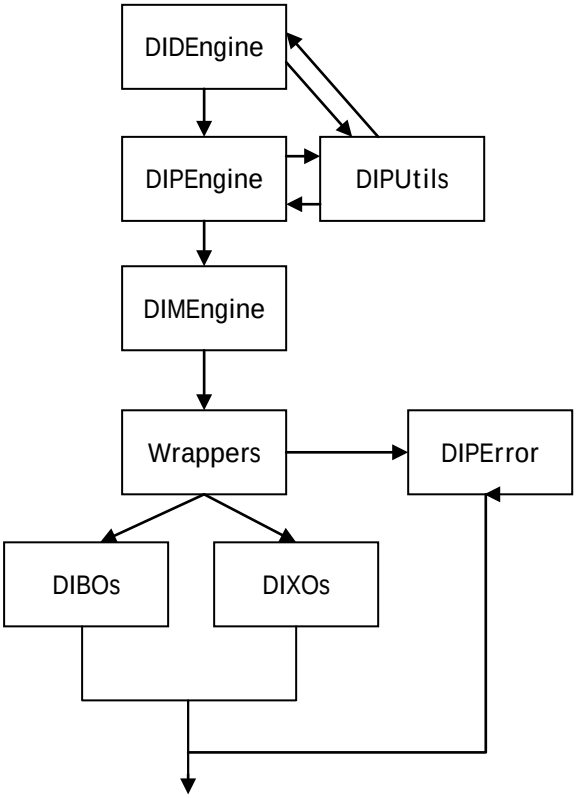


Figura 4.5 – Arquitectura do *software* de referência da norma MPEG-21, envolvendo as relações entre os principais módulos.

4.1.3- Limitações

Com base na análise do *software* de referência e nos testes realizados com ele, encontraram-se os seguintes problemas:

- Não existe retorno visual para o utilizador do resultado da execução de uma DIM e também se não existirem DIMs;
- Reproduz qualquer recurso num *player* (quer seja um PDF) com a DIBO Play, não tendo em conta os *mimeTypes*, sendo necessário mudar a implementação da DIBO de modo a que o utilizador possa visualizar a maioria dos recursos;
- O DIP Engine não está preparado para executar várias DIMs definidas como automáticas;

- Não validação de *Digital Items*;

- URLs dos DIs com acentos ou com outros tipos de caracteres especiais, não são processados.

4.2 – Utilização de DIBOs do Software de Referência

Este tópico faz referência a cada uma das DIBOs existentes na norma MPEG-21 e que estão implementadas no *software* de referência da norma. Dá-se relevância ao que cada uma delas tem por objectivo fazer, bem como aos parâmetros que lhe são passados como argumentos. Relativamente às DIBOs que foram testadas durante a dissertação, é mostrado um exemplo simples de um resultado de teste, relacionado com a sua execução, tal como o raciocínio para ter dado aquele resultado final. Mas para que o raciocínio e o teste sejam feitos é necessário ter uma declaração de uma DID para interpretarmos os resultados dos testes realizados. A figura 4.6 mostra uma DID simples ainda sem métodos (DIMs). Nesta altura será possível para o leitor perceber a forma como ela está hierarquizada, bem como uma parte ou a totalidade da sua interpretação.

```
<?xml version="1.0" encoding="UTF-8"?>
<Container>
        <Descriptor>
                <Statement mimeType="text/plain">Aula de Musica</Statement>
        </Descriptor>
                <Item>
                        <Descriptor>
                                <Statement mimeType="text/plain">Hoje podemos visualizar um video de Musica no
formato mpeg e ouvir uma musica em formato MPEG 1 AUDIO LAYER 3 mais conhecido pelo formato mp3.</Statement>
                        </Descriptor>
                <Choice choice_id="choice_01" default="neyo_video">
                        <Descriptor>
                                <Statement mimeType="text/plain">O que pretendes visualizar?</Statement>
                        </Descriptor>
                        <Selection select_id="neyo_video">
                                <Descriptor>
                                        <Statement mimeType="text/plain">Video do Neyo</Statement>
                                </Descriptor>
                        </Selection>
                        <Selection select_id="b">
                                <Descriptor>
                                        <Statement mimeType="text/plain">Musica da Kelly Rowland</Statement>
                                </Descriptor>
                        </Selection>
                </Choice>
                <Item id="neyo">
                        <Condition require="neyo_video"/>
                        <Descriptor>
                                <Statement mimeType="text/plain">Video so sick do Neyo</Statement>
                        </Descriptor>
                        <Descriptor>
                                <Statement mimeType="text/xml">
                                        <dii:Identifier>urn:dii:example:video</dii:Identifier>
                                </Statement>
                        </Descriptor>
                        <Descriptor>
                                <Statement mimeType="text/xml">
                                        <dip:ObjectType>urn:foo:ALL</dip:ObjectType>
                                </Statement>
                        </Descriptor>
                        <Descriptor>
                                <Statement mimeType="text/xml">
                                        <dip:ObjectType>urn:foo:Movie</dip:ObjectType>
                                </Statement>
                        </Descriptor>
                        <Component>
                                <Resource mimeType="video/mpeg" ref="C:\DIDs_DT\Resources\neyo.mpg"/>
                        </Component>
                </Item>
                <Item id="musica">
                        <Condition require="b"/>
                        <Descriptor>
                                <Statement mimeType="text/plain">Musica da Kelly rowland</Statement>
                        </Descriptor>
                        <Descriptor>
                                <Statement mimeType="text/xml">
                                        <dii:Type>urn:dii:example:type</dii:Type>
                                </Statement>
                        </Descriptor>
                        <Descriptor>
                                <Statement mimeType="text/xml">
                                        <dip:ObjectType>urn:foo:ALL</dip:ObjectType>
                                </Statement>
                        </Descriptor>
                        <Descriptor>
                                <Statement mimeType="text/xml">
                                        <dip:ObjectType>urn:foo:Sound</dip:ObjectType>
                                </Statement>
                        </Descriptor>
                        <Component>
                                <Resource mimeType="video/mpeg" ref="C:\DIDs_DT\Resources\work.mp3"/>
                        </Component>
                </Item>
        </Item>
        <Item id="DIMs">
                        “Acrescentar DIMs aqui”
        <Item>
    </Container>
    </DIDL>
```

Figura 4.6 – Exemplo de uma declaração de um *Digital Item* para análise através das DIBOs da norma.

Os *Components* com as DIMs, que vão ser referidos nos exemplos das DIBOs, podem ser inseridos na figura 4.6 entre o *Item* com id igual a DIMs. (onde diz na figura 4.6 "Acrescentar DIMs aqui").

4.2.1– DIBOs da DIA (Digital Item Adaptation)

- adapt(Component, Metadata) – Esta DIBO permite adaptar um recurso com base nos metadados que lhe são passados.

  Parâmetros:
  Component – Elemento DOM representando o *Component* ou *Descriptor* a ser adaptado;
  Metadata – Vector de elementos DOM representando a informação adicional;

Esta DIBO é usada quando pretendemos apresentar um recurso com uns determinados metadados associados e quando existem diferentes dispositivos finais com requisitos diferentes (por exemplo em termos de capacidade de processamento).

4.2.2– DIBOs da DID (Digital Item Declaration)

- configureChoice(Choice) – Pede ao utilizador para configurar uma *Choice* na DID. Retorna *true* se a *Choice* foi modificada, *false* se não.

  Parâmetros:
  Choice – *Choice* a configurar pelo utilizador;

Para testar-mos esta DIBO poderíamos criar uma DIM, como mostra o *Component* da figura 4.7 , após adicionarmos este *Component* à DID da figura 4.6, e executarmos esta DID com o *software* de referencia da norma, poderemos chegar aos seguintes resultados.

```
<Component>
        <Descriptor>
                <Statement mimeType="text/xml">
                        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
                </Statement>
        </Descriptor>
        <Resource mimeType="application/mp21-method"><![CDATA[
                function teste_da_DIBO_configureChoice()
                {
                        var choice=didDocument.getElementById("choice_01"); //1ª instrução
                        return DID.configureChoice(choice); //2ª instrução
                }]]>
        </Resource>
</Component>
```

Figura 4.7 –DIM que permite testar a DIBO configureChoice.

Relativamente à DIM "teste_da_DIBO_configureChoice()" da figura 4.7, podemos verificar que para encontrar na DID a *Choice* que pretendemos, usamos o id associado a ela, através do uso do DOM na primeira instrução("var choice=didDocument.getElementById("choice_01");"). Este princípio aplica-se quer para a *Choices*, quer para as *Selections*, quer para as *Conditions*, quer para qualquer elemento que tenha um id e queiramos aceder-lhe. Se analisarmos a DID da figura 4.6, observamos que essa *Choice* tem duas *Selections* (opções) e que tem como estado *true* a primeira *Selection*, devido ao facto de ter sido atribuído no atributo *default* da *Choice*, o id da primeira *Selection*. Quanto à segunda *Selection*, como não existe nenhuma referência a ela quanto ao seu estado, conclui-se que não se sabe, ou seja, está no estado *undecided*. Na segunda instrução invocamos a configuração da *Choice*, e se executarmos esta DIM podemos constar realmente destas observações como ilustra a figura 4.8.

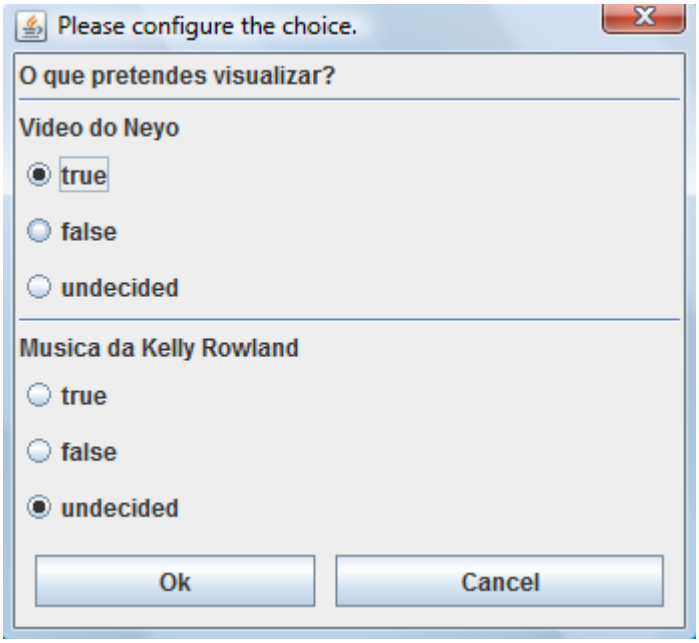


Figura 4.8 – Resultado da execução da DIBO ConfigureChoice.

De salientar que se de facto alterarmos as opções, e premirmos OK será imprimido na consola *true*, se não será imprimido *false*.

- setSelection(Selection, State) – Permite mudar o estado de uma determinada *Selection* para *true, false* ou *undecided.*

  Parâmetros:
  Selection – *Selection* a modificar o estado;
  State – O estado da *Selection* (*true*, *false*, ou *undecided*);

- areConditionsSatisfied(Element) – Testa se uma *Condition* para um determinado elemento DIDL é satisfeita. Retorna *true* se sim, retorna *false* se não.

  Parâmetros:
  Element – Representa o elemento DOM a verificar se a *Condition* é satisfeita;

Para testarmos as DIBOs setSelection e areConditionsSatisfied, poderemos criar o *Component* ilustrado na Figura 4.9. Se reparamos no código podemos observar que estamos a mudar o estado da *Selection* na instrução 2 e 4, assim chegamos à conclusão quando o código executar a instrução 6 de facto a primeira opção aparece como *false* e a segunda como *true* (Figura 4.10). Na última opção vamos testar se o *Item* que tem o id neyo (que tem a *Condition* com id neyo_video) é satisfeito se não alterarmos a primeira opção, isto é deixarmos como *false* ou mudarmos para *undecided* ela retornará *false*, mas se mudarmos o estado para *true* e premirmos OK ela retornará *true*.

```
<Component>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
		</Statement>
	</Descriptor>
	<Resource mimeType="application/mp21-method"><![CDATA[
		function teste_DIBOs_DII()
		{
		var selection=didDocument.getElementById("neyo_video");//1ª instrução
		DID.setSelection(selection,"false");//2ª instrução
		var selection=didDocument.getElementById("b");//3ª instrução
		DID.setSelection(selection,"true");//4ª instrução
		var choice=didDocument.getElementById("choice_01");//5ª instrução
		DID.configureChoice(choice); //6ª instrução
		var item=didDocument.getElementById("neyo");//7ª instrução
		return DID.areConditionsSatisfied(item); //8ª instrução
		}]]>
	</Resource>
</Component>
```

Figura 4.9 – DIM que permite testar as 3 DIBOs da DID.

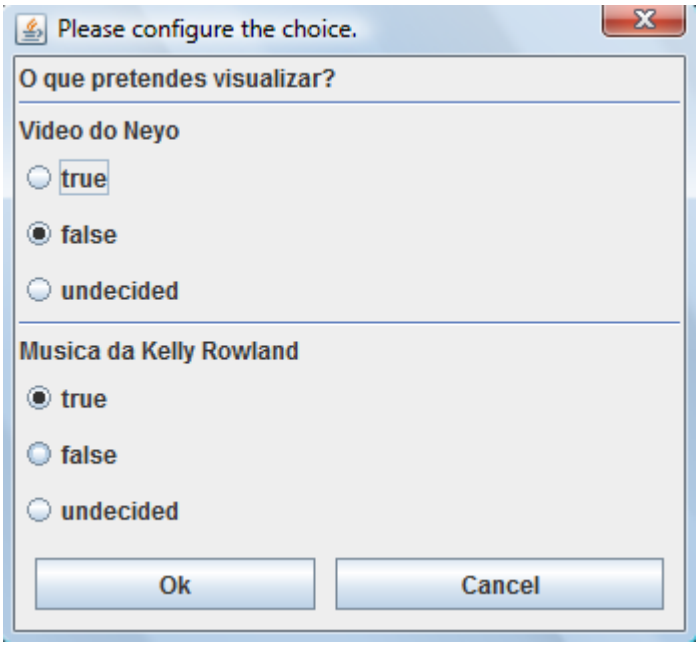


Figura 4.10 – Execução da configureChoice da DIM da figura 4.9.

4.2.3– DIBOs da DII (Digital Item Identification)

- getElementsByIdentifier(sourceDID,value) – Retorna da DID elementos baseados no DII *Identifier*.

  Parâmetros:
  sourceDID – Documento DOM do qual queremos obter o elemento;
  value – Valor do DII *Identifier;*

- getElementsByRelatedIdentifier(sourceDID,value) – Retorna da DID elementos baseados no DII *RelatedIdentifier*.

  Parâmetros:
  sourceDID – Documento DOM do qual queremos obter o elemento;
  value - Valor do DII *RelatedIdentifier;*

- getElementsByType(sourceDID,value) – Retorna da DID elementos baseados no DII *Type*.

  Parâmetros:
  sourceDID – Documento DOM do qual queremos obter o elemento;
  value – Valor do DII *Type;*

Estas 3 DIBOs são muito idênticas, como podemos ver na descrição de cada uma delas e nos seus argumentos. No entanto a forma como se declara na DID para depois na DIM aceder aos elementos é diferente. Se por exemplo quisermos obter elementos através do identificador, teríamos que declarar esta instrução no local da DID onde está o elemento que queremos obter por identificador, ou seja, passa a ser ter o mesmo pai do elemento que queremos obter. Isto é mostrado na figura 4.6 da DID em que vemos o identificador declarado na tag <dii:Identifier> para permitir aceder ao elemento onde se encontra o vídeo do neyo. Agora se quisermos acedê-lo, poderíamos criar uma DIM em que usávamos a instrução abaixo:

- DII.getElementsByIdentifier(didDocument,"urn:dii:example:video");

Depois disto poderíamos ter acesso ao vídeo e fazer o que pretendemos como ele com recurso a outras DIBOs ou DIXOs. Ainda neste tópico, mais à frente é dado um exemplo de uma DIM que usa uma destas DIBOs, juntamente com outras DIBOs de outros módulos para que o leitor possa perceber em que tipo de situações a sua utilização é útil. De realçar que podemos atribuir este identificador a mais de um elemento (getElements) para obtê-lo, tal como podemos ter na DID mais de dois identificadores diferentes, isto é, um identificador com um determinado nome é independente dos outros identificadores com um nome diferente no que consta aos elementos que ele tem associado a ele. O mesmo procedimento descrito para esta DIBO é aplicado para as outras duas deste tópico, à excepção de a tag na DID e da instrução da DIBO na DIM, ou seja, em vez de usarmos uma tag <dii:Identifier> usamos uma tag <dii:RelatedIdentifier> no caso da DIBO getElementsByRelatedIdentifier e

<dii:Type> no caso da DIBO getElementsByType. Para concluir a explicação relativa a este tópico podemos dizer que poderemos fazer uso destes 3 tipos de DIBOs na DID quantas vezes assim o pretendemos.

4.2.4- DIBOs da DIP (Digital Item Processing)

- alert(message, messageType) – Mostra uma mensagem ao utilizador de acordo com o seu tipo (Informativo, Erro, *Warning*).

  Parâmetros:
  message – Mensagem para o utilizador;
  messageType – Tipo de mensagem;

Esta DIBO é uma das que apresenta mais utilidade, não só porque permite avisar o utilizador do que o autor entender que é necessário informar, mas também porque permite ao autor, às vezes se tiver algum problema no seu código, saber em que sitio é que o seu código parou para o poder corrigir (funcionando como um println em Java). Relativamente ao teste desta DIBO é muito simples e já foi visto ao longo deste capítulo, mais especificamente na figura 4.3 e 4.4.

- execute(element) – Executa o recurso associado ao elemento *Component* ou *Descriptor*.

  Parâmetros
  element – Um elemento do tipo *Component* ou *Descriptor;*

A DIBO execute, é usada quando temos ficheiros executáveis, normalmente com a extensão .exe. A sua declaração na DID é semelhante à DIBO play e à DIBO print que serão explicadas mais adiante.

- getExternalData(mimeType, requestMessages) – Pede ao utilizador para seleccionar recursos externos do DI.

  Parâmetros:
  mimeType – *MimeType* do recurso (ex:audio/mpeg);
  requestMessages – Vector de *strings* dando a localização dos recursos;

Um exemplo de uma DIM usando esta DIBO está ilustrado na Figura 4.11 e o resultado da execução dessa DIBO na figura 4.12. A DIM da figura 4.11 quando executada apresenta uma janela *popup*, para o utilizador introduzir um endereço para que posteriormente possa reproduzir um vídeo ou uma musica. Sendo que o objectivo desta DIBO é ter acesso ao URL de um determinado recurso para que depois com base nessa informação o autor possa dar o seguimento que pretende.

```
<Component>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
		</Statement>
	</Descriptor>
	<Resource mimeType="application/mp21-method"><![CDATA[
		function getExternalData()
		{
		var mimeTypes = new Array(new Array("video/mpeg" , "audio/mpeg"));
		var requestMessages = ["Introduza um URL de um video ou uma musica em formato
mpeg."];
		return DIP.getExternalData(mimeTypes,requestMessages);
		}]]>
	</Resource>
</Component>
```

Figura 4.11 – Utilização da DIBO getExternalData numa DIM.

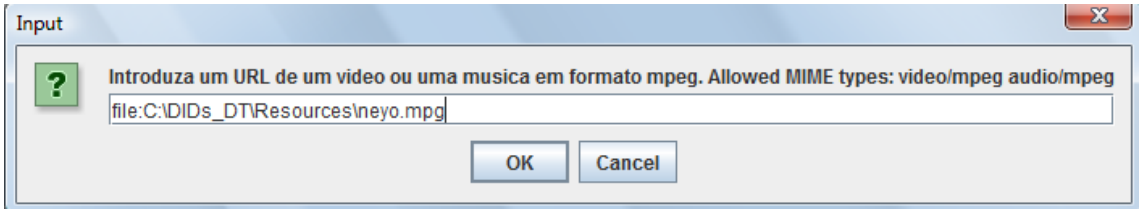


Figura 4.12 – Exemplo de execução da DIBO getExternalData da DIM da figura 4.11.

- getObjects(objectTypes, requestMessages) – Fornece ao utilizador uma ou mais selecções de um objecto com um dado *ObjecType*.

  Parâmetros:
  objectTypes – Vector de *ObjecTypes;*
  requestMessages – Vector de mensagens para o utilizador;

Um exemplo de uma DIM com o uso desta DIBO está ilustrado na figura 4.13 e o resultado da execução da DIBO na figura 4.14. Olhando para estas duas figuras facilmente percebemos o objectivo dela. Sendo assim podemos verificar que no caso desta DIM ela na primeira instrução começa por filtrar dois tipos de *ObjectTypes*, um do tipo "urn:foo:Movie" e outro do tipo "urn:foo:ALL". Se olharmos para a DID da figura 4.6 vemos que: o do tipo "urn:foo:Movie" tem como objectivo identificar na nossa DID os recursos de vídeo, como só temos um, só identifica o vídeo do neyo; o *ObjectType* do tipo "urn:foo:ALL" tem por objectivo identificar todos os recursos presentes na DID, que como se sabe só existem dois, o vídeo e a música. Sendo assim, continuando analisar a DIM podemos verificar que a primeira mensagem da instrução 2 está relacionada com o primeiro tipo de *ObjectType* ("urn:foo:Movie") da instrução 1, e a segunda mensagem da instrução 2 com o segundo tipo de *ObjectType* ("urn:foo:ALL"), também da instrução 1. Quando executarmos a 3ª instrução da DIM o que acontece, é que aparece a primeira mensagem e argumentos associados a ela para escolher (relacionados com o *ObjectType* do tipo "urn:foo:Movie"), como ilustra a figura 4.14 (mensagem da esquerda); após a escolha de uma opção (OK ou Cancel), aparece então a

segunda mensagem e argumentos associados a ela para o utilizador escolher (relacionados com o *ObjectType* do tipo "urn:foo:ALL"), como ilustra a figura 4.14 (mensagem da direita).

```
<Component>
        <Descriptor>
                <Statement mimeType="text/xml">
                        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
                </Statement>
        </Descriptor>
        <Resource mimeType="application/mp21-method"><![CDATA[
         function getObjects()
         {
                var objectTypes = ["urn:foo:Movie","urn:foo:ALL"];//instrução 1
                var requestMessages = ["Seleccione um filme.", "Seleccione um recurso."];//instr. 2
                return DIP.getObjects(objectTypes,requestMessages);// instrução 3
        }]]>
        </Resource>
</Component>
```

Figura 4.13 – Utilização da DIBO getObjects numa DIM.

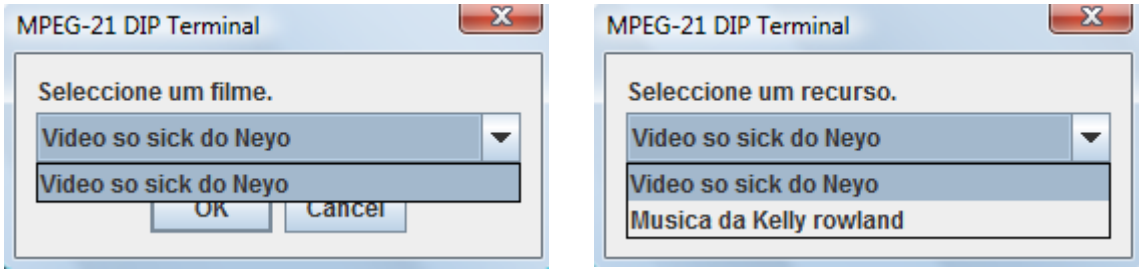


Figura 4.14 – Resultado da execução da DIBO getObjects da DIM da figura 4.13.

- getValues(dataTypes, requestMessages) – Pede ao utilizador para introduzir dados de um ou mais tipos de dados primitivos.

  Parâmetros:
  dataTypes – Tipos de dados (ex:String);
  requestMessages – Vector de mensagens para o utilizador, os quais são do tipo *Boolean*, *String* ou *Int*, correspondendo ao tipo de dado especificado no vector *dataTypes*;

Analisando esta DIBO com base na figura 4.15 podemos visualizar que duas perguntas são feitas ao utilizador, uma do tipo *String* e outra do tipo *Boolean*, como ilustra a figura 4.16. De salientar que o objectivo desta DIBO é o autor da DID ter acesso aos dados provenientes do utilizador para depois dar-lhe o devido processamento, com base no que ele pretende.

```
<Component>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dii:Identifier>urn:dii:example:component:dim_que_vai_ser_corrida_por_outr
		a_dim</dii:Identifier>
		</Statement>
	</Descriptor>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
		</Statement>
	</Descriptor>
	<Resource mimeType="application/mp21-method"><![CDATA[
		function getValues()
		{
		var dataTypes = ["String","Boolean"];
		var requestMessages = ["O que achas do DI?", "Queres ver ouvir uma Musica?"];
		var values=DIP.getValues(dataTypes,requestMessages);
		}]]>
	</Resource>
</Component>
```

Figura 4.15 - Utilização da DIBO getValues numa DIM.

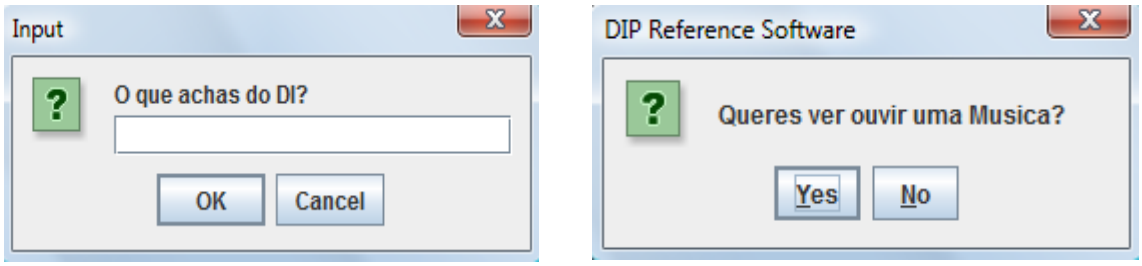


Figura 4.16 - Resultado da execução da DIBO getValues da DIM da figura 4.15.

- getObjectMap(Document) – Retorna o *ObjectMap* de um documento.

  Parâmetros:
  Document – Documento (DID);

A DIBO getObjectMap é usada juntamente com outras DIBOs, sendo que o seu objectivo é a criação do *ObjectMap* para manipular recursos *(Resources)* e *Descriptors* de acordo com o código da DIM. Ela será abordada na próxima DIBO, podendo então o leitor ver a nível prático para que serve o conceito do *ObjectMap*, *ObjectType* e os argumentos da DIM, os quais foram explicados no Capitulo 2.

- play(element,assync) – Reproduz um *Component* ou *Descriptor* específico.

  Parâmetros:
  element – *Component* ou *Descriptor* a ser reproduzido;

assync – Toma o valor *true* no caso de se pretender reproduzir o elemento dessincronizado, e toma *false* se for sincronizado;

A DIBO play é uma das mais úteis, pois permite reproduzir os recursos pretendidos. Na figura 4.17 vemos um exemplo da aplicação desta DIBO, a qual, ao contrário das outras DIMs analisadas, aceita um argumento do tipo urn:foo:ALL. Este argumento permite aceder ao *ObjecType* do tipo urn:foo:ALL e, como foi descrito anteriormente, este *ObjectType* tem como objectivo na nossa DID permitir aceder a todos os recursos. No momento em que passamos o argumento na DIM temos a possibilidade de aceder ao primeiro *Descriptor* de conteúdo textual que tem o mesmo pai desse *ObjectType*, uma vez que é o primeiro filho desse pai; se tivéssemos o *Component* que contém o recurso *(Resource)* como primeiro filho desse pai, ou alternativamente se remover o *Descriptor* de conteúdo textual, acederíamos então ao recurso *(Resource)*. Sendo assim, os argumentos que são apresentados ao utilizador estão na forma de conteúdo descritivo (por causa dos *Descriptors*) como podemos ver na Figura 4.18. No entanto, como queremos aceder a um dos recursos *(Resources)* de um dos *Descriptors*, temos que invocar a primeira instrução da DIM Play_1 da figura 4.17. Após este passo, tem-se acesso ao recurso, sendo então possível apresentá-lo, tal como está ilustrado na segunda instrução. Se quiséssemos fazer o *play* do *Descriptor* escolhido pelo utilizador, teríamos que apagar a primeira instrução, e alterar a segunda para DIP.play(arg1,false).

```
<Component>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:MethodInfo>
				<dip:Argument>urn:foo:ALL</dip:Argument>
			</dip:MethodInfo>
		</Statement>
	</Descriptor>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
		</Statement>
	</Descriptor>
	<Resource mimeType="application/mp21-method"><![CDATA[
		function Play_1( arg1 )
		{
			var cmp = arg1.getElementsByTagName("Component").item(0);//instr. 1
			DIP.play(cmp, false); //instrução 2
		}]]>
	</Resource>
</Component>
```

Figura 4.17 – Exemplo de uma DIM que aceita argumentos e que usa a DIBO play.

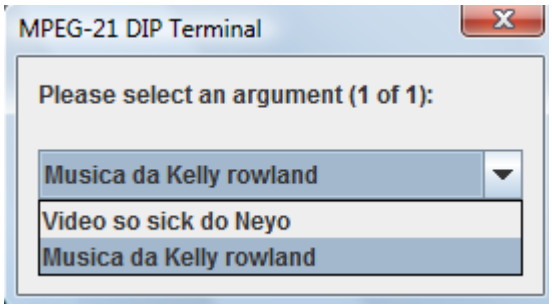


Figura 4.18 – Argumentos que são passados à DIM Play_1 da figura 4.17, e que permitem reproduzir os recursos associado ao *Descriptor* escolhido.

Outra maneira de fazer a reprodução de recursos, desta vez sem utilizar uma DIM com argumentos era, por exemplo, usar a DIBO getOjectMap dentro de uma DIM combinada com outras DIBOs para aceder a determinados recursos. Um exemplo de uma DIM desse tipo está ilustrada na figura 4.19, em que usamos na primeira instrução a DIBO getObjectMap para construir o *ObjectMap* da DID da figura 4.6; na segunda instrução acedemos ao *ObjectType* do tipo urn:foo:Movie com a DIBO getObjectsOfType, das DIBOs do *ObjectMap*, mais concretamente acedemos aos *Descriptors* e *Components* que tem o mesmo pai do *ObjectType* em questão; na 3ª instrução acedemos ao vídeo através do *Resource* do *Component,* que tem o mesmo pai que o *ObjectType* em questão; e na 4ª e última instrução fazemos o *play* do vídeo. Por isso quando é apresentada a lista de DIMs ao utilizador, se ele escolher a DIM Play_2 o *software* reproduz o vídeo. Na figura 4.19 encontramos 3 instruções que estão comentadas com duas barras, e por isso não são executas pelo *software*. Estas instruções permitem fazer a mesma coisa que as anteriores, ou seja reproduzir o vídeo só que em vez de usarmos o *ObjectMap*, usamos uma das DIBOs da DII. Para experimentar esta DIM com as ultimas 3 instruções basta retirar as duas barras, e comentar da mesma forma com duas barras as instruções acima ou então removê-las.

```
<Component>
        <Descriptor>
                <Statement mimeType="text/xml">
                        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
                </Statement>
        </Descriptor>
        <Resource mimeType="application/mp21-method"><![CDATA[
                function Play_2()
                {
                        var objectMap = DIP.getObjectMap(didDocument);
                        var info = objectMap.getObjectsOfType("urn:foo:Movie");
                        var cmp = info[0].getElementsByTagName("Component").item(0);
                        DIP.play(cmp,false);

        //var info = DII.getElementsByIdentifier(didDocument,"urn:dii:example:video");
        //var cmp = info[0].getElementsByTagName("Component").item(0);
        //DIP.play(cmp,false);
                }]]>
        </Resource>
</Component>
```

Figura 4.19 – Exemplo de uma DIM alternativa para fazer *play* de um determinado recurso.

- print(element) – Imprime um determinado elemento do tipo *Component* ou *Descriptor.*

  Parâmetros:
  element – *Component* ou *Descriptor* a ser imprimido;

A DIBO print permite imprimir documentos, imagens, texto, página *Web*, etc. Um exemplo de uso desta DIM era usar os exemplos que dei para a DIBO play e substituir só nas DIMs a DIBO play por print, mantendo o resto do procedimento. É evidente que isso só não chegava porque os recursos em questão não são de impressão, sendo assim teríamos obrigatoriamente alterar a DID da figura 4.6, nomeadamente os recursos, para documentos PDF ou Word. Depois de alterado podemos executar as DIMs e verificar que realmente imprimiu os recursos, apesar da DID não estar coerente em termos de conteúdo textual do que contém na DID *(Descriptors)* e nos *ObjectTypes* em relação aos novos recursos, mas isso pode-se alterar da mesma forma como se construiu a DID, o objectivo era só testar a DIBO print.

- release(playStatus) – Pára o *playback* de um *Component* ou recurso *(Resource)* e dá informação sobre esse mesmo estado de *playback.*

  Parâmetros:
  playStatus – Objecto associado ao elemento que está a ser reproduzido;

Esta DIBO não foi utilizada durante o desenvolvimento da dissertação, daí não fazer referência a ela.

- runDIM(itemIdType, itemId, componentIdType, componentId, arguments) – Permite mandar executar uma DIM dentro de uma DIM.

  Parâmetros:
  itemIdType – Indica o tipo de indentificador (DII *Identifier* ou URI) que é dado ao parâmetro itemId;
  itemId – Identifica o *Item* que contém a declaração DIM de uma DIM ou null;
  componentIdType – Indica o tipo de identificador (DII *Identifier* ou URI) que é dado ao parâmetro componentId;
  componentId – Identifica o *Component* que contém a declaração DIM de uma DIM ou null;
  arguments – Um vector de 0 ou mais objectos que são os argumentos a ser passados na invocação de uma DIM;

A DIBO runDIM permite executar outra DIM ou até a mesma. Um exemplo de instrução que poderíamos introduzir numa linha de código de uma determinada DIM, e que tem como objectivo executar a DIM da figura 4.15.

  - DIP.runDIM("dii","DIMs","dii","urn:dii:example:component:dim_que_vai_ser_corrida_por_outra_dim",new Array());

Analisando este exemplo de instrução da DIBO RunDIM vemos que usamos o dii, no 1º argumento da instrução, é usado para identificar o *Item* onde estão as DIMs, que no caso da DID da figura 4.6, é o *Item* com o id igual a "DIMs" (2ª argumento da instrução). No *Component* da figura 4.15 podemos ver o *Descriptor* a identificar a DIM que queremos executar e que é passado na instrução acima, no 4º argumento. Para concluir o 5º elemento serve para passar argumentos caso a DIM que queremos que a nossa DIM execute, aceite, como não é o caso, vai vazio.

- wait(timeInterval) – Espera um determinado tempo.

  Parâmetros:
  timeInterval – Número inteiro em ms;

A DIBO wait é uma DIBO muito simples e permite a quando da execução de uma DIM que tenha a instrução da DIBO no seu código, parar no local da instrução e aguardar o tempo definido no seu argumento. Em baixo encontra-se um exemplo da instrução que tem a DIBO wait, como podemos observar que ela tem como argumento o numero 20000, que corresponde a 20000 ms, ou seja, a instrução espera 20 segundos antes de passar à próxima instrução.

```
- DIP.wait(20000);
```

4.2.5– DIBOs da REL (Rights Expression Language)

- getLicense(Resource) – Permite obter a licença associada a um determinado recurso.

  Parâmetros:
  Resource – Representa o recurso a obter a licença;

- queryLicenseAuthorization(license,resource,rightNs, rightLocal, additionalInfo) – Permite verificar se tem autorização para aceder a um recurso.

  Parâmetros:
  license – Elemento que representa a licença;
  resource – Elemento que representa o recurso;
  rightNs – *Namespace* do direito a ser verificado ou null;
  rightLocal – Nome local do direito a ser verificado;
  additionalInfo – Vector de elementos que representa informação adicional a ser considerada;

Estas duas DIBOs estão relacionadas com os direitos e as permissões de um utilizador para aceder a recursos multimédia, através do uso de licenças. As licenças podem ser exprimidas através de uma linguagem própria designada de *Rights Expression Language* (REL). O fornecedor de um vídeo pode usar REL para transmitir as condições e/ou restrições que um consumidor deve observar ao interagir com o recurso multimédia por ele criado. No caso de existir uma licença associada a um recurso multimédia, isto é, no caso de o conteúdo ser

protegido, o consumidor antes de pagar para ter acesso ao recurso, pode obter a licença para se inteirar das condições com que poderá interagir com esse conteúdo. Exemplos de algumas condições que poderão existir na licença, expressas usando REL, incluem o número de vezes que o consumidor poderá reproduzir o(s) recurso(s); quais os recursos, de entre os que compõem o *Digital Item*, que estão abrangidos pela licença; o período de tempo durante o qual o consumidor pode interagir com o(s) recurso(s); etc.

4.2.6- DIBOs da DIP Error

- getDIPErrorCode() – Permite obter tipos de erros de código nas DIMs.

Esta DIBO permite identificar que tipos de erros ocorrem numa DIM, para isso ela retorna um número de um a oito, sendo que cada um desses números está associado a um determinando tipo de erro. Por exemplo se a DIBO retornar o numero 2, a que corresponde o erro INVALID_PARAM e que significa que foi passado um parâmetro errado a uma DIBO.

4.2.7- DIBOs do ObjectMap

- getArgumentList(index) – Retorna uma lista de tipos de argumentos.

  Parâmetros:
  index – É o número que indica o índice da *ArgumentList* no *ObjectMap*;

Esta DIBO permite aceder aos *Components* com DIMS, e verifica os tipos de argumentos (DIMs com a *tag* <dip:Argument> como a da figura 4.17), com base no índice que lhe é passado. Se o índice for 0 vai á primeira DIM que contém aquela *tag* e lista os seus argumentos, se o índice for 1 vai à 2ª DIM que tem aquela *tag* e lista os seus argumentos, e assim sucessivamente. Exemplo da aplicação desta DIBO que podem ser aplicadas a uma DIM, é dado através das instruções abaixo:

- var objectMap=DIP.getObjectMap(didDocument);
- return objectMap.getArgumentList(0);

- getArgumentListCount() – Retorna o número de listas de argumentos com argumentos em especifica ordem, que estão definidos no *ObjecMap*.

A DIBO conta o número de DIMs que aceitam argumentos que têm a *tag* <dip:Argument>, os argumentos repetidos não entram para os cálculos. Exemplo de instruções:

- var objectMap=DIP.getObjectMap(didDocument);
- return objectMap.getArgumentListCount();

- getMethodCount(argumentList) – Retorna o número de DIMs que estão definidas para aceitar como parâmetros os argumentos listados em *ArgumentList*.

Parâmetros:
argumentList – Um vector de strings que indica os tipos de argumentos;

Esta DIBO e as duas de baixo funcionam de maneira semelhante às duas DIBOs descritas acima.

- getMethodWithArgs(argumentList) – Retorna um vector de objectos representando a declaração da DIM para DIMs que aceitam como parâmetros os argumentos listados na *ArgumentList*.

  Parâmetros:
  argumentList - Um vector de strings que indica os tipos de argumentos;

- getMethodWithArgs(argumentList, index) - Retorna um objecto representando a declaração da DIM para DIMs que aceitam como parâmetros os argumentos listados na *ArgumentList*.

  Parâmetros:
  argumentList - Um vector de strings que indica os tipos de argumentos;
  index – Número que indica o índice da DIM na lista de DIMs que aceitam como parâmetros os argumentos listados no *ArgumentNames;*

- getObjectType(typeName, index) – Retorna um objecto de um dado *ObjectType*.

  Parâmetros:
  typeName – Nome do *ObjectType;*
  index – Número que indica o indice do objecto na lista de objectos de um dado *ObjectType*;

Esta DIBO ao contrário da DIBO getObjectsOfType só permite aceder a um objecto de um dado *ObjecType*, isto se acedermos ao *ObjecType* do tipo urn:foo:ALL da DID da figura 4.6 podemos ter acesso a um *Descriptor* e a um *Component* do vídeo, e a um *Descriptor* e um *Component* da musica, mas como esta DIBO tem que se especificar um índice só podemos ter acesso a um objecto, ou seja, ao *Descriptor* e *Component* do vídeo (index=0) ou ao *Descriptor* e *Component* da musica (index=1). Em baixo encontra-se duas instruções que podemos colocar dentro de uma DIM para termos acesso à música (através da variável "musica"), em que a segunda instrução recorre a esta DIBO.

- var objectMap=DIP.getObjectMap(didDocument);
- var musica = objectMap.getObjectOfType("urn:foo:ALL",1);

- getObjectsOfType(typeName) - Retorna um vector de objectos de um dado *ObjectType*.

  Parâmetros:

typeName - Nome do *ObjectType*;

O uso desta DIBO já foi mencionado na DIBO play com recurso a exemplos, em que vimos a sua importância.

- getObjectsOfTypeCount(typeName) – Retorna o número de objectos que estão definidos no *ObjectMap* para um dado *ObjectType*.

  Parâmetros:
  typeName - Nome do *ObjectType;*

Esta DIBO permite saber quantos objectos existe de um dado *ObjectType,* tendo em conta a DID da figura 4.6 esta DIBO retornará 1 para o *ObjectType* dos tipos urn:foo:Movie e urn:foo:Sound, e 2 para o *ObjectType* do tipo urn:foo:ALL. As duas instruções em baixo permitem testar esta DIBO, obtendo o resultado 1 porque apenas se declarou uma vez o *ObjectType* do tipo urn:foo:Movie na DID.

- var objectMap=DIP.getObjectMap(didDocument);
- return objectMap.getObjectsOfTypeCount("urn:foo:Movie");

- getObjectTypeCount() – Retorna o número de *ObjectTypes* que estão definidos no *ObjectMap*.

Esta DIBO por sua conta o numero de *ObjectTypes*, que no caso da DID da figura 4.6 são 3, um do tipo urn:foo:Movie, outro do tipo urn:foo:Sound e outro do tipo urn:foo:ALL. Para testar-mos esta DIBO usávamos as duas instruções abaixo:

- var objectMap=DIP.getObjectMap(didDocument);
- return objectMap.getObjectTypeCount();

- getObjectTypeName(index) – Retorna o nome de um *ObjectType*.

  Parâmetros:
  index – Número que indica o índice do *ObjectType* no *ObjectMap;*

A DIBO retorna o nome do *ObjectType* de acordo com a sua ordem no *ObjectMap*, se por exemplo atribuirmos o índice 0 à DIBO ela retorna o *ObjectType* do tipo urn:foo:ALL, permitindo assim aceder a esse *ObjectType.*

4.2.8– DIBOs do PlayStatus

- getStatus() – Permite obter um estado de um recurso quanto à sua reprodução.

Por fim a DIBO getStatus retorna um número de O a 2. Cada um desses números representa o estado de um determinado recurso associado com o objecto *PlayStatus*. Se, por

exemplo, retornar o numero 0, segundo a norma, corresponde ao estado RELEASED, que significa que um determinado recurso não está a ser actualmente reproduzido.

4.3 – Utilização de DIXOs

Este tópico aborda como declarar uma DIXO na DID para que seja possível executá-la. A implementação de DIXOs será vista no capítulo 7.

Uma forma de declararmos uma DIXO numa DID está representada na figura 4.20. Para entender a declaração da DIXO na DID da figura 4.20, tem se ter a noção de que é um ficheiro com a extensão .class. O ficheiro class de uma DIXO não é mais que um ficheiro com o nome da DIXO com a extensão .class que é gerado no directório bin do *workspace* da aplicação em causa, sendo esse ficheiro que tem de ser declarado na DID, isto é, se criamos a DIXO validate.java o ficheiro que tem de ser declarado na DID tem o nome de validate.class, se for o caso de invocarmos a DIXO através de classes, como ilustra a figura 4.20.

```
<Component>
    <Descriptor>
        <Statement mimeType="text/xml">
            <dii:Identifier>urn:dii:example:item:jdixo01</dii:Identifier>
        </Statement>
    </Descriptor>
    <Descriptor>
        <Statement mimeType="text/xml">
            <dip:JDIXOClasses>                                  Declaração da class
                <dip:Class>Validate</dip:Class>
            </dip:JDIXOClasses>
        </Statement>
    </Descriptor>
    <Descriptor>
        <Statement mimeType="text/xml">
            <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIXO:Java</dip:Label>
        </Statement>
    </Descriptor>
    <Resource mimeType="application/java" ref="Validate.class"/>
</Component>
```

Figura 4.20 – Declaração de uma DIXO numa DID com base no ficheiro com a extensão .class.

Se a invocação da DIXO for através de um jar, é recomendado que o nome para o jar deva ser dado pelo programador com o mesmo nome da DIXO desenvolvida em Java (validate.java) ou seja validate.jar (Figura 4.21), porque se o programador não der o nome e não especificar na <dip:Class> o nome da sua DIXO desenvolvida em Java vai dar erro a aceder à classe deste jar uma vez que ela tem um nome fixo validate.class, a qual é diferente na especificada na <dip:Class>, a não ser que ele especifique a validate.class no sitio correcto da DID (na <dip:Class>) e aí já pode dar outro nome ao jar. Para obter o a execução do jar ou do ficheiro com extensão class com sucesso dentro de uma DIM tem de a declarar a DIBO DIP.runJDIXO como ilustra o figura 4.22 com o 4º argumento da DIBO com o nome da DIXO que neste caso designa-se Validate.

```
<Component>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dii:Identifier>urn:dii:example:item:jdixo02</dii:Identifier>
		</Statement>
	</Descriptor>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:JDIXOClasses>
				<dip:Class>Validate</dip:Class>
			</dip:JDIXOClasses>
		</Statement>
	</Descriptor>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIXO:Java</dip:Label>
		</Statement>
	</Descriptor>
	<Resource mimeType="application/java-archive" ref="Validate.jar"/>
</Component>
```

Figura 4.21 – Declaração de uma DIXO numa DID com base no ficheiro com a extensão .jar.

```
<Component>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:MethodInfo autoRun="false" />
		</Statement>
	</Descriptor>
	<Descriptor>
		<Statement mimeType="text/xml">
			<dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
		</Statement>
	</Descriptor>
	<Resource mimeType="application/mp21-method"><![CDATA[
		function Validate_DI()
		{
	DIP.runJDIXO("dii","urn:dii:example:item:jdixos","dii","urn:dii:example:item:jdixo01","Validate",
new Array());
		}]]>
	</Resource>
</Component>
```

Figura 4.22 – Exemplo da declaração da DIBO RunJDIXO numa DIM.

4.4 – Conclusão

Neste capítulo foi explicado o *software* de referência da norma MPEG-21 para DIP, em termos de sua arquitectura e funcionalidades, para que no próximo capitulo que aborda a aplicação desenvolvida no âmbito desta dissertação, se perceba a sua implementação e a ligação que faz com os diferentes módulos do *software*. Neste capítulo constatou-se que se pode combinar diferentes DIBOs e fazer coisas interessantes a nível da interacção do utilizador com o *Digital Item*. Também se chegou à conclusão que existe várias maneiras de chegar ao mesmo resultado, havendo assim alguma redundância no que respeita a DIBOs. No entanto, cada uma tem a sua importância para a qual foi desenvolvida. Para concluir ainda

foi visto como se declara DIXOs numa DID, para que no capítulo 7 se possa abordar a sua implementação.

No próximo capítulo será abordada a aplicação desenvolvida no âmbito desta dissertação, a qual se designa de MPEG21 DT (MPEG21 DIP Teach).

Capítulo 5

MPEG21 DIP Teach

Neste capítulo será descrito em termos de funcionalidades, requisitos e arquitectura a aplicação desenvolvida no âmbito desta dissertação, para consumo de *Digital Items* utilizando *Digital Item Processing* ou através de outras formas, também desenvolvidas. A aplicação foi desenvolvida em Java e é designada de MPEG21 DIP Teach, tendo como cenário de aplicação o ensino à distância. O *software* referência da norma MPEG-21 como foi referido no capítulo anterior, serviu de base para o desenvolvimento da MPEG21 DIP Teach. Contudo ele apresenta algumas limitações, que tiveram que ser corrigidas, e em alguns casos também foi necessário implementar no *software*, respeitando contudo a norma MPEG-21. Além disso foi criado novo código (novos módulos) para satisfazer certas necessidades que uma aplicação deste tipo requer e que de alguma forma se relacionam com o *software* de referência da norma. Sendo assim não é objectivo deste capitulo explicar o *software* de referência novamente, mas sim explicar e justificar as suas modificações, assim como fazer uma análise dos novos módulos criados.

O capítulo está estruturado da seguinte forma: 5.1 – Análise da MPEG21 DIP Teach; 5.1.1 – Arquitectura da MPEG21 DIP Teach; 5.2 – Análise das funcionalidades e implementação dos módulos da MPEG21 DT; 5.3 – Conclusão.

5.1 – Análise da MPEG21 DIP Teach

A aplicação desenvolvida e que está ilustrada na figura 5.1, está dividida em módulos que permitem controlar determinada parte da interface. Sendo assim existem 5 principais módulos e que estão listados abaixo:

- Módulo MPEG21DT – representa as acções dos componentes da interface, mais propriamente do lado direito da interface, e que é responsável pela interacção pelos diversos módulos, nomeadamente o DIDEngine2 e a ligação com o *software* de referência, e pela criação da própria interface.

- Módulo controlMPEG21DT – representa o controlo da aplicação no que diz respeito à organização dela e que está associada à barra de menus.
- Módulo DIDEngine2 – módulo responsável por navegação e processamento da DID mostrando o resultado do seu processamento nos respectivos componentes da interface.
- Módulo DIDTreeView – permite visualizar o DI em forma de árvore, mostrando ao utilizador a forma como está hierarquizado.
- Módulo do Software de Referência da norma MPEG-21 – representa as modificações feitas para um funcionamento correcto da aplicação.

De salientar que o módulo DIDEngine2 designa-se por esse nome, devido ao facto de, tal como a DIDEngine do *software* de referência da norma MPEG-21, este módulo ser responsável por interpretar a DID, sendo que apresenta a sua própria interpretação, de acordo com o que vai ser descrito mais à frente sobre este módulo.

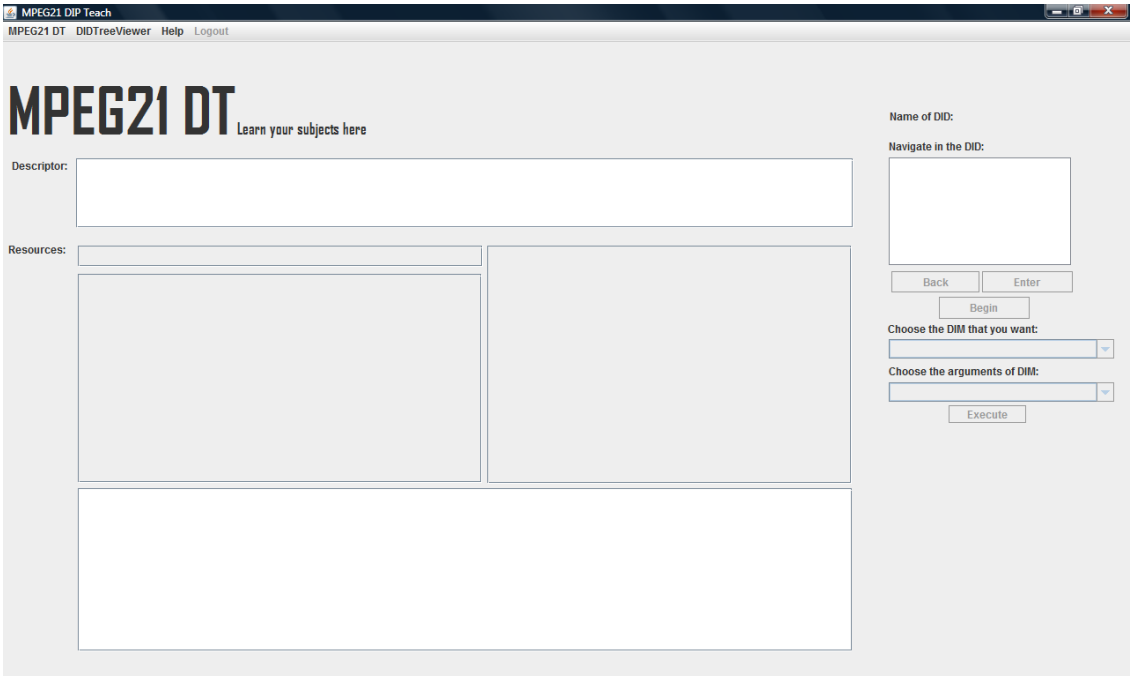


Figura 5.1 – Interface da aplicação MPEG21 DIP Teach.

A aplicação MPEG21 DIP Teach tem duas formas de processamento que actuam em separado, mas ao mesmo tempo no consumo de um *Digital Item*. Estas suas formas de processamento ao mesmo tempo no consumo de *Digital Items*, permite ao utilizador usufruir duma maior variedade de funcionalidades, do que fosse só uma a fornecer. Sendo assim os benefícios para aplicação destas formas de processamento são explicados em separado, como é referido em baixo:

- **Vantagens da aplicação no consumo dos DIs sem recorrer à DIP:** O utilizador tem noção da forma como o DI se encontra estruturado, ou seja, o utilizador pode navegar no DI segundo a sua estrutura hierárquica, podendo ver assim a maior parte dos seus conteúdos à medida que navega. O utilizador apenas navega para onde pretende, de acordo com a informação que lhe é apresentada, além de ter a possibilidade de retroceder ou voltar ao início. Este processo é semelhante por exemplo a um menu do telemóvel, em que o utilizador acede ao topo do DI (ao menu do telemóvel) e a partir daí vai acedendo aos elementos que pretende, através do conteúdo que cada elemento tem para lhe fornecer (da mesma forma como navega no telemóvel a partir do menu). Esta forma de visualização é mais intuitiva e flexível para o utilizador, contudo tem algumas limitações as quais a DIP resolve. Uma dessas limitações, é o facto de existirem partes no DI que o utilizador não tem acesso, devido ao facto da aplicação nesta forma de processamento ter definido as suas próprias regras no que respeita à visualização do DI.

- **Vantagens da aplicação no consumo dos DIs com recurso à DIP:** Existe a possibilidade de acrescentar funcionalidades por parte da interacção do utilizador com o DI, através da definição de métodos; várias formas de visualização dos *Digital Items*; controlar a forma como os conteúdos são visualizados; colocar licenças sobre os recursos; aceder directamente a determinado conteúdo; processamento de *Choices*, etc. Ou seja as regras que a DIP põe na interpretação de um DI são "universais", resolvendo assim o problema da interoperabilidade.

5.1.1– Arquitectura da MPEG21 DIP Teach

Com base no que foi referido em cada módulo, e com base na arquitectura e explicação do *software* de referência da norma que foi vista no capítulo anterior, é ilustrada na figura 5.2, a arquitectura da MPEG21 DIP Teach.

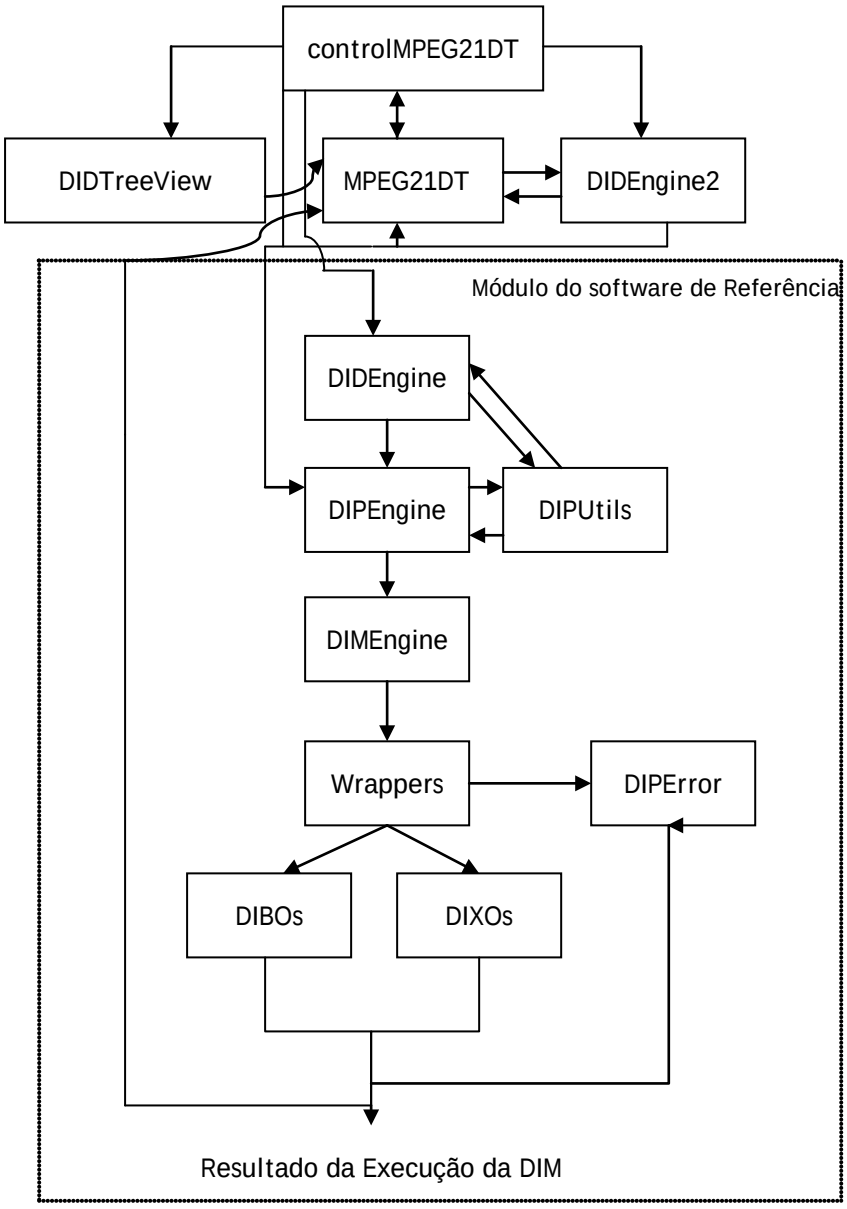


Figura 5.2 – Arquitectura da MPEG21 DIP Teach.

Analisando a figura 5.2 podemos ver que o módulo principal, a MPEG21DT acede à DIDEngine2 por causa da navegação na DID e esta última retorna-lhe os resultados pedidos; acede ao módulo controlMPEG21DT para gerá-lo na sua interface, uma vez que a criação desse módulo é da responsabilidade dele próprio; e por fim acede à DIPEngine para poder executar a DIP, sendo que este também lhe acede para actualizar os estados das variáveis que permitem executar a DIP.

O módulo DIDEngine2 retorna os seus resultados de processamento ao MPEG21DT, mas também pode aceder à DIPEngine no caso de os *Components* passarem a ser controlados eventualmente por uma DIM.

O módulo controlMPEG21DT é responsável pelo controlo da MPEG21DT. Para isso acede-lhe no que diz respeita à sua gestão, por exemplo, para a remoção de recursos; acede à DIDEngine2, à DIDEngine, e à DIPEngine, quando da inicialização destes módulos, mais concretamente na altura do carregamento do *Digital Item*; e à DIDTreeView para gerar a árvore do DI, bem como controlar se a árvore do DI aparece ou não na interface.

O módulo DIDTreeView acede à MPEG21DT para apresentar a vista hierárquica do *Digital Item*, quando o módulo controlMPEG21DT assim o invocar.

Relativamente à arquitectura do *software* de referência, além do que foi explicado no capítulo 4, à que salientar que o resultado de uma execução de uma DIM poderá ter como destino apresentar ou controlar alguma coisa na interface, e o módulo DIPUtils só acedido pela DIPEngine quando é para mostrar os argumentos de um DIM definida como automática.

5.2 – Análise das funcionalidades e implementação dos módulos da MPEG21 DT

Módulo DIDEngine2

O módulo DIDEngine2 foi desenvolvido como todos os outros, em Java, mas com auxílio a uma linguagem própria de XML fornecida por uma API designada JDOM [33] e que permite interpretar o XML, ou seja, a DID. A opção por esta API em deteriorimento da API DOM que é utilizada pelo *software* de referência da norma, recaiu devido à sua simplicidade de programação comparativamente ao DOM, e ao facto de não apresentar as limitações que o DOM apresentava.

O módulo é responsável por mostrar na interface os elementos da DID e o seu conteúdo associado, à medida que o utilizador os percorre carregando nos respectivos elementos, além de permitir ao utilizador com o auxilio do módulo MPEG21DT, a possibilidade de retroceder, avançar e voltar ao inicio de como se páginas de HTML se tratasse. A filosofia de navegação no DI por parte deste módulo é realizada com base na estrutura hierárquica do *Digital Item*.

O seguinte diagrama dá um exemplo de uma estrutura hierárquica de um *Digital Item*:

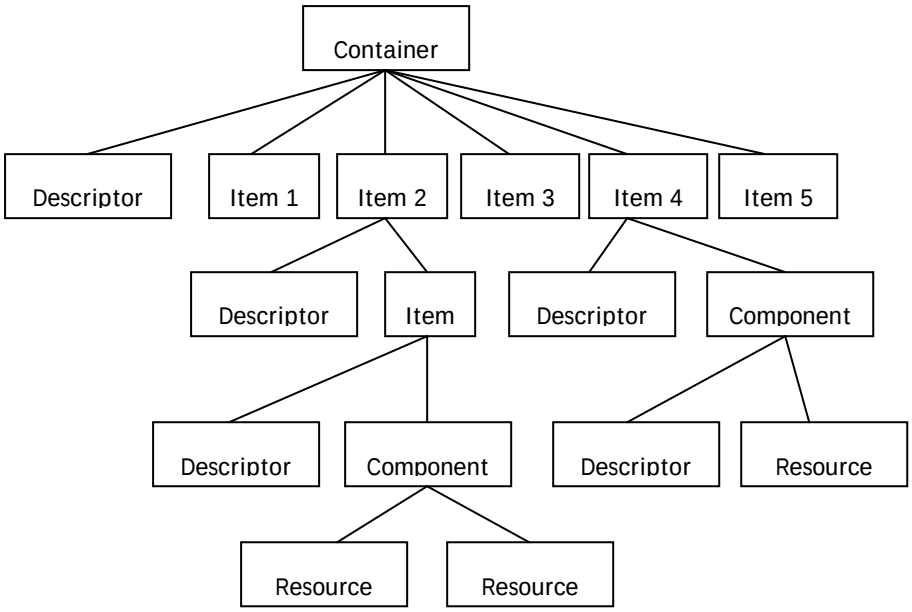


Figura 5.3 – Exemplo de uma estrutura hierárquica de um *Digital Item*.

O módulo DIDEngine2 fornece à aplicação a seguinte filosofia: a navegação do *Digital Item* é feita do topo do elemento da DIDL para baixo, ou seja, o utilizador pode navegar no DI carregando nos elementos que pretende aceder, de acordo com sua a estrutura hierárquica. Observando a estrutura hierárquica da figura 5.3, se um utilizador aceder ao elemento *Container*, é visualizada a informação do elemento, e são listados na lista da interface os filhos do elemento *Container*, que o utilizador escolheu, que no caso da figura 5.3, lista os elementos: *Item 1; Item 2;Item 3; Item 4; Item 5*. Se o utilizador pretender aceder a um desses *Items*, o procedimento é o mesmo, e assim sucessivamente. De salientar que na lista só aparecem os elementos *Container* e *Item*, e por isso os *Descriptors* e os *Components* que contêm os *Resources*, não aparecem na lista, sendo reproduzidos automaticamente, fazendo assim do *Item* como sendo a unidade de navegação no DI.

Os componentes da interface que o módulo DIDEngine2 utiliza para inserir o conteúdo de acordo com a informação proveniente da DID estão ilustrados na figura 5.4, identificados com um número.

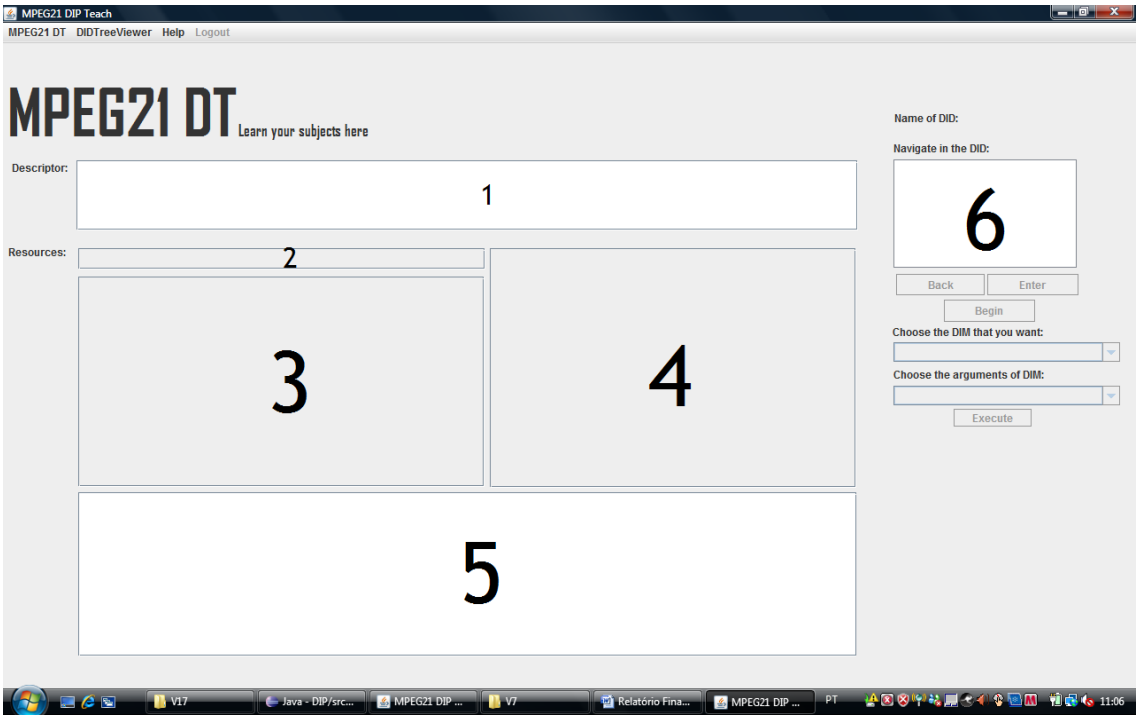


Figura 5.4 – Numeração dos componentes da aplicação, aos quais o módulo DIDEngine2 tem acesso.

Componentes da interface que a DIDEngine2 utiliza para apresentar os seus resultados:

- **Componente 1** – Painel que serve para apresentar texto contido nos *Descriptors* com o *mimeType* text/plain.
- **Componente 2** – Painel que serve para apresentar áudio de acordo com sua localização contida num *Resource*.
- **Componente 3** – Apresenta a imagem de acordo com sua a localização contida num *Resource*.
- **Componente 4** – Apresenta vídeo de acordo com a localização contida num *Resource*.
- **Componente 5** – Apresenta texto contido num *Resource* com um *mimeType* do tipo text/plain.
- **Componente 6** – Lista, onde os elementos da DID são apresentados ao utilizador para ele escolher.

O módulo DIDEngine2 é um módulo essencialmente criado para ser acedido por outros módulos através das funções que estão nele implementadas, mais concretamente pelo módulo controlMPEP21DT quando na altura do carregamento do DI e pelo módulo MPEG21DT no que diz respeito á navegação na DID (botões ***Back, Begin*** e ***Enter*** e a lista acima deles como mostra a figura 5.4). As funções que o módulo suporta e a sua explicação vão ser descritas abaixo:

- Analisa DID – Faz o *parsing* da DID; verifica se o *Digital Item* é válido, se sim, lista na interface os filhos do elemento de topo do DI (DIDL), se não retorna ao utilizador que o DI não validado, bloqueando a interface.

- Analisa Elementos – Mostra determinados filhos de um elemento na lista da figura 5.4 (Componente 6). Em baixo encontra-se o procedimento caso ela encontre como filho um elemento com o nome:

  - *Container* – Verifica se ele tem atributo, um id, se tiver adiciona na lista o valor desse atributo, se não tiver adiciona o elemento á lista, neste caso adiciona *Container*.

  - *Item* – Mesmo procedimento que o *Container*, só que em vez de *Container* é *Item.*

  - *Descriptor* – Reproduz o *Descriptor* na parte correspondente a ele na interface, não adicionando assim à lista.

  - *Component* – Reproduz o *Component* se contiver como filhos *Descriptors* ou *Resources*, e imprime na lista "A reproduzir resource…", isto se o processamento do *Component* tiver como modo *default*, ou seja, se não existir na DID nenhuma DIM chamada DIXO_Component, porque se existir ele executa o *Component* através daquela DIM. De salientar que os *Resources* são reproduzido com base no seu *mimeType* [43]. Na tabela 5.1 são apresentados os *mimeTypes* que a aplicação MPEG21 DIP Teach processa.

mimeTypes	Modo de Abrir
video/mpeg	Abre no Componente 4 da figura 5.4 através do player da Java Media Framework
audio/mpeg	Abre no Componente 2 da figura 5.4 através do player da Java Media Framework
video/x-msvideo	Abre no Componente 4 da figura 5.4 através do player da Java Media Framework (formato .avi)
application/pdf	Abre através dum player pdf desenvolvido em java
application/vnd.ms-powerpoint	Abre um powerpoint com o programa apropriado para este tipo de documentos, definido como default no computador do cliente.
application/ms-word	Abre um documento word com o programa apropriado para este tipo de documentos, definido como default no computador do cliente.
application/rtf	Abre um documento com a extensão rtf com o programa apropriado para este tipo de documentos, definido como default no computador do cliente
application/x-gzip	Abre a aplicação com o programa apropriado para este tipo de aplicações, definido como default no computador do cliente
text/html	Abre a página web com o Internet Explorer
application/x-shockwave-flash	Abre a aplicação com programa apropriado para este tipo de aplicações, definido como default no computador do cliente. (formato .swf)
audio/x-wav	Abre no Componente 2 da figura 5.4 através do player da Java Media Framework (formato .wav)
image/bmp	Abre no Componente 3 da figura 5.4
image/gif	Abre no Componente 3 da figura 5.4
image/jpeg	Abre no Componente 3 da figura 5.4
image/tiff	Abre no Componente 3 da figura 5.4
video/quicktime	Abre no Componente 4 da figura 5.4 através do player da Java Media Framework
text/plain	Abre no Componente 5 da figura 5.4

Tabela 5.1 – Lista de *mimeTypes* processados pela aplicação MPEG21 DIP Teach.

Relativamente à reprodução de recursos, à que salientar que o *Java Media Framework* [38], é uma API a qual se baseia num *player* de vídeo e áudio desenvolvido em Java pela Java Sun [32]. Foi criado um módulo chamado PlayerResources com código para a criação do *player* de áudio no Componente 2 e para a criação do *player* de vídeo no componente 4. A opção por esta ferramenta em vez do *player* que o *software* de referência da norma tem já incluído e adaptado a funcionar no seu código, deve-se ao facto do *Java Media Framework* ter um maior leque de *codecs* de áudio e vídeo em comparação com o outro *player* que só reproduz mov, mp3 e mp4. Além disso podemos aumentar a biblioteca de *codecs* deste *player* instalando na máquina o ffmpeg [48].

Entre as API para pdf existem uma da Adobe e uma da Java Sun para apresentar pdfs em Java. A da Adobe é a que apresenta melhor performance em termos de visualização, no entanto apresenta algumas noances que limitam o seu uso, de entre as quais destacam-se: não é compatível com todos os sistemas operativos e apresenta um leque de várias opções para utilizador, as quais não tem interesse como a procura de ficheiros pdfs, a impressão, uma vez que o autor do DI pode querer que o documento não possa ser impresso, se tal for necessário o autor tem as licenças ou a utilização das DIBOs para a autorização dessa impressão. Sendo assim foi escolhida a API PDF Renderer [42] da Java Sun, que permite visualizar os pdfs, e não tem os constrangimentos da outra API.

Nota: Para que se possa usufruir do *player* de áudio e vídeo tem que se instalar a aplicação da *Java Media Framework* disponível no site da Java Sun [32].

A figura seguinte ilustra um fluxograma resumido da DIDEngine2 e a interacção com os outros módulos:

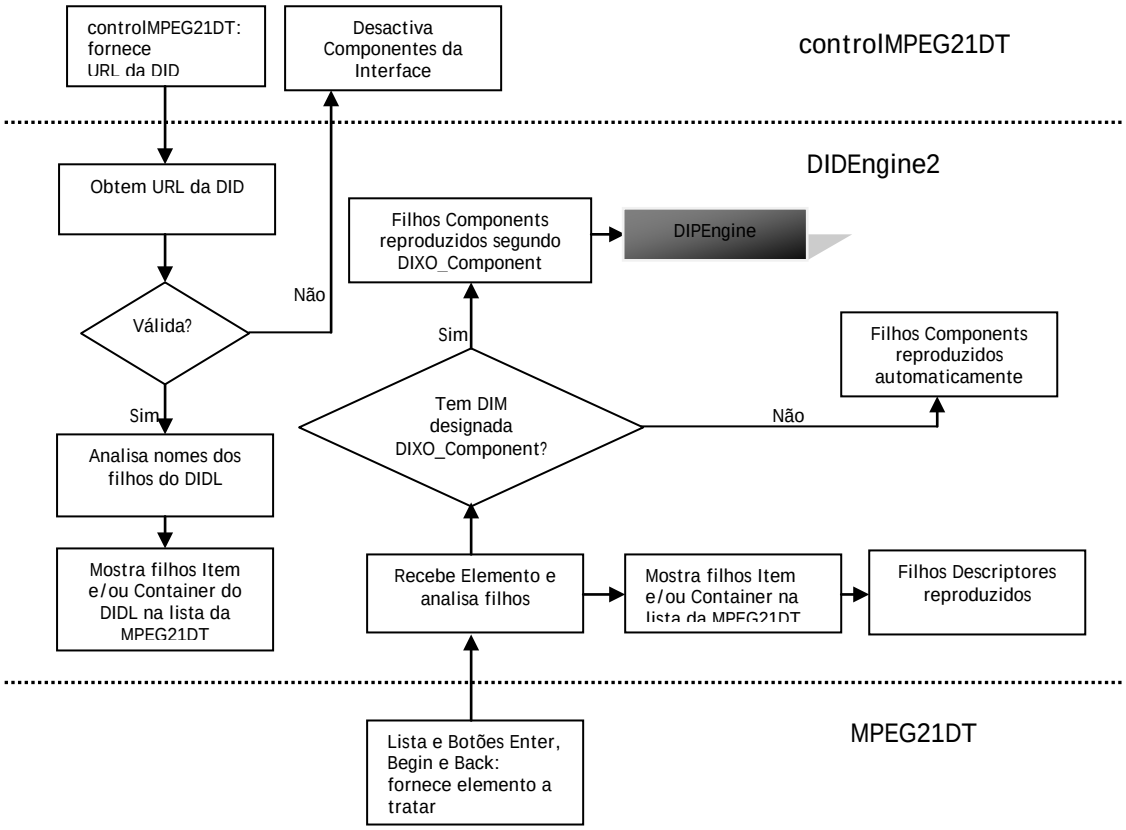


Figura 5.5 – Fluxograma resumido do módulo DIDEngine2.

Módulo MPEG21DT

O módulo MPEG21DT é o módulo responsável por gerar a interface da MPEG21 DIP Teach, bem como comunicar com os outros módulos através das acções dos seus componentes, nomeadamente com o DIDEngine2 e com o DIPEngine do *software* de referência da norma

MPEG-21. A figura 5.6 representa uma parte dos componentes da interface MPEG21 DIP Teach. De salientar que esses componentes encontram-se desactivados quando do arranque da aplicação bem como todos os *items* do menu DIDTreeViewer que serão vistos no módulo controlMPEG21DT.

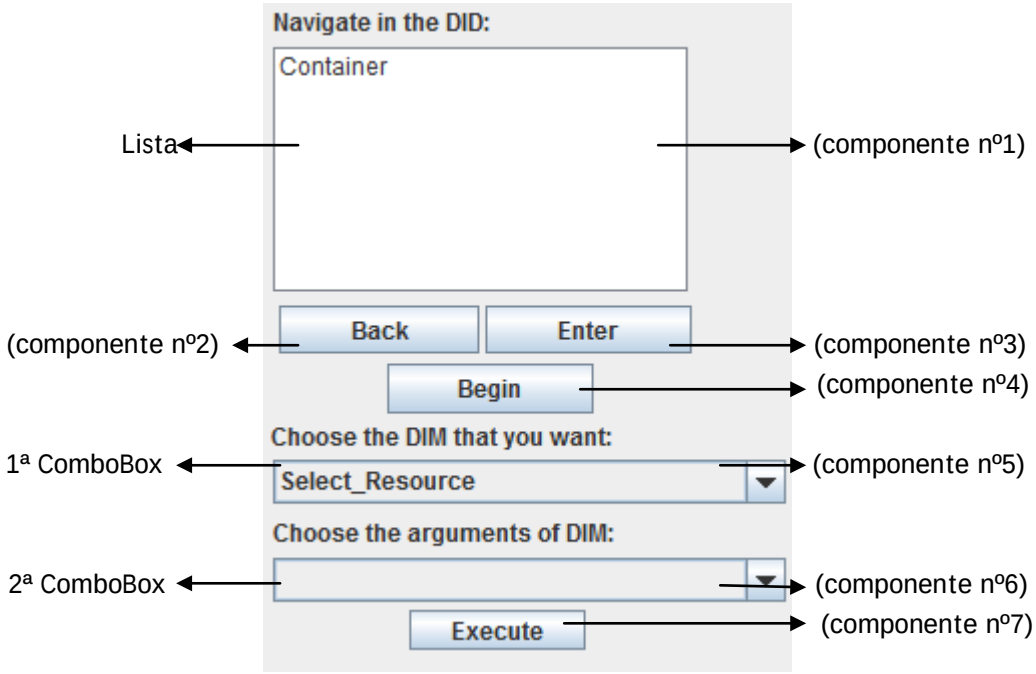


Figura 5.6 – Componentes com acções do Módulo MPEG21DT.

Analisando a figura 5.6 observamos 7 componentes, 4 dos quais tem ligação à DIDEngine2, se bem que também podem ser controlados por DIXOs como vamos poder ver mais adiante (lista e botões ***Enter, Begin*** e ***Back***). E os restantes 3 fazem ligação ao DIPEngine do *software* de referência da norma (Componentes nº 5 e 6, e botão ***Execute***).

Sendo assim vamos começar a analisar as funcionalidades dos primeiros 4 componentes e que são responsáveis por navegar na DID como se de páginas HTML se tratasse fornecendo ao utilizador opções de retroceder, voltar ao inicio e avançar. Quando se carrega num dos 4 primeiros componentes, se a variável DIP estiver inactiva, é a DIDEngine2 que controla esse botão, se ela tiver activa então é uma DIXO que controla. No modo *default* ela está sempre a inactiva, isto é quando se faz o carregamento de um DI. Os componentes só são controlados por DIXOs quando existir num DI uma DIM que é executada e tem por missão controlar esses componentes mudando a variável DIP para o estado activo.

Funcionamento dos componentes 1 a 4, através da DIDEngine2 (DIP na estado desactivo):

- Botão ***Enter*** e Lista – A lista e que o botão ***Enter*** tem as mesmas funções associadas. Quando o utilizador escolhe o elemento que pretende aceder e acciona um dos botões: 1º - todos os recursos que estão em reprodução na interface são removidos e a lista é limpa para receber os novos elementos; e 2º - analisa os filhos do elemento escolhido através da função Analisa Elementos vista no módulo DIDEngine2.

- Botão ***Begin*** – Ao carregar neste botão: 1º - apaga todos os recursos que estão a ser reproduzidos e limpa a lista; 2º - Volta ao estado inicial, passando o elemento de topo do DI (DIDL) através da função Analisa Elementos, vista no módulo DIDEngine2.

- Botão ***Back*** - Ao carregar neste botão: 1º apaga todos os recursos que estão a ser reproduzidos e limpa a lista; 2º acede ao elemento avô do elemento actual, passando através da função Analisa Elementos, vista no módulo DIDEngine2. O elemento actual é o último filho que pode estar representado ou não na lista, e por isso aceder ao elemento avô dele para retroceder, porque se acedesse ao elemento pai, a interface listava os mesmos filhos que estão listados actualmente.

Funcionamento dos componentes 1 a 4, através da DIPEngine do software de referência (DIP no estado activo):

- Botões ***Enter, Begin, Back*** e lista:

Estes 4 componentes quando controlados por DIXOs apresentam o mesmo procedimento em termos de código do módulo MPEG21DT, à excepção que cada um destes botões tem um nome que é dado por uma DIXO, em que esse nome não é mais que o nome da DIM que vão executar quando o utilizador carrega num deles. De salientar que antes de mandar executar a DIM remove todos os recursos que estão a ser reproduzidos.

Por fim, para terminar a análise deste módulo falta explicar os outros três componentes da figura 5.6 (Componentes 5 a 7). A explicação de cada um deles é referida abaixo:

- Componente nº 5 - Lista todas as DIMs existentes numa DID através do módulo DIPEngine do Software de referência.

- Componente nº 6 – Lista os argumentos da DIM escolhida no Componente nº5.

- Botão ***Execute*** – Este botão permite executar as DIMs com base nas opções escolhidas nas Componentes nº 5 e 6, e apagar todos os recursos que estão a ser reproduzidos na interface. Se não existirem DIMs então é retornado ao utilizador essa mesma informação se ele carregar neste botão.

Fluxograma do módulo MPEG21DT:

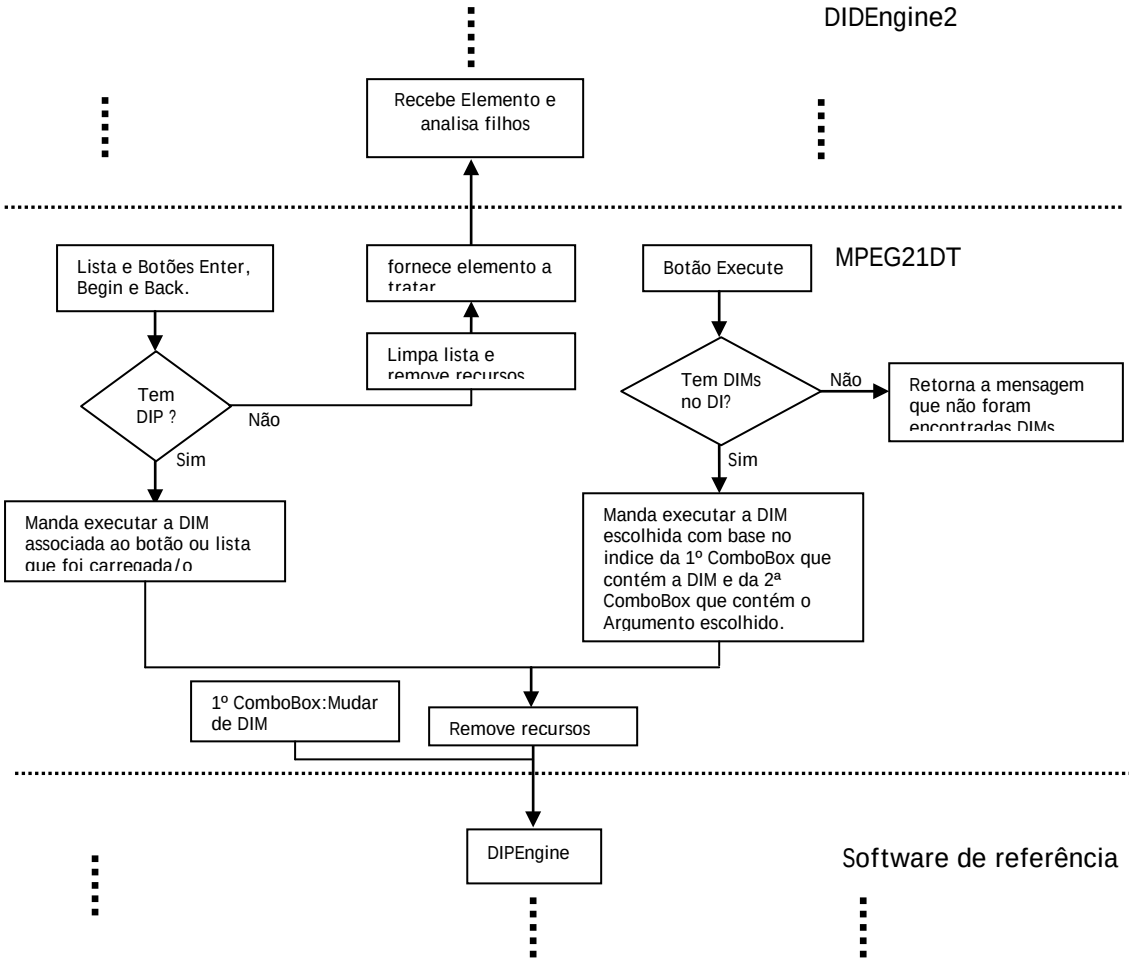


Figura 5.7 – Fluxograma resumido do módulo MPEG21DT.

Módulo do Software de Referência da norma MPEG-21

O *software* de referência da norma foi adaptado ao meu código sobretudo no que diz respeito à alteração do DIPEngine para execução de DIP por parte da aplicação MPEG21 DIP Teach. Além do DIPEngine do *software*, e face à existência de algumas limitações existentes no *software* e que foram analisadas no Capitulo 4, foram alterados a implementação da DIBO play e o módulo DIMEngine, respeitando contudo as restrições que a norma impõe. Cada um desses 2 módulos (DIPEngine e DIMEngine) e a alteração da implementação da DIBO play vão ser analisados de seguida, no que diz respeito à modificação, visto que já foram explicados no capítulo anterior.

DIBO play:

A implementação da DIBO play do *software* de referência, não tem em conta os *mimeTypes*, relevando-se assim uma limitação do software, visto que todos os recursos reproduzidos por esta DIBO eram enviados para serem reproduzidos por um *player* de áudio e vídeo. Por isso, a implementação da DIBO foi alterada, para que não reproduza através desse *player*, e sim para que através do DOM encontre o *mimeType* do *Resource*, e com base nele reproduza o recurso.

DIMEngine:

Este módulo não apresenta nenhum retorno visual para o utilizador após a execução de um DIM. Sendo assim foi acrescentado a este módulo, a funcionalidade de apresentar uma janela *popup* com a informação do resultado da execução de uma determinada DIM. Como vimos no capitulo 4, se executarmos uma DIM, por exemplo com um return de uma configureChoice, ela retorna *true* se a *Choice* foi alterada, ou *false* se não. Se este módulo não fosse modificado, o utilizador executava a DIM, e não acontecia nada, com a modificação do módulo aparece o resultado da execução da DIM na janela *popup* como ilustra a figura 5.8.

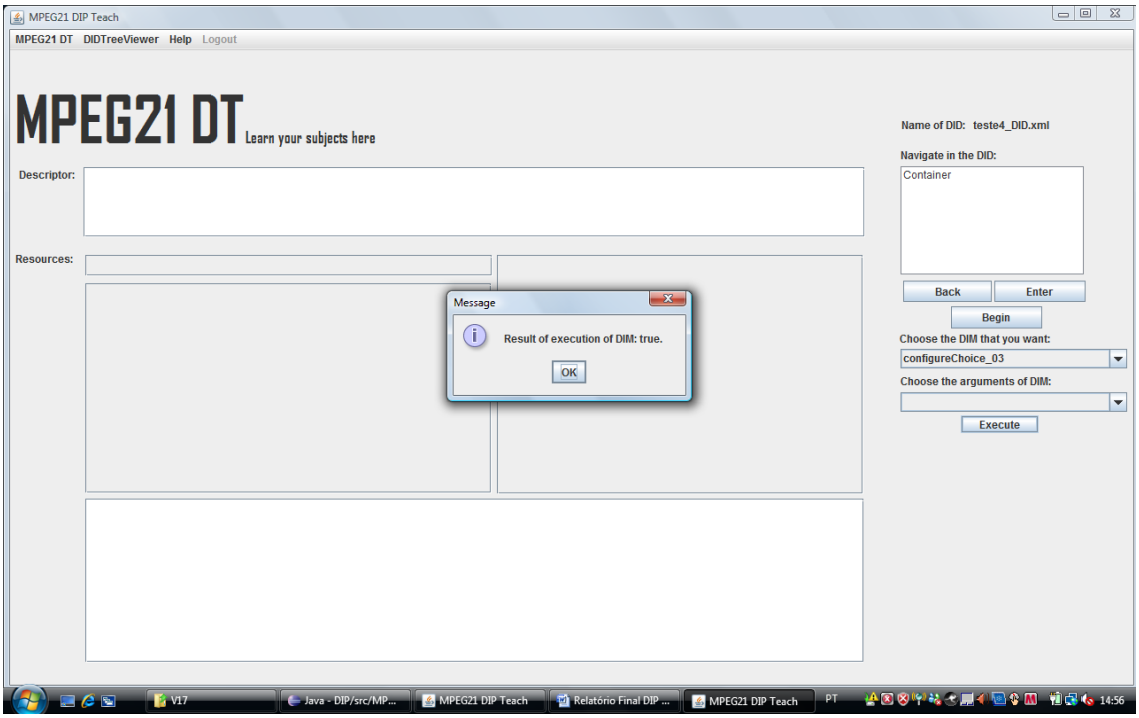


Figura 5.8 – Exemplo de retorno visual do resultado de execução de uma DIM.

DIPEngine:

Uma das coisas que foi modificada neste módulo, foi que em vez de mostrar as DIMs e os argumentos da DIM seleccionada, em janelas *popup*, mostrá-las/los nas ***ComboBoxes*** da

interface. Também foi implementado neste módulo a possibilidade mandar executar todas as DIMs definidas como automáticas, na altura do carregamento do *Digital Item*, visto que ele apenas só executava uma, como foi visto no Capitulo 4.

Fluxograma resumido do módulo Software de referência, com base no que foi referido neste tópico e no capitulo anterior:

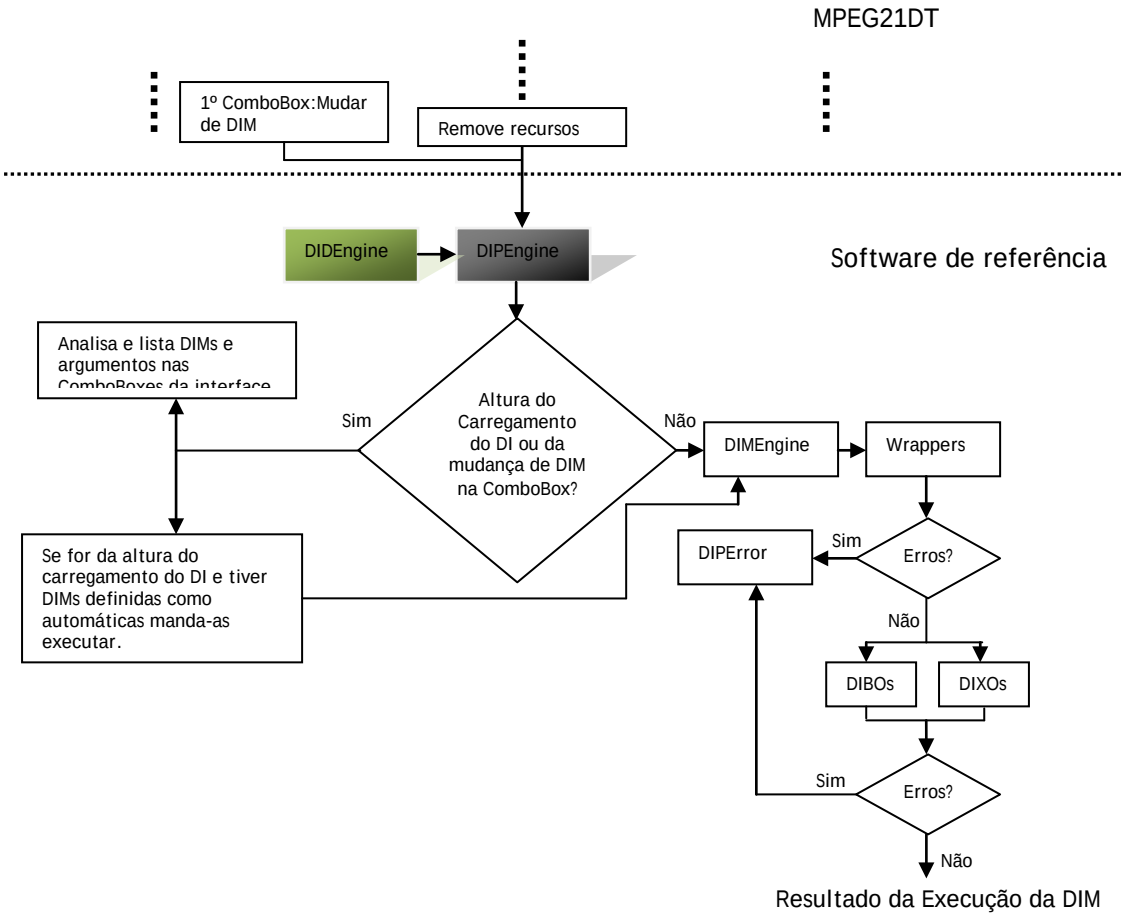


Figura 5.9 – Fluxograma resumido do módulo Software de Referência.

Módulo DIDTreeView

Este módulo é responsável por gerar a vista em árvore do *Digital Item*, mostrando ao utilizador a forma como ele está hierarquizado e o que contém cada elemento, para facilitar ao utilizador a navegação pelo DI. Esta opção de vista do DI não é obrigatória, isto é, o utilizador é que escolhe através da barra de menus do módulo controlMPEG21DT, se quer visualizar o DI desta forma ou não, podendo escolher caso queira visualizá-lo desta forma se quer ver os elementos do DI com atributos ou sem atributos. De salientar que a quando do arranque da aplicação ou a quando do carregamento de um DI, esta árvore não é apresentada para o utilizador, estando oculta na interface como vimos na primeira imagem desta aplicação (Figura 5.1). Na figura 5.10 mostra a interface com a vista em árvore de um *Digital Item*.

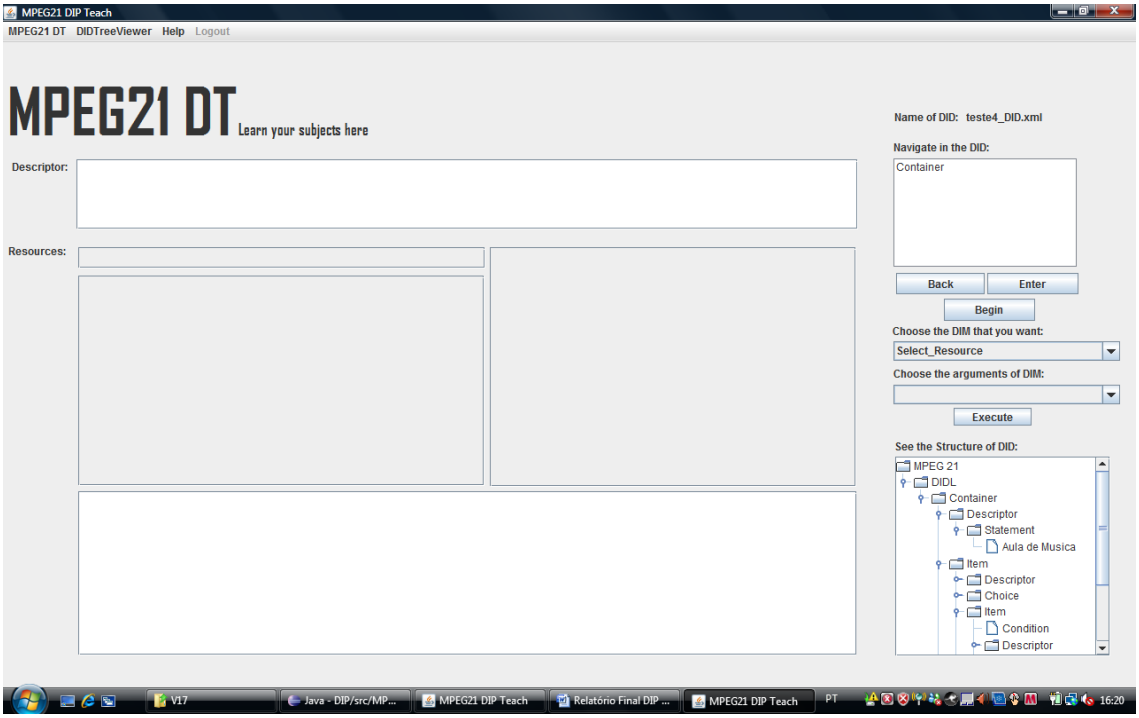


Figura 5.10 – Interface com o modo de vista em árvore activado.

Fluxograma do DIDTreeView:

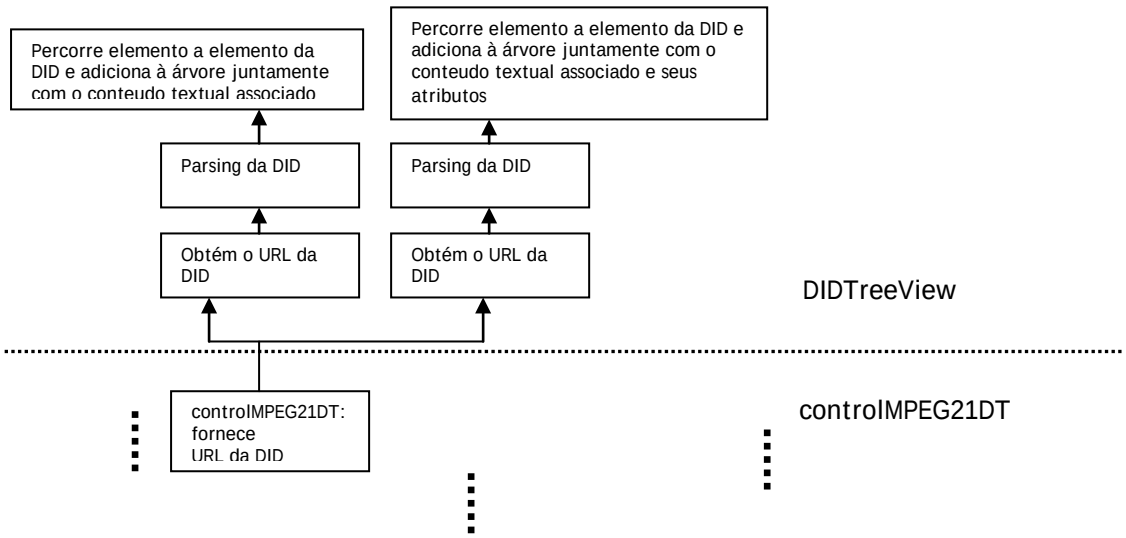


Figura 5.11 – Fluxograma resumido do módulo DIDTreeView.

Módulo controlMPEG21DT

O módulo controlMPEG21DT é responsável pelo controlo da aplicação no que diz respeito à organização dela, bem como da criação da barra de menus da MPEG21 DIP Teach, ao qual o módulo está associado e que é está ilustrada na figura 5.12 em diferentes perspectivas.

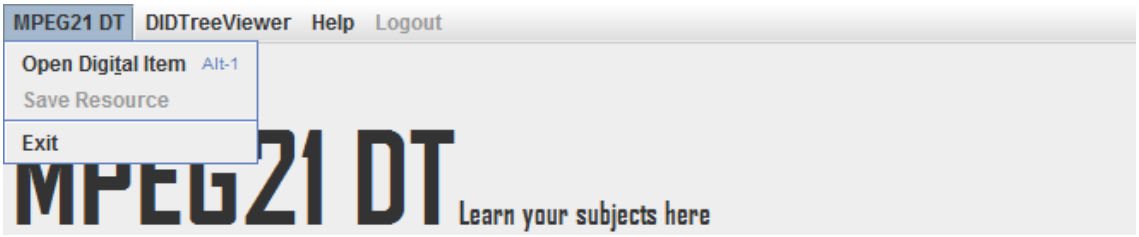


Figura 5.12.1 – Perspectiva 1 em que se vê os *items* do menu MPEG21DT.

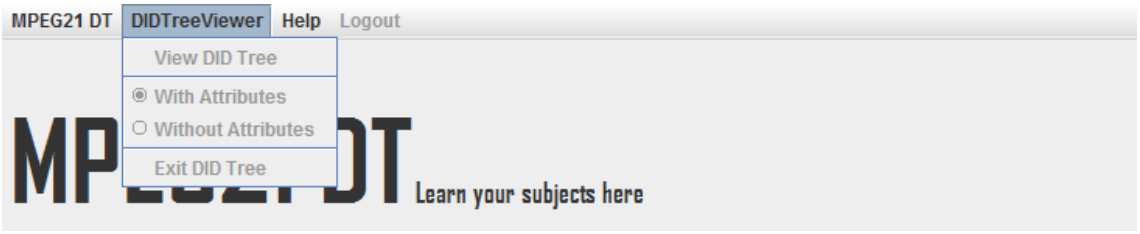


Figura 5.12.2 - Perspectiva 2 em que se vê os *items* do menu DIDTreeViewer.

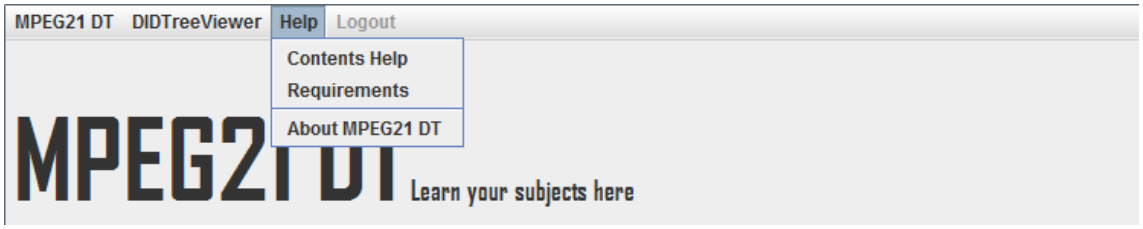


Figura 5.12.3 - Perspectiva 3 em que se vê os *items* do menu Help.

Figura 5.12 – Diferentes perspectivas da barra de menus da MPEG21 DIP Teach.

Analisando a figura 5.12 por partes, observamos que na primeira perspectiva (figura 5.12.1) tem um menu ao qual designei de MPEG21 DT, e que é considerado o menu principal da barra de menus. Este menu tem 3 items para o utilizador escolher, o ***Open Digital Item***, o ***Save Resource*** e o ***Exit***, cada um deles tem uma função associada a si quando é premido no estado activo, à excepção do segundo que está sempre desactivado. Os outros dois vão ser analisados de seguida:

- ***Open Digital Item*** – Este *item* quando acedido, ele tem como função abrir uma janela *popup* para utilizador escolher um DI. Após a escolha de um DI, apaga todos os recursos que estão a ser reproduzidos; limpa a lista e remove todos os elementos da árvore da MPEG21 DIP Teach; coloca todos os botões activos da interface incluindo todos os *items* do menu da DIDTreeViewer; gera a árvore e preenche a interface da MPEG21 DIP Teach com a informação proveniente do DI escolhido, através da inicialização dos módulos DIDEngine2, DIPEngine e DIDTreeView.

- ***Exit*** – A função deste *item* quando acedido é apenas sair da aplicação.

Na 2ª Perspectiva da figura 5.12 temos o menu DIDTreeViewer, este menu é composto por 4 *items*, o ***View DID Tree***, ***With Attributes***, ***Without Attributes***, ***Exit DID Tree***. Analisando a função associada a cada um destes *items* temos:

- ***View DID Tree*** – Torna visível a árvore do DI na aplicação.
- ***With Attributes*** – Remove os elementos que estão actualmente na árvore e gera a árvore do DI com atributos.
- ***Without Attributes*** – Remove os elementos que estão actualmente na árvore e gera a árvore sem atributos.
- ***Exit DID Tree*** – Torna oculta a árvore do DI na aplicação.

Na 3ª Perspectiva da figura 5.12 temos o menu Help em que os *items* nele presentes servem para disponibilizar informação ao utilizador sobre a aplicação.

Fluxograma resumido do controlMPEG21DT:

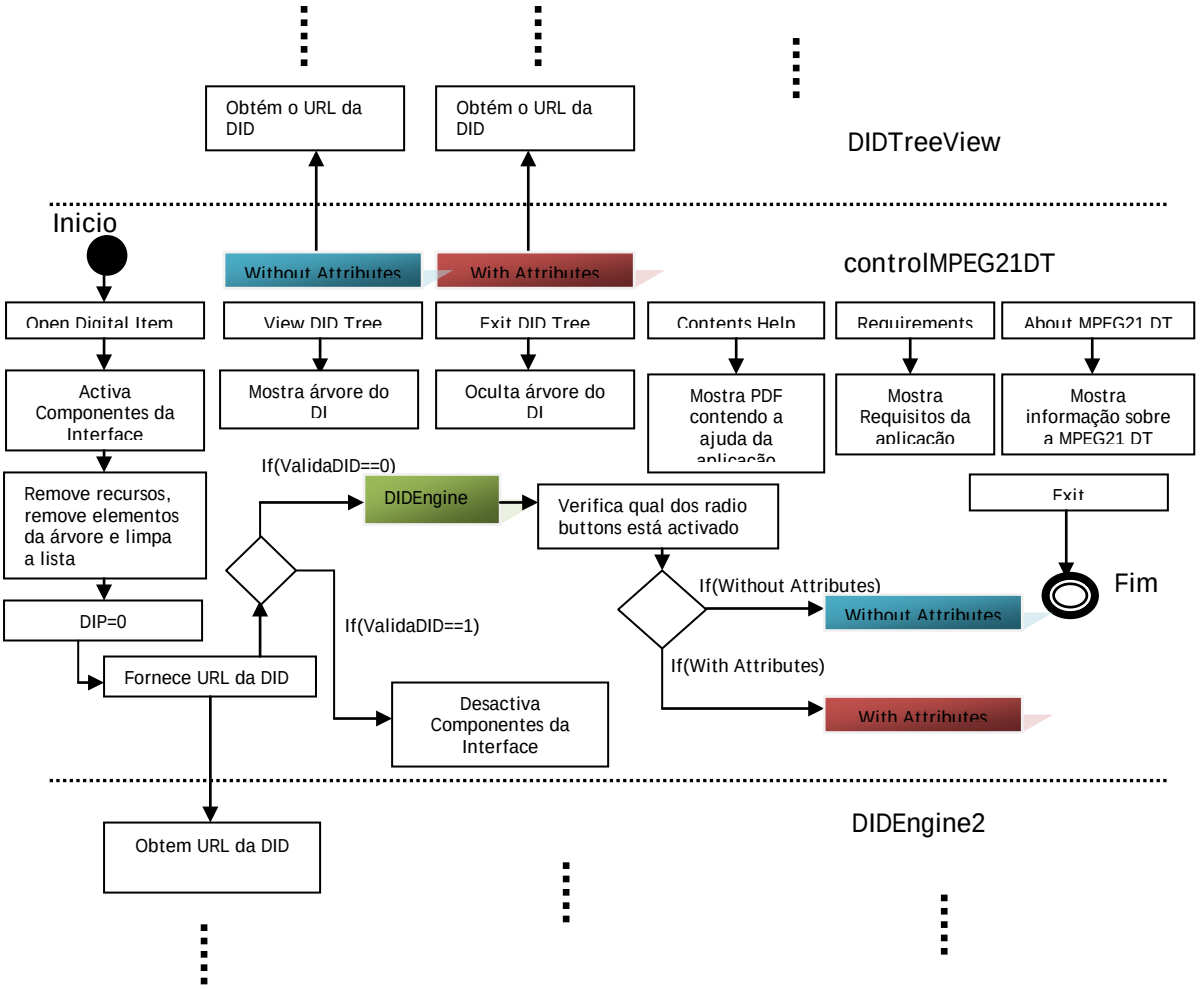


Figura 5.13 – Fluxograma resumido do módulo controlMPEG21DT.

O Fluxograma completo da aplicação MPEG 21 DIP Teach, o qual resulta da ligação dos diagramas de cada módulo, pode ser visto nos anexos.

5.3 – Conclusão

Neste capítulo foi explicada a aplicação MPEG21 DIP Teach, em termos de sua arquitectura e funcionalidades. Foram também discutidas quais as vantagens e as desvantagens de usar DIP ou não, no processamento dos *Digital Items*. Concluiu-se que com o uso da DIP resolve-se o problema da interoperabilidade, além de podermos adicionar mais funcionalidades ao *Digital Item* no que diz respeito ao seu processamento e à possibilidade de usar licenças para aceder a determinados recursos com direitos de autor. Por outro lado sem a DIP, podemos ter a noção de como o DI está estruturado, e dos conteúdos que ele apresenta de uma forma simples e prática.

No próximo capítulo é testado o funcionamento da MPEG21 DIP Teach.

Capítulo 6

Testes da utilização da MPEG21 DT

Após abordado todos os módulos da aplicação MPEG-21 desenvolvida em termos de funcionamento e arquitectura, este capítulo ilustra o funcionamento MPEG21 DT com base na unidade de transacção da norma MPEG-21, o *Digital Item*. Primeiro será abordado o comportamento da MPEG21 DT antes do carregamento do DI, depois o consumo do DI sem recurso à DIP, noutra fase o consumo do DI com recurso à DIP e por fim será também abordado um dos objectivo da dissertação *DIP based MPEG-21 Player*.

6.1 – Testes da utilização da MPEG21 DT

Quando o utilizador inicia a aplicação, todos os componentes da interface estão desactivados (Figura 6.1) de modo a que o utilizador não possa interagir com eles, visto que ainda não escolheu nenhum *Digital Item*, além destes componentes também estão os *items* do menu DIDTreeViewer, pela mesma razão. Ficando apenas activados para o utilizador o menu Help para consultar, caso necessite de ajuda e o menu principal, MPEG21 DT, no qual pode fazer o carregamento do DI e também sair da aplicação.

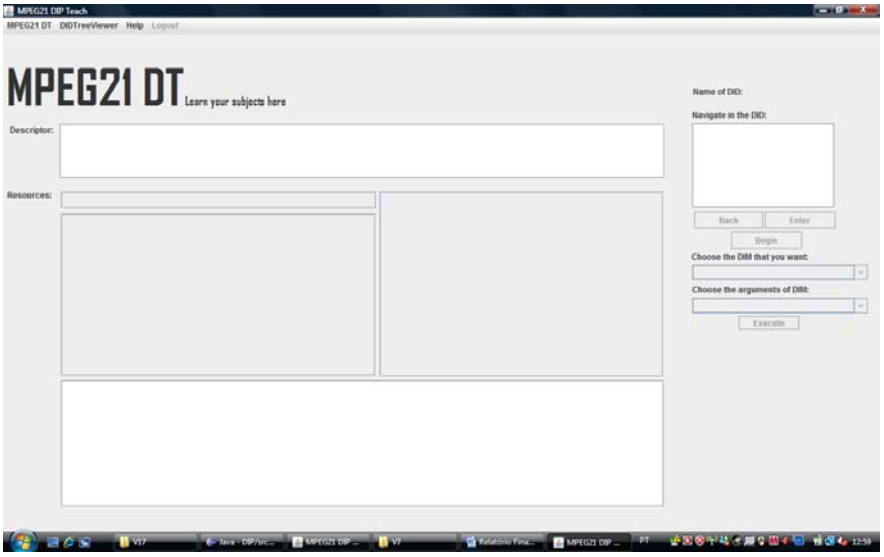


Figura 6.1 – Início da aplicação em que todos os componentes da interface estão desactivados.

Supondo que o utilizador ainda desconhece como trabalhar com a aplicação, ele pode a qualquer momento visualizar os conteúdos dos três *items* do menu Help para se inteirar do funcionamento da aplicação.

Para que o utilizador possa usufruir do *Digital Item*, tem que aceder ao menu MPEG21 DT, e carregar em ***Open Digital Item***. Após esse passo surge na aplicação uma janela de diálogo onde o utilizador pode procurar o *Digital Item* que pretende visualizar (Figura 6.2), sendo que essa procura pode ser em qualquer directório do seu computador.

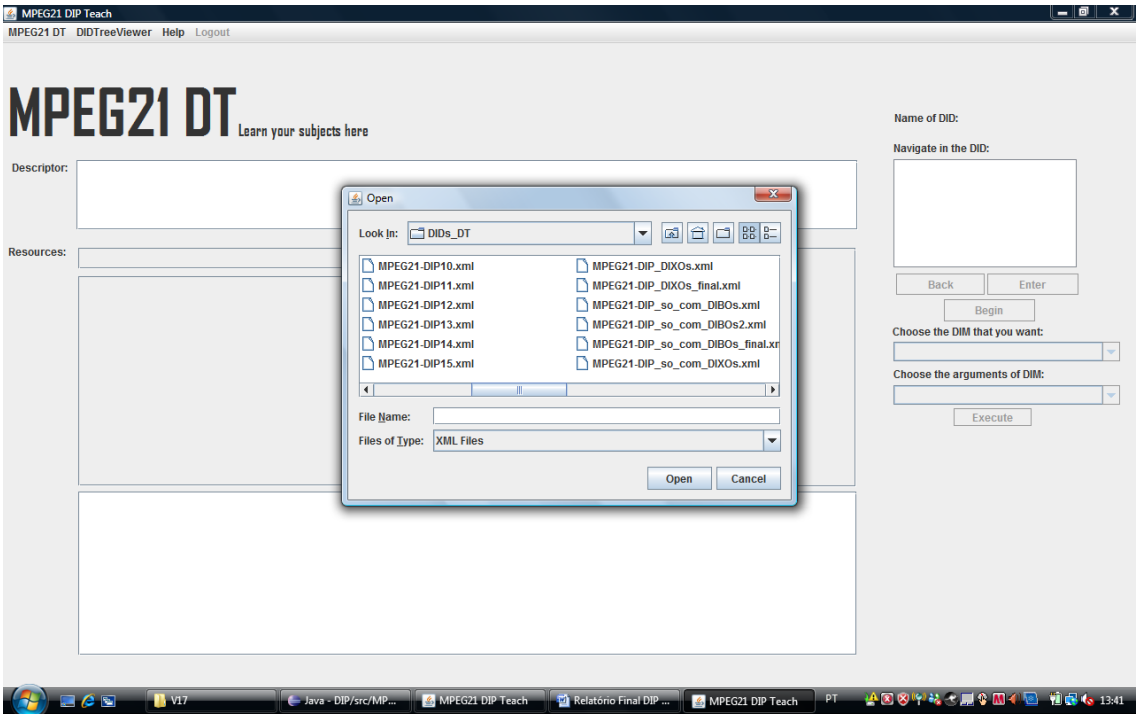


Figura 6.2 – Escolha de um DI na aplicação MPEG21 DIP Teach.

Escolhendo um *Digital Item*, a aplicação activa todos os componentes da interface, depois vai analisar se ele é valido, se não for, é retornado para o utilizador para escolher outro, desactivando de seguida os componentes e os *items* da interface tal como quando se inicia a aplicação; removendo os recursos que possam estar a ser reproduzidos nos painéis; removendo todos os *items* que possam estar na lista e todos elementos da árvore. Como ainda não foi feito um carregamento de um DI válido, esses recursos, elementos e *items* não existem. A figura 6.3 ilustra a informação de retorno da escolha de um *Digital Item* não validado pela aplicação, chamado de teste, e a figura 6.4 ilustra a interface após o utilizador confirmar a informação.

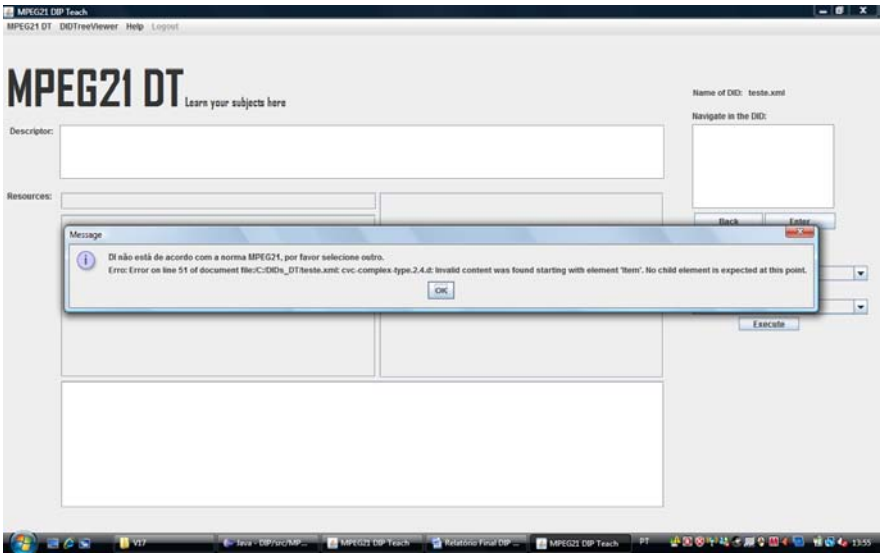


Figura 6.3 –Exemplo de informação de retorno no caso de um DI não validado.

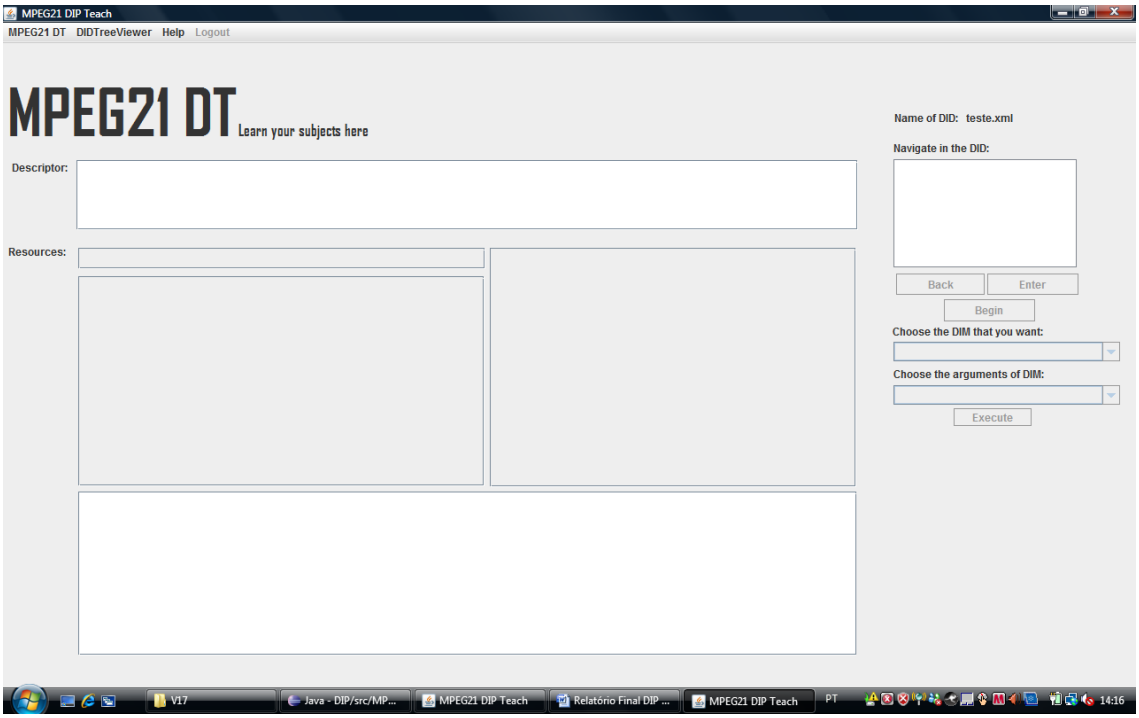


Figura 6.4 – Estado da interface após a escolha de um DI não válido.

Se o utilizador escolher outro DI, e se ele for validado, então ele poderá ser analisado, activando para isso todos os componentes da interface e *items* do menu DIDTreeViewer que estavam desactivados. A figura 6.5 ilustra isso mesmo, em que foi escolhido um DI válido designado de teste5.

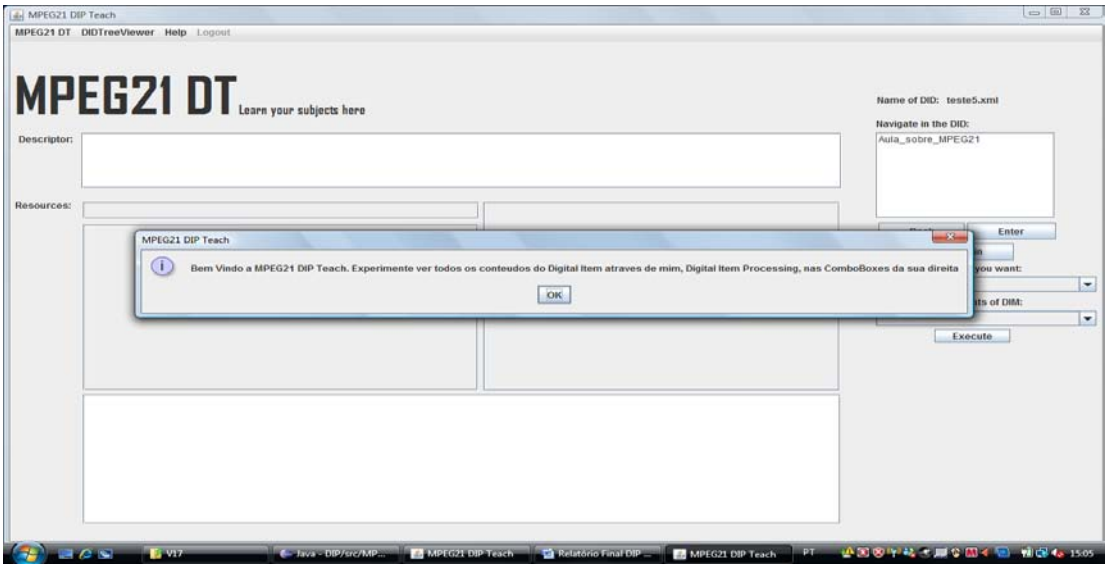


Figura 6.5 – Exemplo de escolha de um DI válido.

Como podemos constar na figura 6.5, os componentes foram activados e preenchidos com o conteúdo requerido, alem de que foi executada uma DIM que estava definida como automática na DID escolhida, com o objectivo de mostrar uma mensagem de boas-vindas para o utilizador. Sendo assim estamos na presença de um DI com DIP não só por ter executado a DIM definida como automática mas também por apresentar os seus métodos (DIMs) na interface, e que permitem manipular o DI escolhido de forma dinâmica. Todas as DIMs do DI, incluindo as definidas como automáticas estão listadas na 1ª ***ComboBox*** por a ordem a qual se encontram na DID, para permitir ao utilizador executá-las quantas vezes ele assim o pretender. A 2ª ***ComboBox*** é para listar os argumentos da DIM que está escolhida na 1ª ***ComboBox*** (Figura 6.6).

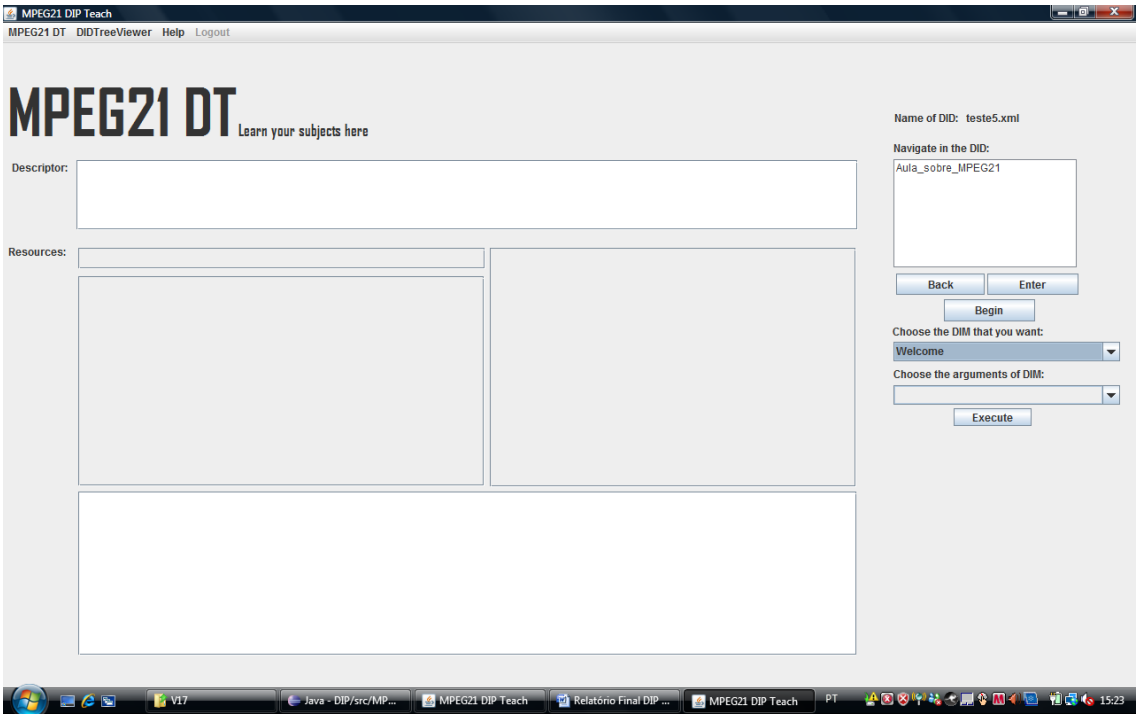


Figura 6.6 – Activação dos componentes da interface e seu preenchimento após a escolha de um DI válido.

Além de podermos executar os métodos, podemos navegar no DI através da lista e dos 3 botões que estão por baixo dela. Para navegar na lista então usamos esses 3 botões que fazem o que o próprio nome sugere, ou seja, o botão ***Back*** permite retroceder para o elemento anterior, o botão ***Begin*** permite voltar ao início, e o botão ***Enter*** permite entrar dentro de um elemento. De referir ainda que na lista só são mostrados os elementos *Container* e *Item* ou o id deles se eles tiverem. Os *Components* e os *Descriptors*, não aparecem na lista e são reproduzidos automaticamente sempre que navegamos para um elemento que tenha como filhos *Components* e *Descriptors*. Voltando à lista o/s primeiro/s elemento/s que aparecem quando do carregamento de um DI são os filhos do elemento root, que neste caso são os filhos do elemento DIDL. Como na DID escolhida só temos um elemento como filho do root, que é o *Container*, e como este tem um id associado a ele, então é esse id que vai ser adicionado à lista ("Aula_sobre_MPEG21"). Se navegarmos para dentro deste elemento, seleccionando o id e carregando na tecla ***Enter***, ou alternativamente clicando duas vezes sobre o id, vamos listar os filhos dele, tendo em conta o que foi referido atrás (Figura 6.7).

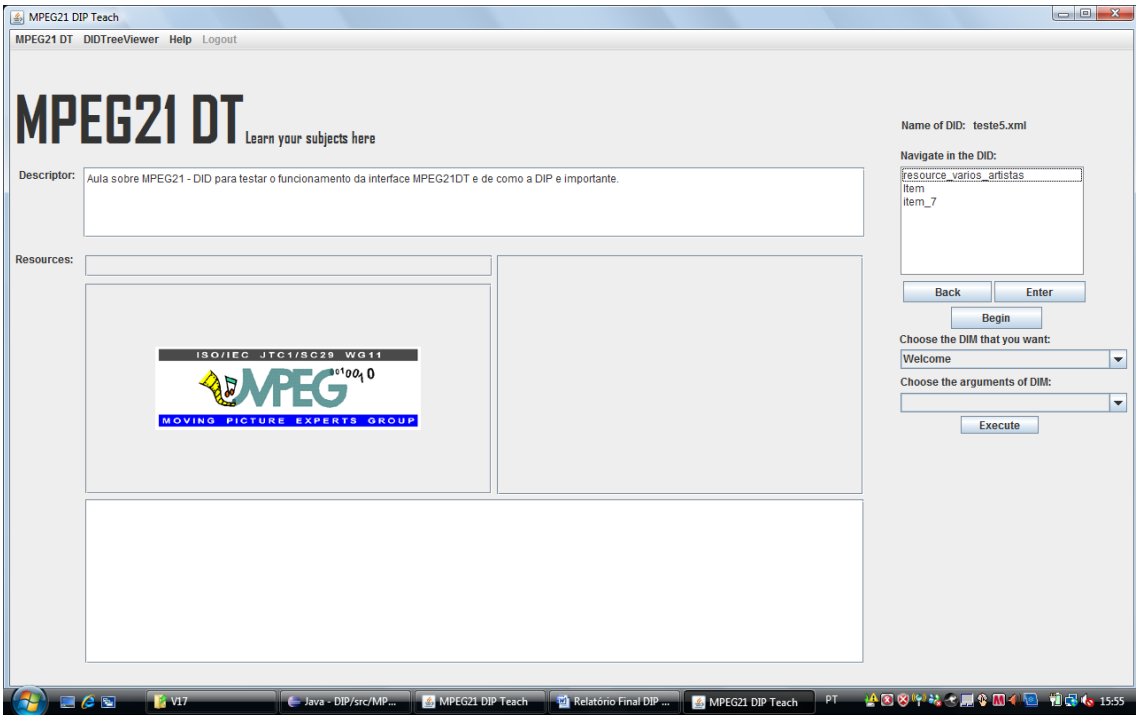


Figura 6.7 – Exemplo de navegação para um dentro de um elemento de um DI.

Analisando a figura 6.7 chegámos à conclusão que o elemento ao qual entramos, tem três filhos que podem ser listados, neste caso são três *Items*, se bem que dois deles são representados através do id, o primeiro e o ultimo *item* da lista, e tem pelo menos mais um que não é listado na lista, mas que foi reproduzido automaticamente, neste caso um *Descriptor*, o qual tem informação textual e uma imagem associada. Se agora navegarmos para dentro do primeiro *item* que está representado na lista da figura 6.7 com o id "resource vários artistas" obtemos o resultado ilustrado na figura 6.8.

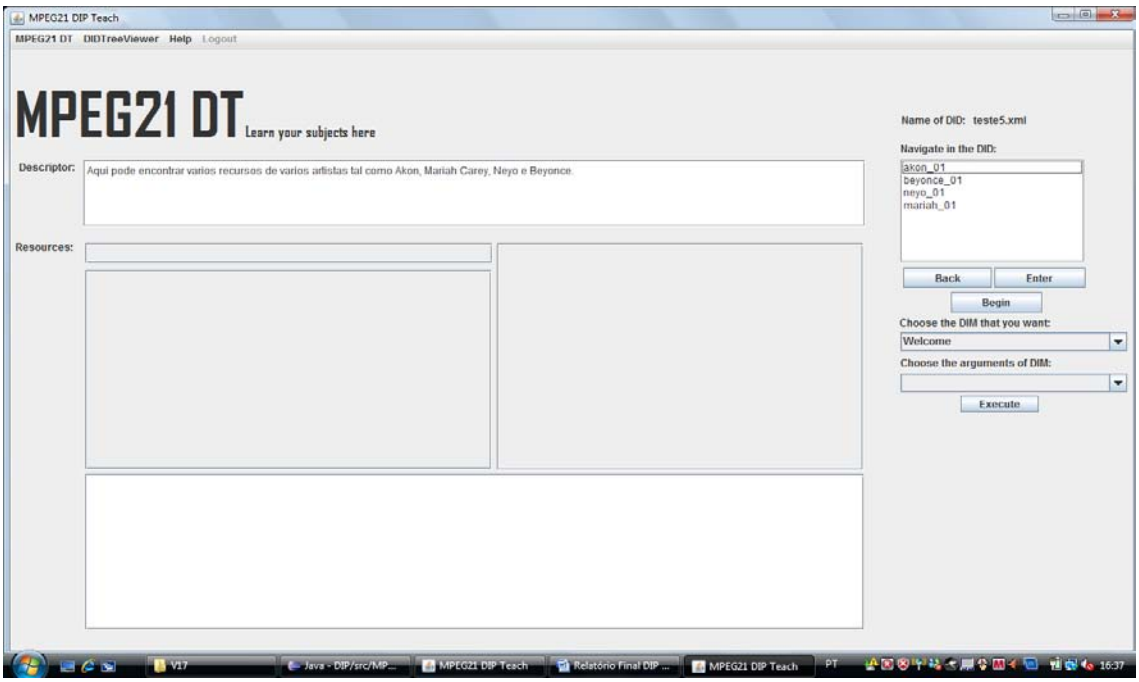


Figura 6.8 – Exemplo de navegação para dentro de um *item* que contém outros *items*.

Sendo que agora podemos aceder a 4 *items*, cada um deles correspondendo a um artista de música segunda a informação proveniente da interface ilustrada na figura 6.8. Se acedermos ao primeiro *item* representado na lista, ele contém um *item*, o qual dentro desse *item* tem 4 *items*, sendo que estes últimos, estão cada um associado a recursos do artista escolhido (figura 6.9).

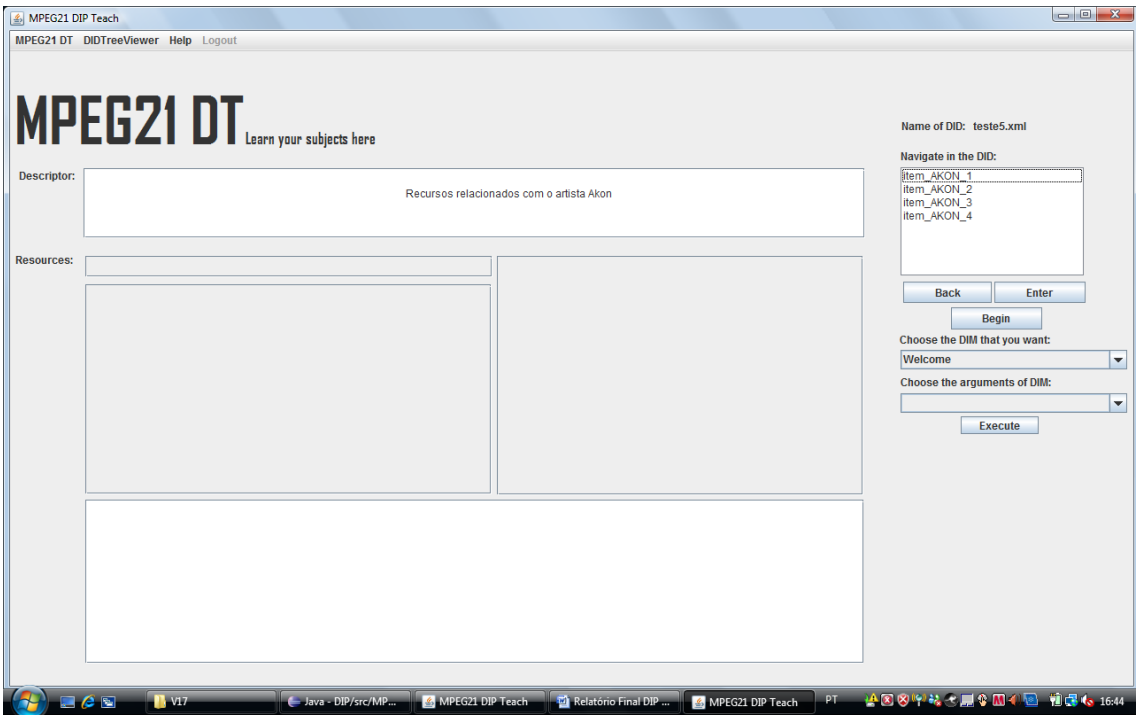


Figura 6.9 – Exemplo de navegação para dentro de um *item* contendo vários *items* com recursos.

Se continuarmos a navegar na DID e acedermos ao segundo *item* da lista que tem o id "item_Akon_2" da figura 6.9 é reproduzido um recurso de música automaticamente no painel correspondente, como podemos observar na figura 6.10.

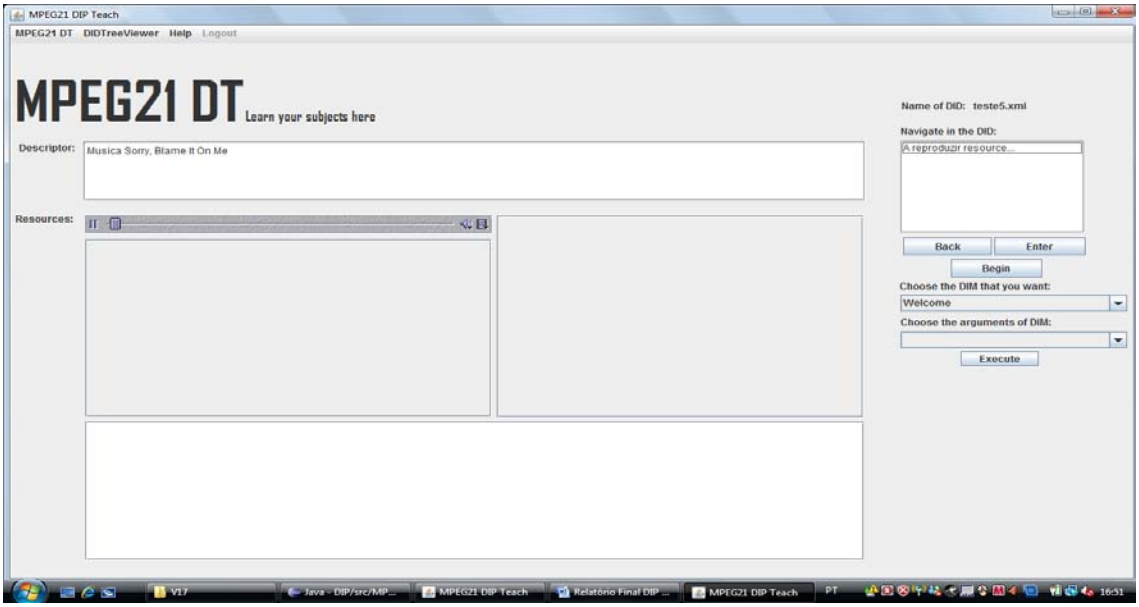


Figura 6.10 – Exemplo de reprodução de um recurso de música.

A partir daqui não podemos avançar mais na DID porque como podemos ver na lista da figura 6.10, ela não tem mais elementos para aceder, a informação que aparece na lista a informar de que um recurso está em reprodução, não passa disso mesmo, uma informação, não se tratando assim de um elemento para aceder. Contudo podemos retroceder ou voltar ao início carregando no botão ***Back*** ou ***Begin*** respectivamente. Se optarmos por ***Back*** regressámos ao estado anterior e que está representado na figura 6.9, se for por ***Begin*** regressamos ao estado da figura 6.6, e assim podemos continuar a navegar na DID, visualizando o que ela nos pode oferecer com base na aplicação.

De salientar que todos os *Resources* e *Descriptors* em reprodução são apagados, sempre que se acede a um elemento, se retrocede ou se volta ao início.

Depois de ilustrado o funcionamento da aplicação no que respeita à navegação na DID sem recurso à DIP, será ilustrado agora o funcionamento da DIP na aplicação. Para tal são executados dois métodos desenvolvidos, para testar o funcionamento da DIP. Os métodos estão listados na 1ª ***ComboBox*** e os argumentos do método seleccionado na segunda. A figura 6.11 ilustra alguns dos métodos presentes no DI escolhido.

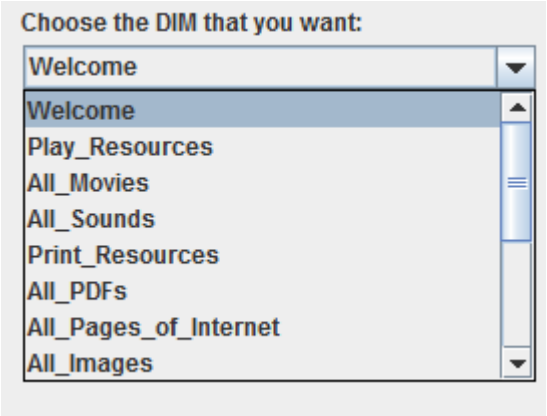


Figura 6.11 – Lista de DIMs do DI escolhido.

O primeiro método que está representado na lista, Welcome, trata-se de uma DIM que tem como objectivo mostrar uma mensagem de Boas-Vindas e que já foi executado quando do carregamento do DI, visto estar definida como automática. O segundo método ainda não foi executado, e por isso vai ser executado e analisado o funcionamento desta DIM. Esta DIM é constituída por várias DIBOs de modo a permitir fazer reprodução de vários recursos. Ao seleccionar esta DIM, observamos que é uma DIM que não aceita argumentos (Figura 6.12), e se quisermos executá-la basta carregar no botão ***Execute***.

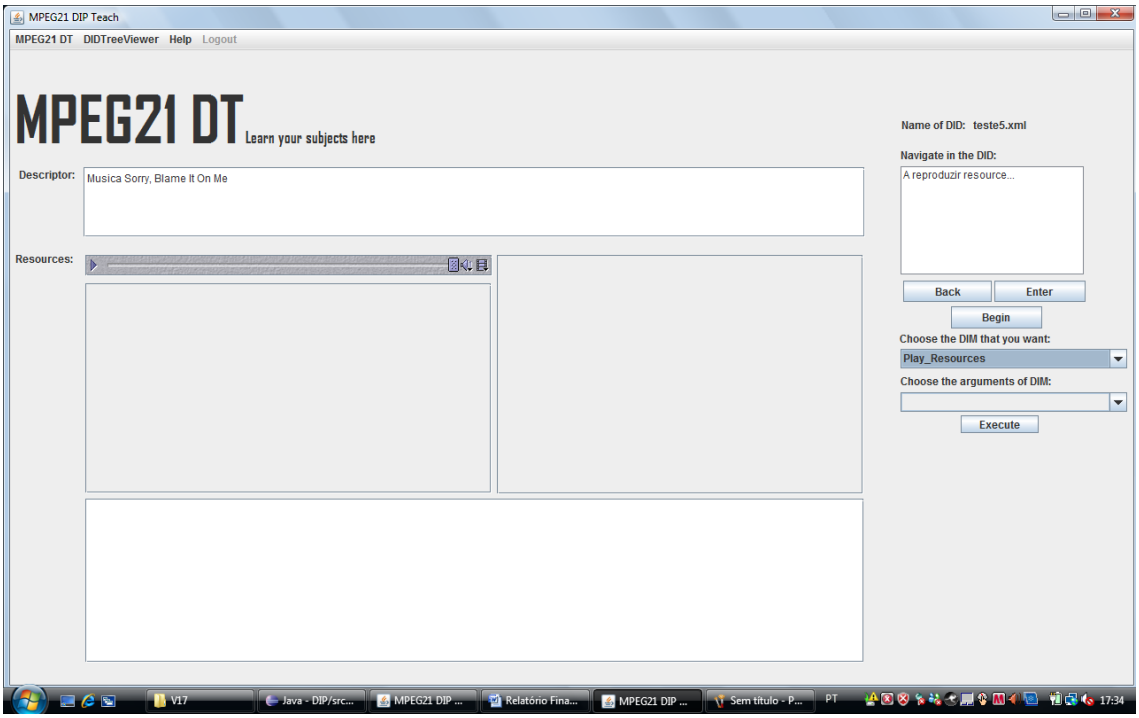


Figura 6.12 – Selecção da DIM Play_Resources do DI escolhido.

Executando então a DIM Play_Resources, verificamos que se trata de uma DIM que usa a DIBO configureChoices, uma vez que nos apresenta uma *Choice* para nós escolhermos uma opção (figura 6.13.1). Se optarmos por não escolher nenhuma das opções *(Selections)*, carregando em ***Cancel*** é retornada uma mensagem de informação (figura 6.13.2), se por

outro lado não respeitarmos o que nos pede a *Choice*, isto é, se seleccionarmos mais de uma opção (mais de um *true*) é nos retornado uma mensagem de erro (figura 6.13.3), e por fim se escolhermos apenas uma opção vamos para outra *Choice*, a qual permite escolher recursos a serem reproduzidos (figura 6.14). De salientar tal como a primeira *Choice*, esta também está "equipada" com mensagens do género, caso escolhermos recursos do mesmo tipo ou caso carregamos em ***Cancel***.

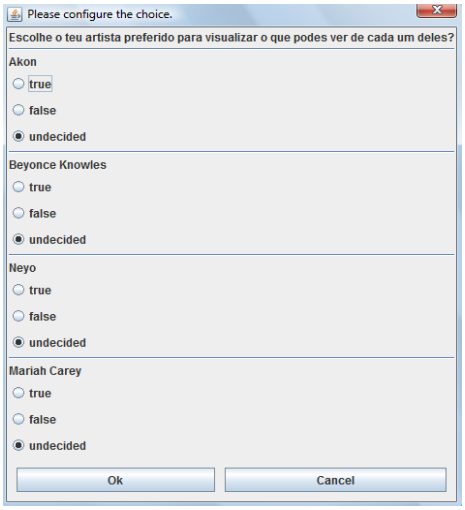


Figura 6.13.1 – Primeira *Choice* que é apresentada da DIM Play_Resources.

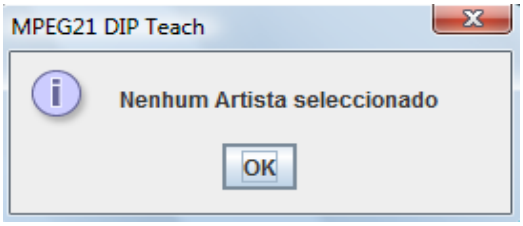


Figura 6.13.2 – Mensagem de informação.

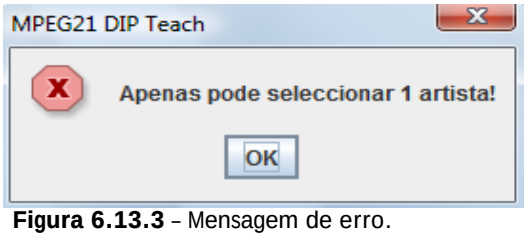


Figura 6.13.3 – Mensagem de erro.

Figura 6.13 – Resultado da execução da DIM Play_Resources (1ª Parte).

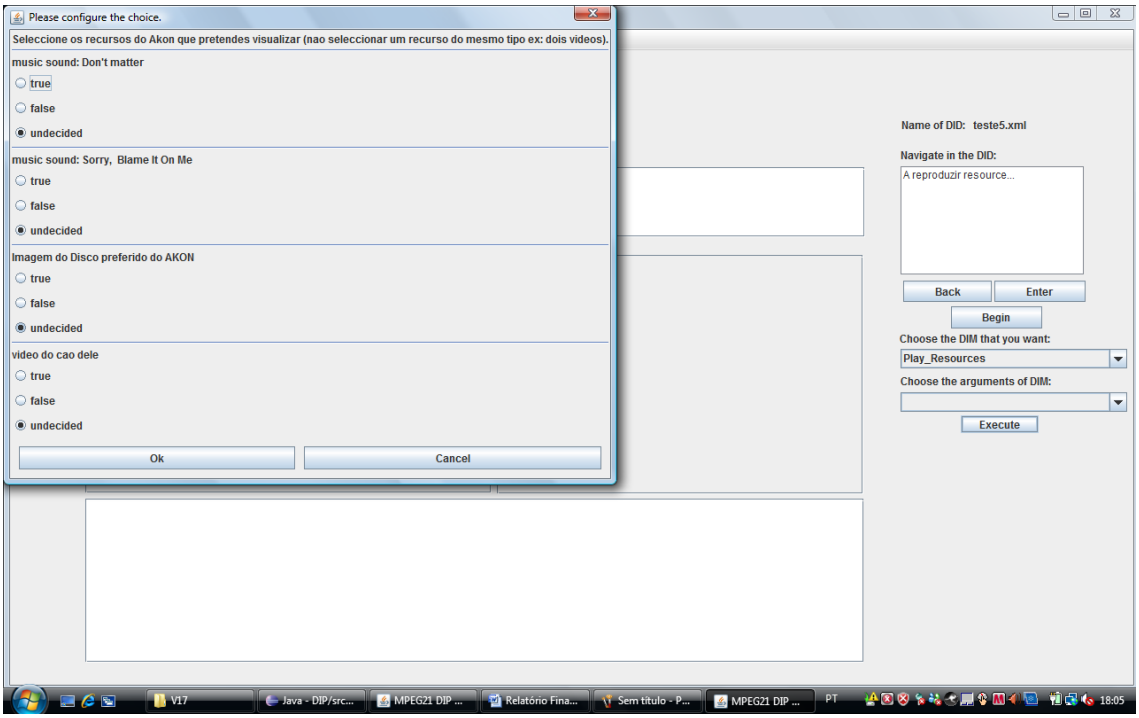


Figura 6.14 – Resultado da execução da DIM Play_Resources (2ª Parte).

No caso da DIM Pay_Resources definiu-se como *true* a opção Akon da primeira *Choice*, o que resultou na apresentação de outra *Choice* com os recursos disponíveis para visualizar

deste artista (figura 6.14). Nesta última *Choice* vaie-se executar um recurso de áudio, um de vídeo e outro de imagem, para isso define-se como *true* a primeira ou a segunda opção, a terceira e a quarta. O resultado final do *play* destes recursos está ilustrado na figura 6.15.

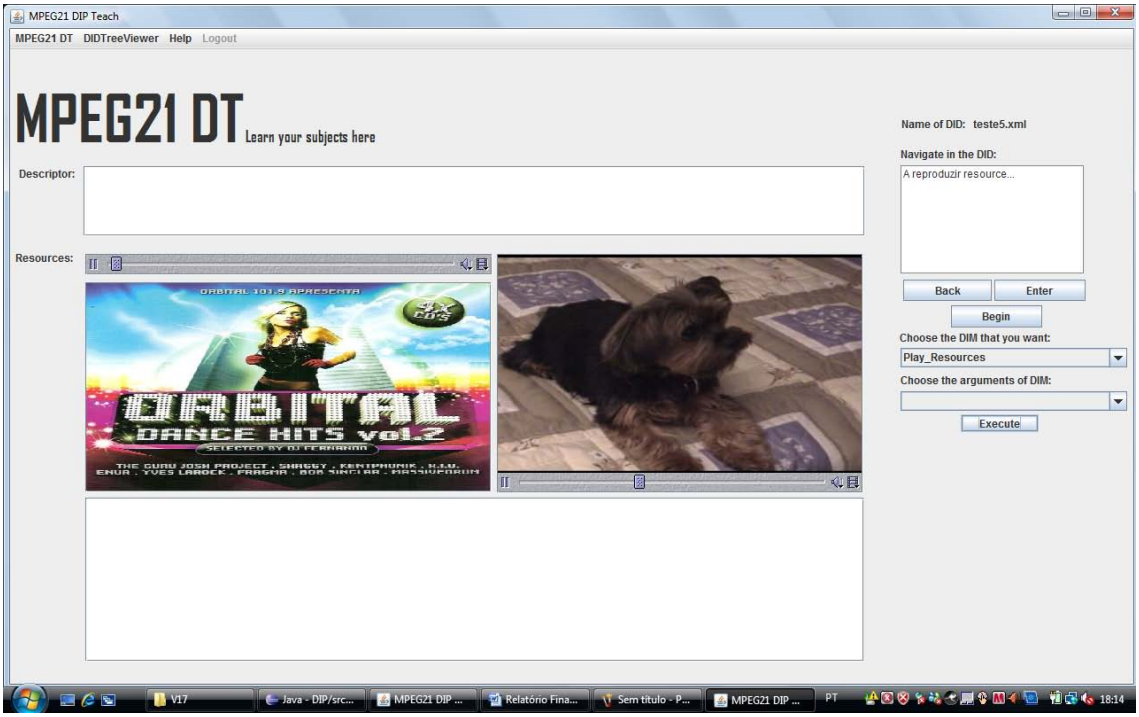


Figura 6.15 – Resultado da execução da DIM Play_Resources (3ª Parte).

Para finalizar esta parte da DIP, vai ser executada uma DIM com argumentos, mais concretamente a DIM All_PDFs da figura 6.11. Esta DIM tem como função filtrar todos os PDFs para o utilizador escolher qual deles quer visualizar. Como no DI escolhido, só existe um PDF esta DIM tem só um argumento como mostra a figura 6.16. Se mandarmos executar esta DIM obtemos o resultado que está ilustrado na figura 6.17.

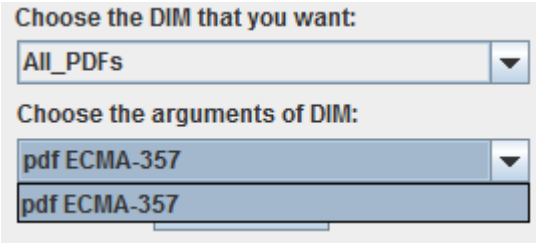


Figura 6.16 – Argumentos da DIM All_PDFs.

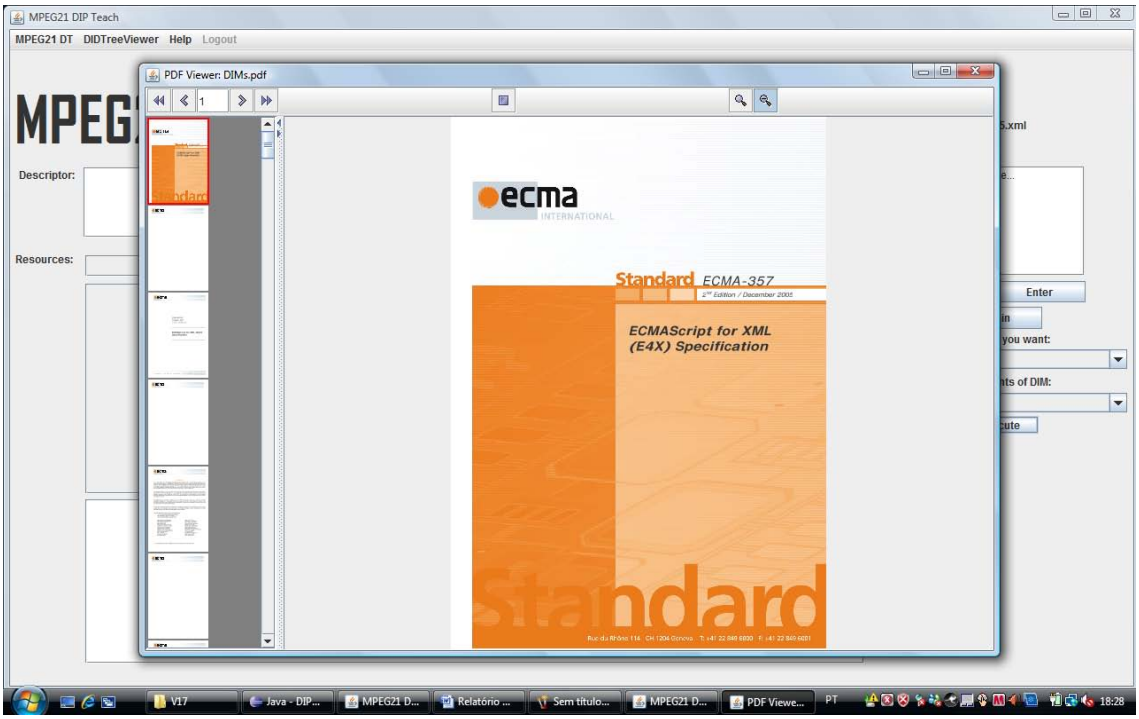


Figura 6.17 – Resultado da execução da DIM All_PDFs.

De salientar que sempre que se carrega no botão ***Execute***, todos os recursos que estão a ser reproduzidos na interface são removidos e que no final de execução de cada DIM é retornado uma mensagem de que a DIM foi executada, que permite assim ao utilizador saber. Se bem que nos testes destas DIMs não se mostrou a mensagem, porque todas as DIMs que foram executadas tinham retorno visual para o utilizador, o que permite ao utilizador saber que ela foi executada, de qualquer maneira a mensagem que aparece é a que está ilustrada na figura 6.18, e normalmente o conteúdo textual muda quando na DIM que construirmos na DID, tem uma instrução do tipo return.

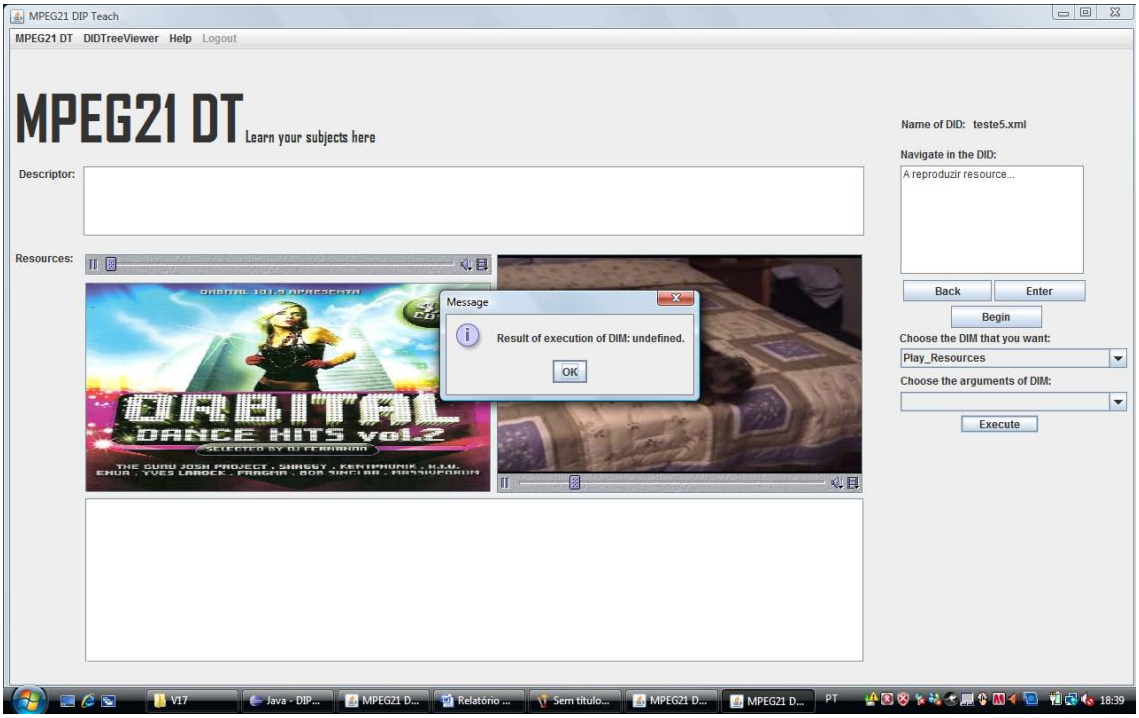


Figura 6.18 – Mensagem que permite saber se a DIM foi executada com sucesso.

Para concluir os testes, falta abordar o menu DIDTreeViewer que permite caso o DI seja validado ver a estrutura hierárquica do DI. Este menu dispõe de 4 *items*: ***View DID Tree***, ***With Attributes***, ***Without Attributes*** e ***Exit DID Tree*** (figura 6.19).

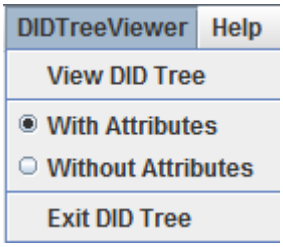


Figura 6.19 – Menu DIDTreeViewer da MPEG21 DIP Teach.

Quando se carrega na opção ***View DID Tree***, aplicação mostra a árvore do DI no canto inferior direito com ou sem atributos, dependendo de qual selecção está activada, se tiver activada a opção ***With Attributes***, mostra a árvore do DI com atributos, se for a ***Without Attributes***, mostra a árvore do DI sem atributos (figura 6.20). Se carregarmos na selecção ***With Attributes***, mostra a árvore do DI com atributos se a árvore estiver visível, o mesmo procedimento acontece com a outra selecção, só que sem atributos. Por fim o ***Exit DID Tree*** permite ocultar a árvore.

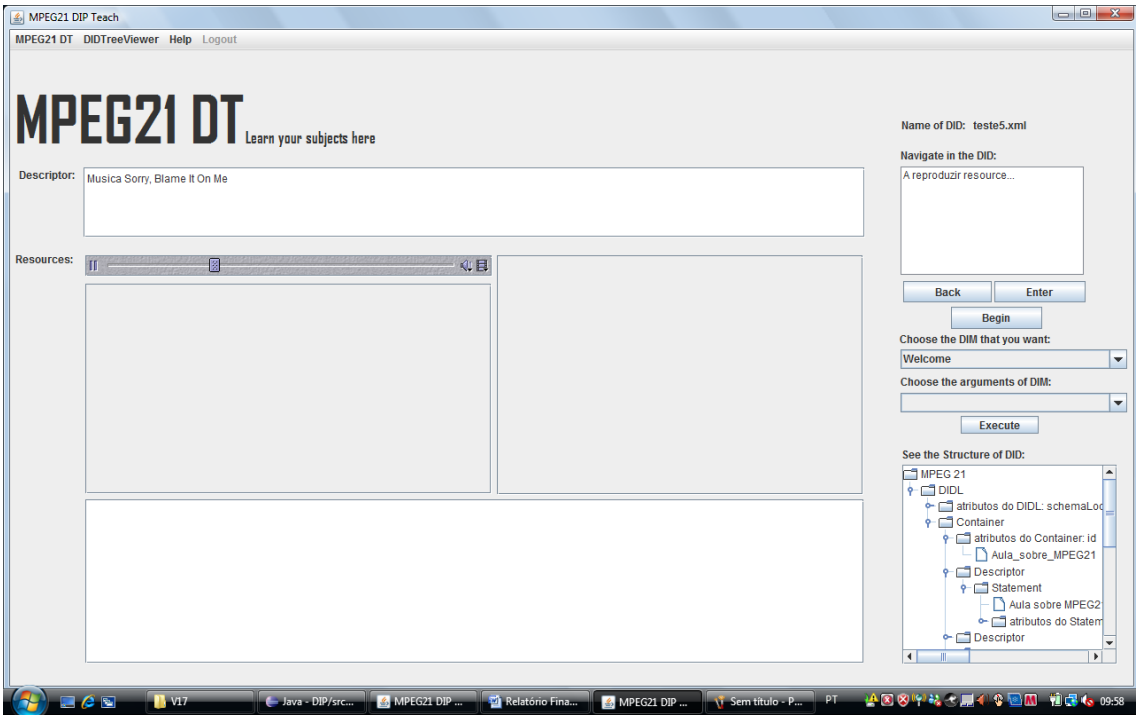


Figura 6.20 – DID Tree View com atributos na MPEG21 DIP Teach.

6.2 – Visualizações diferentes do mesmo DI

Um dos objectivos da Dissertação foi mostrar que é possível termos duas máquinas diferentes, neste caso dois computadores, a executar a aplicação MPEG21 DIP Teach e a aceder à mesma DID ao mesmo tempo, obtendo visualizações diferentes da navegação do mesmo DI em cada máquina. Para mostrar que isso é possível, seguiu-se o seguinte procedimento: após ter a aplicação desenvolvida, fez-se uma cópia da aplicação para outro computador e foi mudado nessa cópia o layout da Interface MPEG21 DIP Teach, mudando as cores e as posições dos componentes, tal como da janela *popup* que apresenta a *Choices*. A figura 6.21 mostra isso mesmo, em duas máquinas diferentes, o mesmo DI carregado e o modo de visualização é completamente distinto nas duas máquinas.

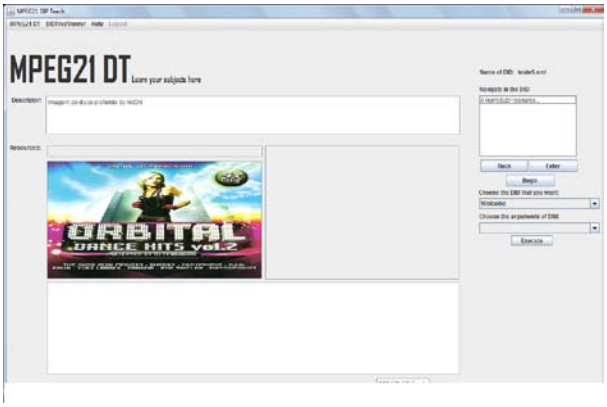


Figura 6.21.1 – Interface no Computador 1.

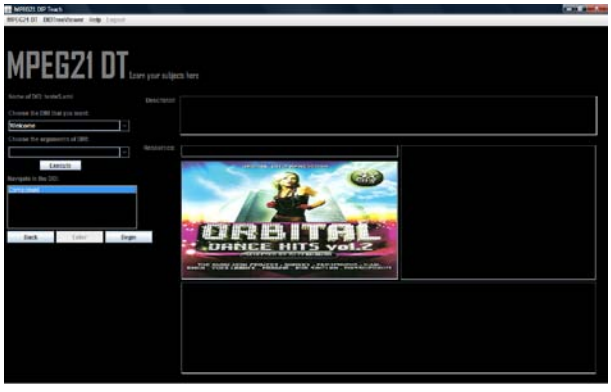


Figura 6.21.2 – Interface no Computador 2.

Figura 6.21.3 – *Choice* no Computador 1.

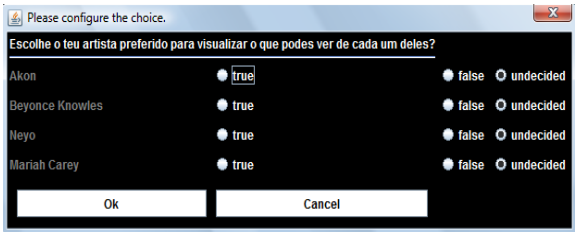


Figura 6.21.4 – *Choice* no Computador 2.

Figura 6.21 – Figuras que ilustram um dos objectivos da dissertação.

6.3 – Conclusão

Este capítulo ilustrou o funcionamento da aplicação no consumo de *Digital Items* com recurso à DIP e sem recurso à DIP. Foi também explicado um dos objectivos da dissertação e que consistiu na possibilidade de visualizar de forma diferente a mesma DID, em duas máquinas diferentes, consoante a "pilha" DIP que cada máquina tem.

No próximo capítulo são analisadas as DIXOs desenvolvidas, bem como as limitações encontradas no desenvolvimento delas e que de algum modo não permitem fazer tudo o que o autor pretende.

Capítulo 7

DIXOs

As DIXOs são operações que permitem adicionar interactividade a um DI, dependendo do que o autor pretende e do que as DIXOs podem fornecer. Contudo elas têm algumas regras que têm de ser seguidas. Com base nas DIXOs desenvolvidas e que vão ser explicadas neste capítulo, foram encontradas algumas limitações, que impedem o desenvolvimento do que se pretende que elas façam. No entanto, as DIXOs desenvolvidas para navegar na DID e para reproduzir os *Components*, que são as DIXOs de maior importância, foram realizadas com sucesso.

7.1 – Limitações das DIXOs

As DIXOs podem ser invocadas através de um ficheiro class ou então de um ficheiro jar como foi visto no capítulo 4. De salientar que quando se desenvolve uma DIXO, por exemplo, com o nome validate.java, se construirmos essa DIXO temos que ter atenção ao ficheiro class ou ficheiros class gerados para essa DIXO validate no directório bin do *workspace* da aplicação, ou seja, se é gerado apenas um ficheiro class (validate.class) a DIXO é bem executada se a implementação desta estiver obviamente correcta, quer invocando na DID pelo ficheiro validate.class ou quer pelo validate.jar. Mas se a DIXO validate tiver mais do que um ficheiro class (por exemplo validate.class e validate$1.class), e mesmo que a implementação da DIXO esteja correcta, quer a invocação da DIXO validate seja através do ficheiro validate.class ou pelo ficheiro validate.jar, a DIXO não é executada dando erro na consola por tentativa ilegal de aceder à classe validate$1.class, mesmo que se coloque esse ficheiro validate$1.class no directório onde está a validate.class ou o validate.jar . Isto torna-se uma limitação muito incómoda para o programador porque não pode construir uma DIXO com mais de uma class, tendo que tentar optar por outras vias, como foi o caso de, por exemplo, a DIXO desenvolvida para mudar as cores da interface. Visto que na implementação desta DIXO são criados vários ficheiros com a extensão class devido ao facto de ela ter definida acções para alguns componentes, a solução foi que este ficheiro já faz parte da aplicação e foi usada uma DIXO que apenas executa esse ficheiro, funcionando como se se

carrega-sse num botão, fazendo então aquilo que pretendia e que uma DIXO era suposto fazer na sua totalidade. Normalmente são criadas mais que uma classe referente a um ficheiro Java quando esse ficheiro tem funções dentro de funções, o que é muito comum nas acções de um componente de uma interface. A figura 7.1 mostra um exemplo de código que gera um ficheiro class a mais.

```
	private JButton getJButton1() {
		if (jButton1 == null) {
			jButton1 = new JButton();
			jButton1.setBounds(new Rectangle(227, 216, 83, 23));
			jButton1.setText("Cancel");
			jButton1.addActionListener(new
java.awt.event.ActionListener() {
				public void
actionPerformed(java.awt.event.ActionEvent e) {
					System.out.println("actionPerformed()");
					// TODO Auto-generated Event stub
actionPerformed()
					thisClass.show(false);
					//System.exit(0);
				}
			});
		}
		return jButton1;
	}
```

Figura 7.1 - Exemplo de código que gera um ficheiro class a mais.

Uma possível solução para tentar resolver este problema foi criar uma DIXO que executa Jars, ou seja uma DIXO que executa o jar do ficheiro que tem o código que gera a interface e muda as cores. Esta tentativa foi em vão, apesar de apresentar a interface para o utilizador mudar as cores, não funciona quando ele carrega para mudar as cores, porque é como se tivesse a executar duas aplicações distintas ao mesmo tempo, não conseguindo reconhecer a outra aplicação para lhe mudar as cores. Contudo esta possível solução é a ideal quando se quer apresentar uma interface em que o código dela não depende da aplicação resolvendo assim o problema da DIXO e funcionando na perfeição. Outra solução que foi passar o jar que contêm o código da interface para mudar as cores através da DIBO execute, sendo que esta maneira não resultou, dando erro na interface, sem sequer abrir o ficheiro. Sendo assim e devido à falta de mais soluções para executar a DIXO para mudar as cores, ficou a funcionar, ou seja, o código na aplicação e através de uma DIXO abre-o, permitindo assim mudar as cores.

Outro problema com que deparamos prende-se com as relações entre DIXOs. Foi constatado que na maior parte dos casos elas são independentes uma das outras, à excepção dos casos em que possa existir várias DIXOs dentro de uma DIM sendo que um retorno de uma pode ser o parâmetro de entrada da outra. Mas se tivermos várias DIMs constituídas com DIXOs, as quais a execução de uma DIM necessita de aceder às variáveis de outra DIXO que até já foi executada, ela não vai conseguir aceder e assim não consegue fazer o pretendido. Esta lacuna é importante na óptica do programador, uma vez que ele pode querer que um conjunto de DIMs estejam relacionadas entre si, para ter uma interface interactiva para o utilizador. Como é caso de podemos associar DIXOs a botões da interface, quando se carrega

num botão é executada uma DIXO, por exemplo o botão Next em que a DIXO correspondente permite avançar em alguma coisa, se o utilizador querer voltar atrás premindo a tecla Back, a qual está associado à DIXO Back que tem como função retroceder para o ponto anterior, vai precisar saber qual o estado do ponto em que está através duma variável da DIXO next. A DIXO back ao tentar aceder a essa variável para saber para que ponto retroceder, não lhe vai ser possível, uma vez que a DIXO já foi executa e não está em memória da aplicação, logo nunca poderíamos ter relacionamento em cada um destes dois Botões (duas DIXOs). Para colmatar esta lacuna a ideia foi criar no código da aplicação uma base de dados, ou seja, um ficheiro em Java com declaração de variáveis que funcionam como variáveis temporárias que o autor sabendo da existência delas possa utilizá-las e reutilizá-las para poder então relacionar as suas DIXOs.

7.2 – DIXO Validate

Uma das DIXOs desenvolvidas, foi uma DIXO para validar um DI actual, o que no caso da aplicação desenvolvida não faz muito sentido, visto que se um DI está a ser visualizado então é porque está validado, caso contrário, a aplicação bloqueava a interface. De qualquer maneira a DIXO é bastante simples de implementar, foi desenvolvida com base na API jDOM a qual permite validar os *schemas* e fazer o *parsing*, retornando para o utilizador se o DI é válido ou não, e se não retorna qual o erro de não dar e a linha onde se encontra o erro na DID do DI. A figura 7.2 mostra o resultado da execução da DIM Validate que contém esta DIXO.

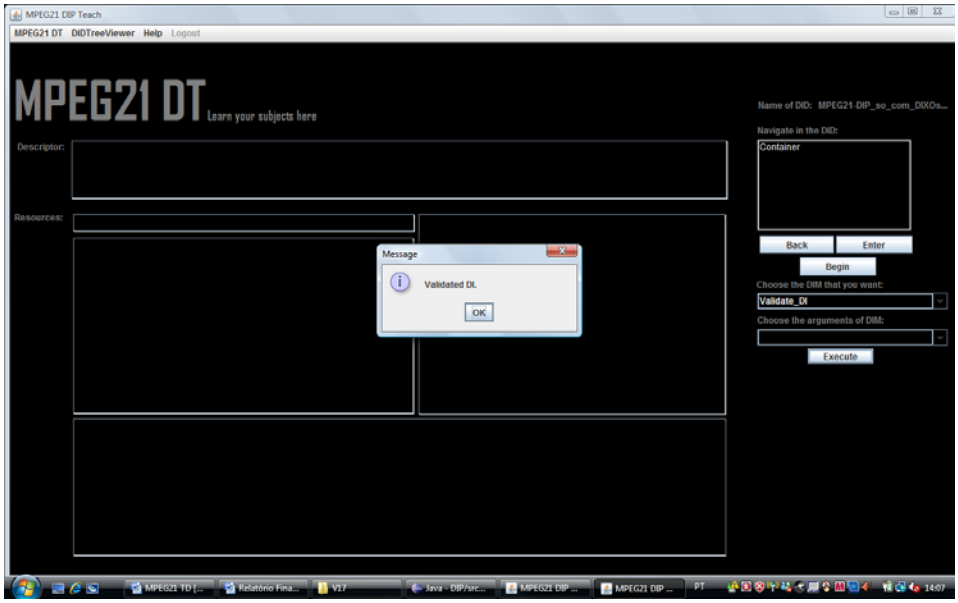


Figura 7.2 – Resultado da execução da DIXO Validate.

7.3 – DIXO Color

A DIXO Color é a responsável por permitir ao utilizador mudar as cores da sua aplicação através de uma interface que lhe é apresentada quando da sua execução. A DIXO chama a função que tem o código que permite mudar as cores da interface como foi visto porquê de ser assim nas limitações das DIXOs vistas neste capítulo. O resultado da execução desta DIXO

está ilustrada na figura 7.3 e o resultado final, após o utilizador mudar as cores está ilustrado na figura 7.4 segundo as cores escolhidas na figura 7.3.

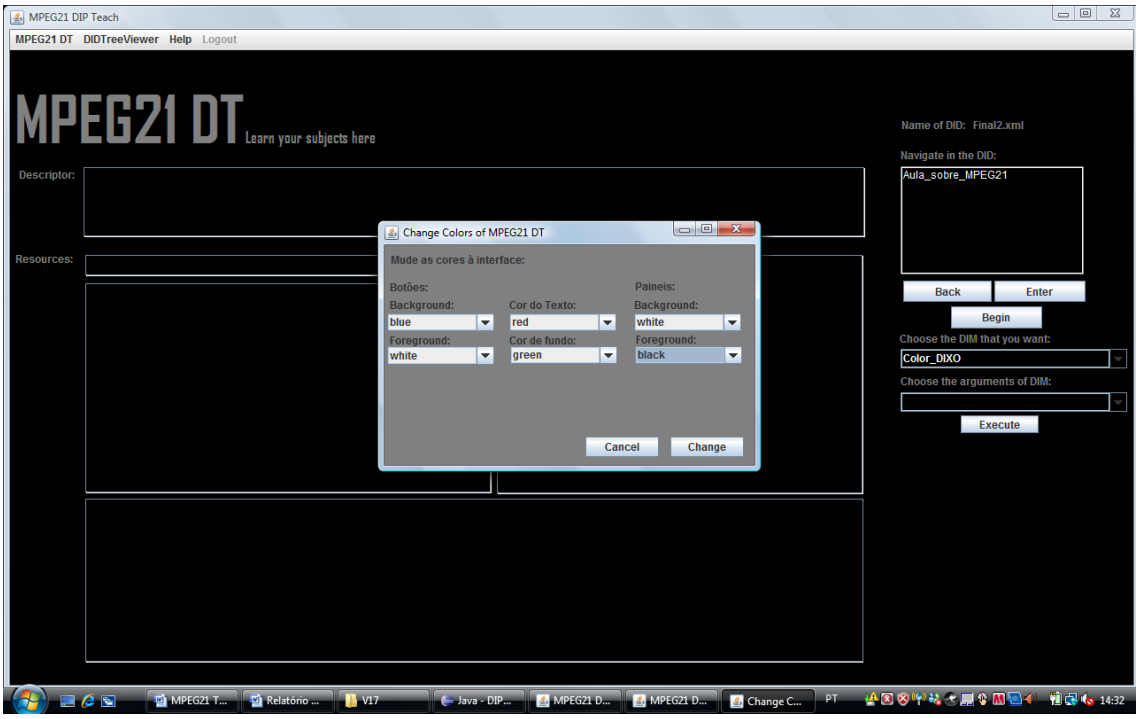


Figura 7.3 – Resultado da execução da DIXO que permite mudar as cores da interface.

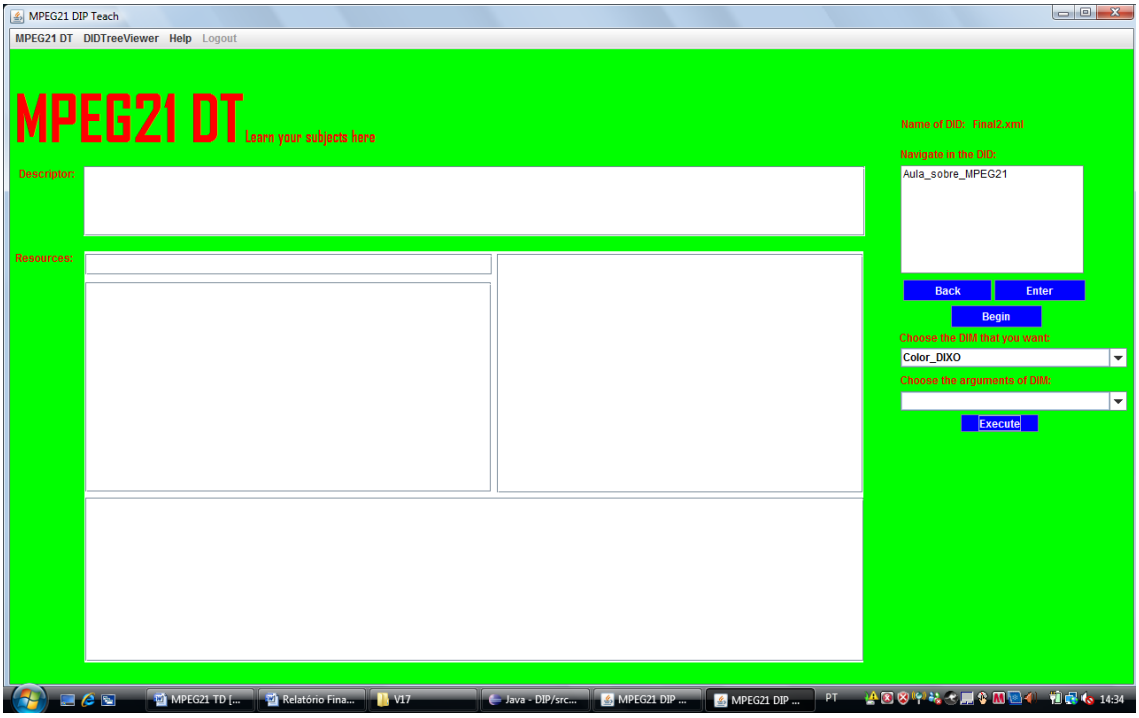


Figura 7.4 – Resultado final após a escolha das cores na DIXO que permite mudar as cores.

A nível de implementação da interface para mudar as cores mostrada na figura 7.3 é bastante simples, cada ***ComboBox*** tem um conjunto de opções que não são mais de que os nomes de cores, incluindo manter a mesma cor. Inicialmente quando a DIXO é executada cada

ComboBox tem definido manter a mesma cor, o utilizador pode então alterar a cor de *background* e f*oreground* de cada grupo de componentes de acordo com a sua preferência e carregar no botão ***Change*** que muda as cores da aplicação, mas não sai da interface das cores para dar a oportunidade de voltar a mudar as cores caso não gostasse das que escolheu.

Quando se carrega no botão ***Change***, ele vai ler qual a cor que está definida em cada ***ComboBox***, e com base nisso modificar os grupos de componentes para essas cores. Para sair da interface das cores carrega-se em ***Cancel*** ou no ***X***.

7.4 – DIXO Color_automatic

A DIXO color automatic muda as cores da interface para o aspecto que tem na figura 7.2, daí a aplicação ter agora essas cores uma vez que esta DIXO está incluída numa DIM do DI que está a ser analisado, e que está definida na DIM como automática.

7.5 – DIXO Email

A DIXO Email tem como finalidade o utilizador da aplicação enviar um email para um endereço de destino, que está definido como sendo um email fixo. O código está a funcionar, isto é, permite enviar um email para o endereço de destino, com o endereço de origem, o assunto e a mensagem e o nome da pessoa. Para concretizar isso usamos a API Java Mail [40], visto que a própria API pertence à Java Sun tal como o *Java Media Framework*, e devido á sua grande popularidade, e simplicidade de implementação, foi a escolha requerida para desenvolver a DIXO. O utilizador tem que preencher todos os campos para que seja possível o envio do email, se não é retornada uma mensagem de erro. A figura 7.5 mostra a interface para enviar o email, já com os campos preenchidos para enviar a mensagem, após ter mandado executar esta DIXO, e a figura 7.6 o email recebido, em que podemos observar os dados preenchidos incluídos na mensagem recebida. De salientar se o email for enviado com sucesso o utilizador é avisado, tal como se não for enviado (a conexão falhar).

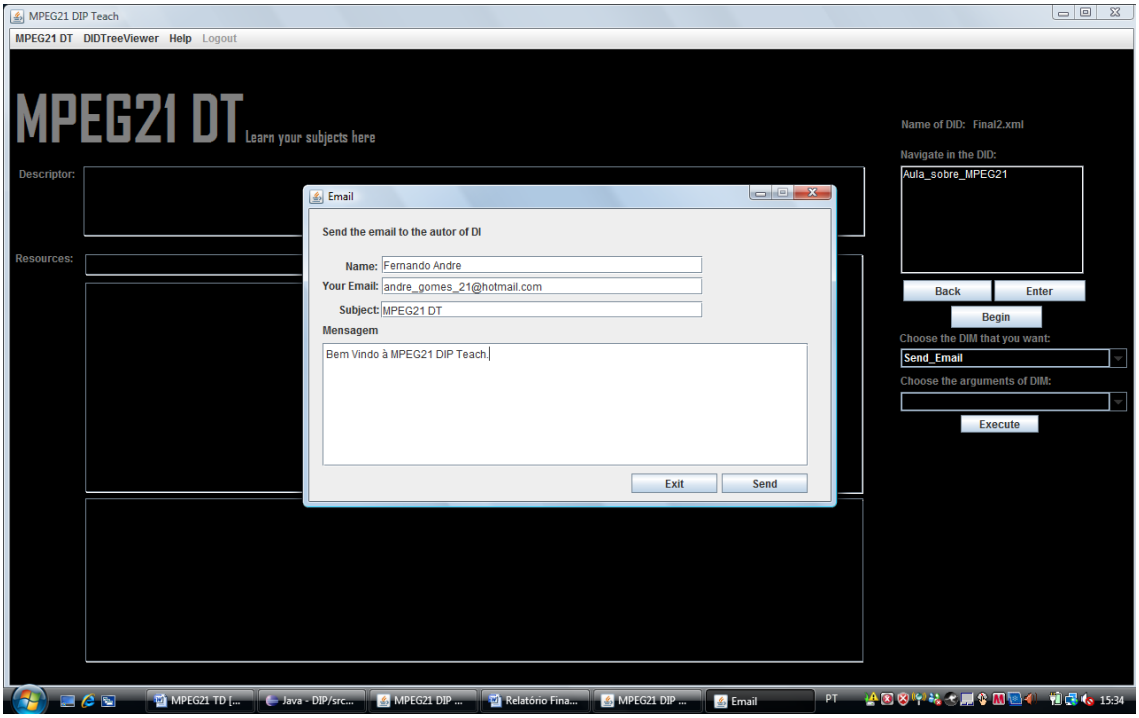


Figura 7.5 – Interface que resulta da execução da DIXO Email, mais concretamente da DIM Send_Email.

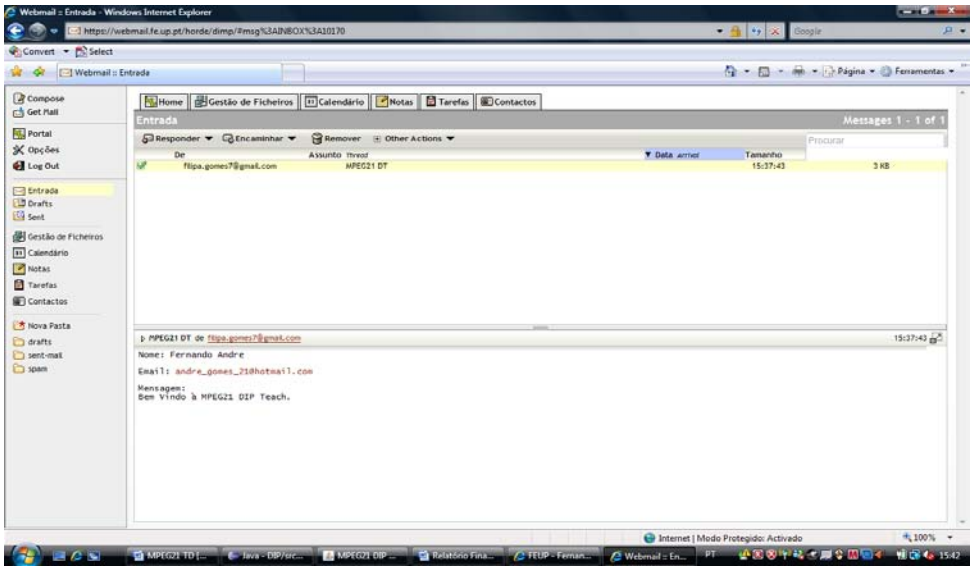


Figura 7.6 – Email recebido após o envio dos dados preenchidos na interface da figura 7.5.

7.6 – DIXOs para navegar na DID

Um dos desafios deparados no desenvolvimento da dissertação, foi mostrar que é possível navegar na DID através de DIXOs, ou seja, fazer o mesmo que a lista, o botão ***Enter***, o botão ***Back*** e o botão ***Begin*** fazem, mas com recurso a DIXOs e tendo como acrescento a capacidade de processar *Choices*. Além disso o objectivo passou por associar as DIXOs aos componentes acabados de referir, para que sempre que carregue neles execute a DIM que contém a DIXO correspondente ao componente.

Face a este cenário, foram construídas 4 DIXOs, uma para configuração da aplicação, outra para controlar a lista e o botão ***Enter***, outra para o ***Begin*** e outra para o ***Back***. Daí no capítulo 5 no módulo MPEG21DT, foi referido que havia duas maneiras de controlar esses componentes, uma através do módulo DIDEngine2 e outra através de DIXOs. Para isso foi usado uma variável que funciona como um comutador, a variável DIP. Enquanto ela está inactiva os componentes fazem ligação com o módulo DIDEngine2, quando estiver activa os componentes são controlados por DIXOs. Sendo que quando está activa, cada um desses botões vai estar associado a uma DIM, para que quando se carregue num desses componentes, ele execute a DIM que conterá a DIXO correspondente. O comutador muda de estado com base na DIXO desenvolvida para configurar, nesta mesma DIXO é '"dito" aos botões que DIM é que vão executar quando são pressionados.

Com base nisto criei então as 4 DIXOs as quais estão listadas abaixo:

- DIXO_Back – para o botão ***Back***
- DIXO_Show – para o botão ***Enter*** e lista
- DIXO_Begin – para o botão ***Begin***
- DIXO_Configure – Para controlo do comutador e dos nomes das DIMs associadas a cada componente.

Também devido ao facto de estas DIXOs necessitarem de saber o estado actual de processamento, ou seja, o elemento que está a ser analisado de momento, foi criada uma base dados, para que cada uma quando for executada, aceda lá para saber o estado corrente de processamento da DID, isto é, o actual elemento e actualizá-lo consoante o seu processamento.

As três primeiras DIXOs foram muito "fáceis" de fazer, foi só reaproveitar o código do DIDEngine2 desenvolvido e das acções dos botões quando a variável DIP está desactiva no módulo MPEG21DT. Apenas é explicado o procedimento para a DIXO Show, as outras duas (DIXO_Back e DIXO_Begin) seguem o mesmo raciocínio. Por exemplo para a DIXO_Show foi "copiar" o código desenvolvido para o botão ***Enter*** quando a variável DIP está desactiva e "colá-lo" na função "main" da DIXO_Show, e "copiar" todas as funções que desenvolvidas para a DIDEngine2 e "colá-las" na DIXO também para que durante a execução da DIXO possa-as aceder. Depois foi só foi alterar as variáveis que são acedidas pelas outras DIXOs para as variáveis temporárias, para que assim as DIXOs estejam sempre ao corrente do actual estado de processamento.

Relativamente ao processamento de *Choices*, por parte das DIXOs, só a DIXO_Show apresenta este tipo de processamento, uma vez que não faz muito sentido retroceder e aparecer uma *Choice*, o mesmo para a DIXO_Begin, em que quando volta-se ao inicio certamente que não se encontra uma *Choice*. Para isso acrescentei na função Analisa Elementos da DIXO_Show, para que quando encontre uma *Choice*, converta-a para DOM e chame então a DIBO ConfigureChoice para o utilizador seleccionar a opção *(Selection)* que pretende. Aí encontrou-se um problema, a dificuldade de chamar as DIBOs em Java, uma vez que sempre que chamamos a função da DIBO configureChoice, dá um erro no código dizendo

que essa função tem de estar definida em modo estático, mudando então a função da implementação da DIBO para estática, o erro na DIXO desaparece, mas por outro lado aparece erro na implementação da DIBO dizendo para retirar o modo estático, tendo como consequência o erro não desaparecer, andando sempre neste impasse. A solução foi passar a implementação da DIBO para a DIXO e aí já não dava erro. De salientar que este erro acontece para cada DIBO que é chamada em Java. Outra dificuldade encontrada a seguir foi que a DIBO areConditionsSatisfied, que testa se o utilizador seleccionou alguma coisa ou não, dava sempre falso, mesmo que tenha seleccionado, não podendo dar seguimento ao código que tentava construir com DIBOs. O facto de não saber se o erro se deve pelo facto de usar Java como linguagem de programação (uma vez que foi testado que nas DIDs essa DIBO funciona), daí não se poder dizer com toda a certeza que é impossível chamar as DIBOs em Java como são chamadas nas DIMs. Face a esta dificuldade foi pensado noutro tipo de raciocínio, basicamente fazer tudo à "mão" ou seja, usar a DIBO configureChoice na mesma, saber que *Selection* é que o utilizador seleccionou, ir a essa *Selection* e extrair o id da *Selection*, com base nesse id, ir procurar pela DID se existe uma *Condition* que requer esse id, se sim e se essa *Condition* engloba *Choices*, porque se englobar chama-as, fazendo o mesmo procedimento para saber que *Selection* é que seleccionou, se não é feito o *play* dos recursos associado a essa *Selection*. De salientar que se o utilizador seleccionar mais que uma *Selection* é só processada a primeira *Selection.*

Todavia o processamento da *Choice* não foi tão simples quanto isso porque existia a dificuldade de saber como as DIXOs desenvolvidas seriam feitas com o JDOM, acontecia que não se conseguia passar os elementos como argumento da DIBO configureChoice, uma vez que ela só aceita argumento do tipo DOM. A solução passou por converter os elementos de JDOM a passar à DIBO configureChoice para DOM usando uma funcionalidade da API JDOM. O problema desta conversão é que a última versão do JDOM não permitia a conversão de elementos JDOM para DOM, apesar de permitir documentos e atributos para DOM, sendo uma lacuna encontrada nesta API. Face a isto foi analisado de onde vinha o problema, e chegou-se à conclusão que era da última versão, sendo assim foram testados outras versões do JDOM que convertessem elementos para DOM, e foram encontradas 2 ou 3 versões de 2004 que funcionavam, só que tinham como inconveniente o problema de não validarem o DI. Face a este cenário e como o funcionamento daqueles problemas era imprescindível para a aplicação desenvolvida, o que foi feito, foi fundir as duas versões, ou seja, adicionar essa funcionalidade de conversão para DOM à nova versão, depois exportar tudo para um jar o qual designamos de JDOM, ficando com uma nova API JDOM com ambos os problemas resolvidos.

Por fim falta abordar a DIXO_Configure, esta começa por mostrar uma janela *popup* para que o utilizador configura a sua interface para que os componentes mencionados anteriormente sejam controlados por DIXOs ou então para configurar para o modo normal, ou seja, os componentes funcionam segundo a DIDEngine2. Se o utilizador escolheu o modo normal desactiva a variável DIP, limpa a lista e chama a função Analisa Elementos da DIDEngine2 vista no capítulo 5; se for a outra opção, então atribui o valor activo à variável DIP e fornece aos botões o nome da DIM que vão executar quando pressionados; após este passo limpa a lista e chama a função Analisa Elementos, que também como todas as outras funções da DIDEngine2 desenvolvidas foram "copiadas" para esta DIXO.

Após a explicação, falta mostrar os testes. De salientar que faz o mesmo que a DIDEngine2, à excepção de processar *Choices*, daí não ter que apresentar tudo o que acontece passo a passo, visto que já foi analisado e o modo de processamento é idêntico.

Se executarmos a DIXO Begin, ou a DIXO Back, ou a DIXO Show sem termos configurado os botões para serem processados por DIXOs através da DIXO Configure, é nos retornada a seguinte mensagem:

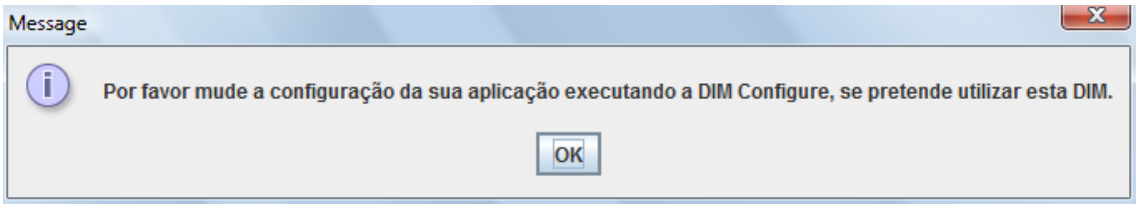


Figura 7.7 – Mensagem que é retornada, se executarmos a DIXO Begin, ou a DIXO Back, ou a DIXO Show sem termos configurado os botões para serem processados por DIXOs.

Então temos que executar primeiro a DIXO Configure, se a executarmos é nos mostrada a seguinte janela popup que nos permite então escolher o modo como queremos controlar os componentes:

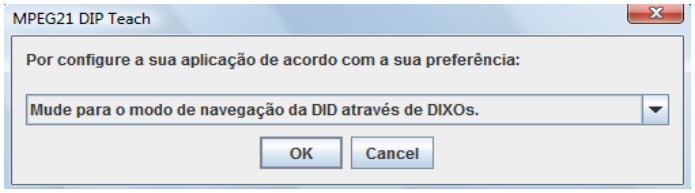


Figura 7.8 – Janela popup que aparece durante a execução da DIXO Configure.

Após termos escolhida a opção que vemos na figura 7.8 podemos então controlar os componentes através das DIXOs. Em baixo na figura 7.9 vemos alguns screenshots que resultam da execução destas DIXOs:

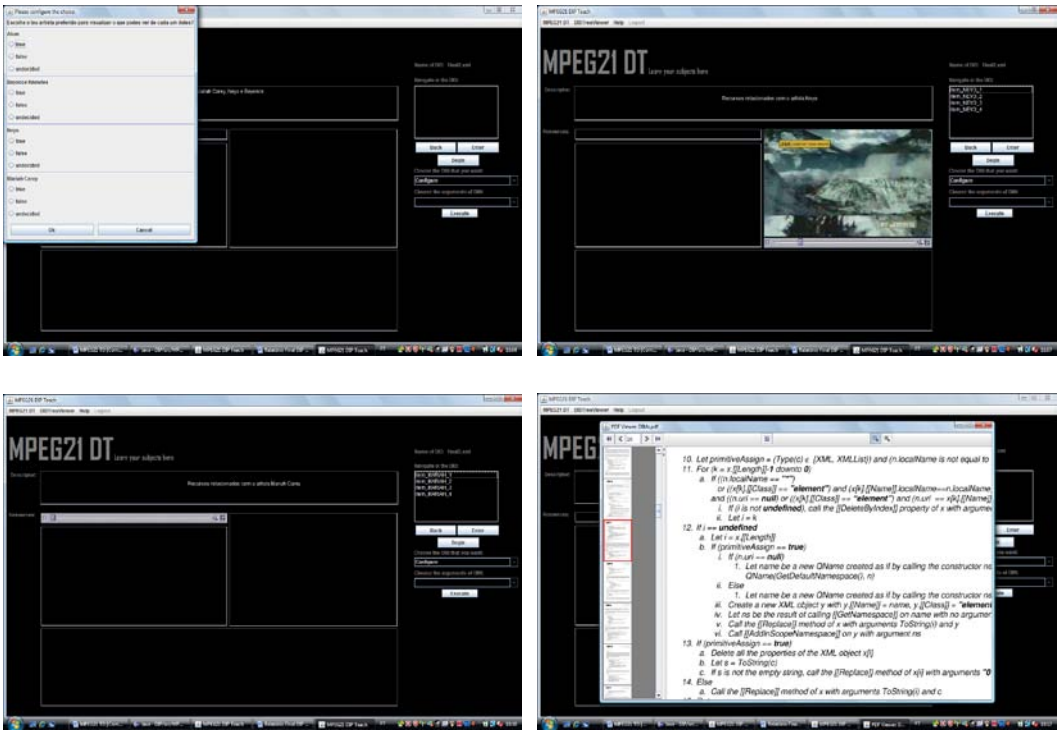


Figura 7.9 – Alguns screenshots da aplicação que ilustram que os botões quando processados por DIXOs funcionam.

7.7 – DIXO Component

Devido ao facto de existirem aplicações que reproduzem automaticamente *Components* com o mesmo pai e se tivermos em conta que existem DIs em que esses *Components* podem conter o mesmo recurso, só que com qualidades diferentes, não faz sentido nestes casos reproduzir todos os *Components*. Face a este cenário foi lançado o desafio de poder mostrar que é possível o autor do DI, definir o modo como é feito a reprodução desses *Components*.

No capítulo 5 foi abordado que a aplicação MPEG21 DIP Teach já está preparada para este cenário, ou seja, quando se faz o carregamento de um DI, a aplicação vai verificar se existe uma DIM chamada DIXO_Component na DID, se sim sempre que encontrar um *Component* à medida que navega nessa DID, executa a DIM DIXO_Component, se não reproduz o modo *default* que é reproduzir automaticamente os *Components* à medida que os encontra durante a navegação na DID. Relativamente à implementação desta DIXO, foi desenvolvida uma que apenas reproduz o segundo *Component* de um grupo de *Components* que tem o mesmo pai, outra para reproduzir os recursos desse grupo por ordem crescente, e outra por ordem decrescente.

7.8 – Conclusão

Neste capítulo vimos aspectos muito importantes sobre as DIXOs, principalmente no que diz respeito às suas limitações. Estas limitações encontradas levam a concluir que ainda existe muito trabalho a fazer para que este tipo de operações sejam totalmente transparente

para o autor, para que de algum modo as consiga desenvolver de acordo com aquilo que ele pretender, sem este tipo de constrangimentos. Isso leva-nos a crer que as DIXOs devem ter que ser revistas, no que diz respeito à interpretação do código delas pela aplicação, para que não apresente tantas restrições.

Apesar do trabalho desenvolvido que fazia parte da dissertação já estar todo explicado, vai ser mostrado no próximo capítulo as outras interfaces desenvolvidas, para o caso desta aplicação tiver que funcionar num "verdadeiro" cenário de Ensino à Distância.

Capítulo 8

Complementos MPEG21 DIP Teach

Neste capítulo serão descritas as interfaces desenvolvidas face ao cenário da aplicação desenvolvida ser Ensino à Distância e que funcionam como um complemento para esta aplicação, se bem que não estão ligadas. De salientar que estas interfaces desenvolvidas, não faziam parte dos objectivos da dissertação.

8.1 – Utilizadores

Os utilizadores da aplicação são os alunos, os professores e os administradores. Os alunos são os consumidores que poderão visualizar e interagir com os *Digital Items*. Os professores tem como objectivos colocar os conteúdos, os DIs, para os alunos poderem visualizar e eles próprios visualizarem. Os administradores têm acesso a todas as funções do sistema e por isso vai ser responsável pela manutenção do funcionamento correcto do sistema.

8.2 – Funcionamento dos Complementos da MPEG21 DT

Ao utilizar a aplicação desenvolvida os utilizadores são confrontados com um sistema de autenticação, para permitir um controlo e gestão não só da própria aplicação mas também dos privilégios dos diferentes tipos de utilizadores: os alunos, os professores e os administradores (Figura 8.1).

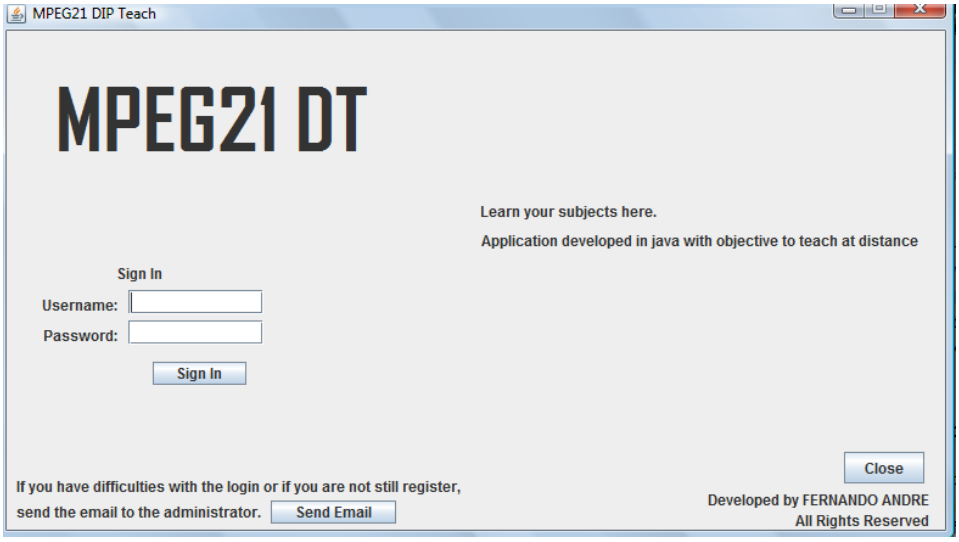


Figura 8.1 – Interface de autentificação dos utilizadores da MPEG21 DIP Teach.

Os utilizadores autenticam-se na interface com base no seu *username* que o distingue de forma inequívoca e respectiva *password*. Para poder prosseguir na aplicação tem que se autenticar correctamente. De salientar que se algo incorrecto acontecer como, por exemplo, a base de dados não estar acessível, são avisados que o servidor está indisponível de momento, tal como se o *login* estivesse incorrecto ou se esquecesse de preencher alguns dos campos também seriam avisados na respectiva interface. Este tipo de mensagens de erros e de confirmação aparecem em todas as interfaces referidas neste capítulo, daí que isso seja omitido na explicação das próximas interfaces.

Na base de dados será mantida uma tabela designada de autenticação, que permite como o nome revela a autenticação de utilizadores, e que apresenta os seguintes campos:

←T→	username	password	nome	categoria	ano_entrada	email
	Andre	Andre	Fernando Andre	pupil	2007	
	b	b	Helena Sofia Gomes Silva	pupil	2009	
	c	c	Maria Teresa Andrade e Pedro Miguel Carvalho	teacher	2070	
	DIP	DIP	Digital Item Processing	pupil	2002	DIP@hotmail.com
	Filipe	Filipe	Filipe	pupil	2005	filipe@hotmail.com
	h	h	Maria Teresa Andrade e Pedro Miguel Carvalho	teacher	2009	
	lena	lena	Helena Sofia Gomes Silva	pupil	2009	lena_silva@hotmail.com
	Marta	Marta	Marta Filipa Gomes Silva	pupil	2009	marta@hotmail.com
	MPEG21TD	Andre	Fernando Andre Gomes Silva	administrator	2004	ee04001@fe.up.pt
	orientador	INESC	Pedro Miguel Carvalho	teacher	2000	pedro.carvalho@inescporto.pt
	orientadora	INESC	Maria Teresa Andrade	teacher	1997	mandrade@fe.up.pt
	Vanessa	Vanessa	Vanessa	pupil	2004	vanessa@hotmail.com

Tabela 8.1 – Tabela autenticacao da Base de Dados.

A chave primária da tabela é o *username*, isto indica que não podem existir dois *usernames* iguais, evitando assim possíveis conflitos.

As funcionalidades dos outros campos serão explicadas mais à frente uma vez que aplicação recorre várias vezes a esta tabela, mas a maior parte dos campos servem para opções de filtragem.

O utilizador após autenticação correcta procede para a interface correspondente com base na sua categoria. Se for um caso de autentificação válida de um aluno a interface que se segue é a que está ilustrada na figura 8.2. Se for a dum professor é a que está na figura 8.3 e se for um administrador é a que está na figura 8.4.

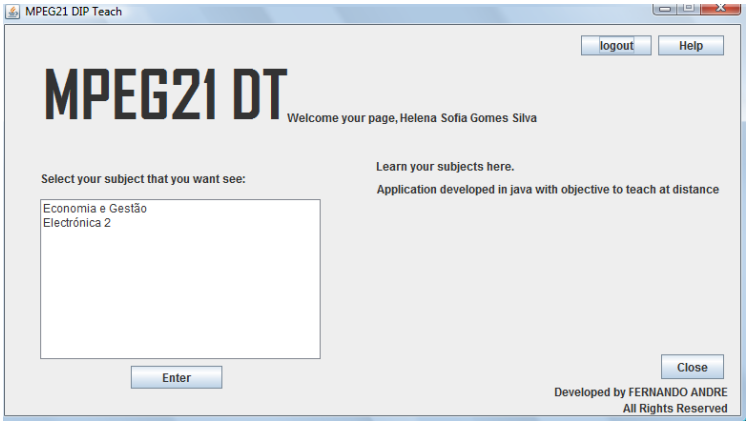


Figura 8.2 – Interface aluno.

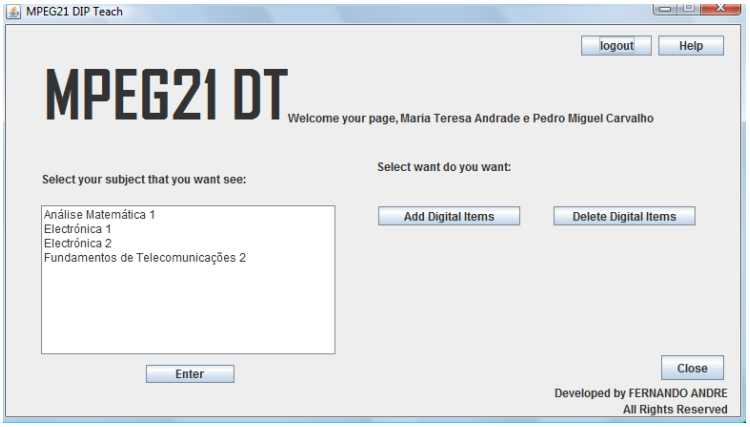


Figura 8.3 – Interface professor.

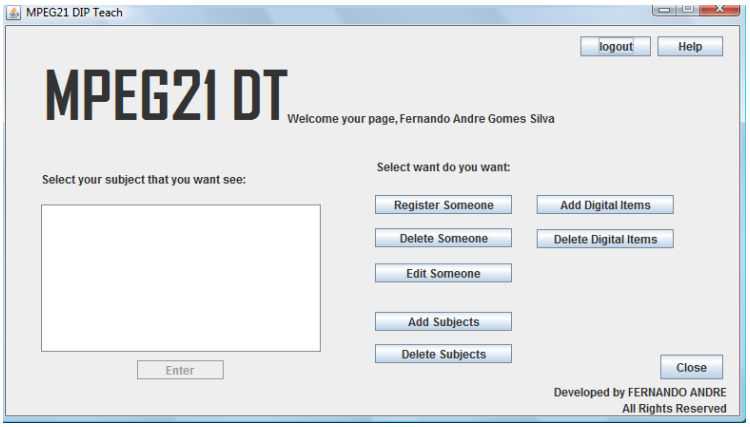


Figura 8.4 – Interface Administrador.

Se alguém se esqueceu do *login* ou se ainda não está registado e quer-se registar, poderá enviar um email ao administrador (botão ***Send Email***). A pessoa em causa poderá enviar um

email tendo como dados o seu nome, o seu email, o assunto e a mensagem em causa, como ilustra a figura 8.5.

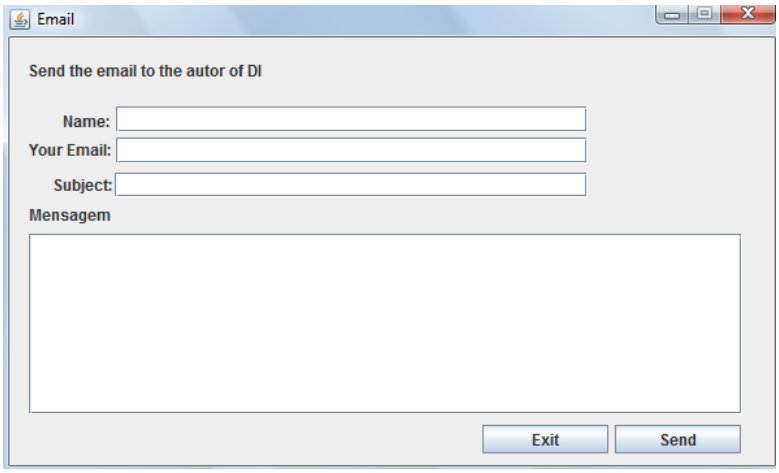


Figura 8.5 – Interface Email.

Após a autenticação de um aluno ele acederá à interface da figura 8.2 de onde poderá seleccionar a disciplina que pretende visualizar os seus conteudos. De salientar que as disciplinas são as que se encontra inscrito. Se não tiver inscrito em nenhuma disciplina é retornado ao utilizador que não se encontra inscrito em nenhuma disciplina e por isso põe os botões da interface *disable*. Após o aluno aceder uma disciplina para visualização, a intenção era ligar até à próxima interface que é a principal do sistema, ou seja, a interface vista no capítulo 5. Mas como disse na introdução estes complementos não estão ligados à interface principal da MPEG21 DIP Teach. De salientar que as disciplinas são listadas de acordo com *username* do aluno, tabela 8.2.

Tabela utilizadores_disciplinas:

username	subjects
c	Análise Matemática 1
c	Electrónica 1
c	Electrónica 2
c	Fundamentos de Telecomunicações 2
b	Economia e Gestão
b	Electrónica 2
h	Economia e Gestão
h	Electrónica 2

Tabela 8.2 – Tabela utilizador_disciplinas

Após a autenticação de um administrador ele será reencaminhado para a interface correspondente a ele (Figura 8.4) onde poderia visualizar as disciplinas que se encontra inscrito, visto que um administrador poderá ser um próprio professor, daí vir a poder ver as

disciplinas que lecciona. De salientar que um administrador além de poder visualizar as disciplinas pode registar alguém, apagar alguém, editar alguém, adicionar e apagar disciplinas, e adicionar e apagar *Digital Items*. As interfaces que o administrador tem acesso, após carregar nos botões correspondentes da interface da figura 8.4 e a sua explicação são referidas de seguida:

- Add subjects (adicionar disciplinas) – Esta interface que está ilustrado na figura 8.6 permite ao administrador adicionar disciplinas com base no nome da disciplina e no ano de entrada. Se não existir a disciplina, a disciplina vai ser inserida na tabela 8.3 da base de dados e ao mesmo tempo criado um directório no servidor com o nome dessa disciplina, onde serão os DIs armazenados juntamente com os recursos.

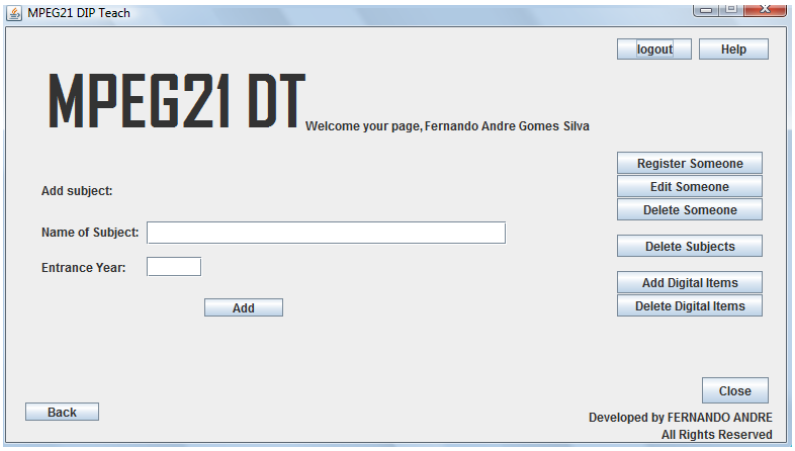


Figura 8.6 – Interface Add Subjects.

	subjects	ano_entrada_2
	Análise Matemática 1	2004
	Análise Matemática 2	2005
	Economia e Gestão	2008
	Electrónica 1	2004
	Electrónica 2	2007
	Fisica	2004
	Fundamentos de Telecomunicações 1	2006
	Fundamentos de Telecomunicações 2	2007
	Laboratório de Integração de Sistemas	2009
	Laboratório Multimedia	2007
	Preparação da Dissertação	2009
	Quimica	2004
	Redes de Computadores	2009
	Sistema de Telecomunicações	2006
	Sistemas Digitais	2004
	Televisão Digital e Novos Serviços	2008

Tabela 8.3 – Tabela disciplinas da Base de Dados.

- Register Someone (Registar alguém) – Nesta interface o administrador pode registar três tipos de utilizadores, alunos, professores e outros administradores. A interface correspondente está ilustrada na figura 8.7. Ao entrar nesta interface são listadas todas as disciplinas com base na tabela 8.3. Os campos obrigatórios aos quais o administrador, tem que preencher para registar alguém são o nome, *username*, *password*, categoria e ano (ano do registo da pessoa na aplicação), sendo o email e as disciplinas estando definidas como campos não obrigatórios. Após o preenchimento dos campos obrigatórios o administrador pode registar essa pessoa, fazendo o registo pressionando a tecla ***Save All Dates***. Ao guardar os dados será enviado um email do administrador para o email da pessoa que acabou de registar com as credenciais para usar a aplicação, caso tenha preenchido o campo de email. Os campos acima das listas serão guardados na tabela autenticação já referenciada anteriormente, as disciplinas que se encontram na lista da direita no momento de pressão do botão ***Save All Dates*** serão guardadas na tabela 8.2.

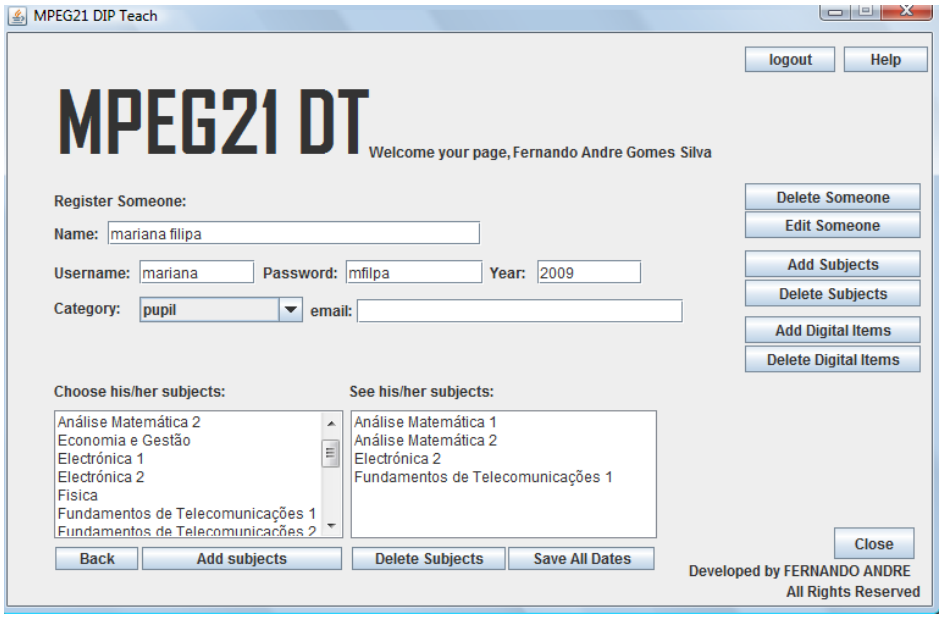


Figura 8.7 – Interface Register Someone.

- Delete Subjects (Eliminar disciplinas) – Ao entrar nesta interface é apresentado na lista todas as disciplinas que existem na base de dados com base na tabela 8.3 como ilustra a figura 8.8. O utilizador ainda tem a opção de filtrar as disciplinas com base no ano de entrada de funcionamento da disciplina. Para eliminar a disciplina selecciona-se a que se pretende na lista, e carrega-se no botão ***Delete***, ao fazer isto o administrador estará a apagar a disciplina da base de dados, bem como a inscrição dos utilizadores a essa disciplina e ainda a pasta da disciplina no servidor que poderá conter DIs associados e outros recursos, sendo isto tudo apagado.

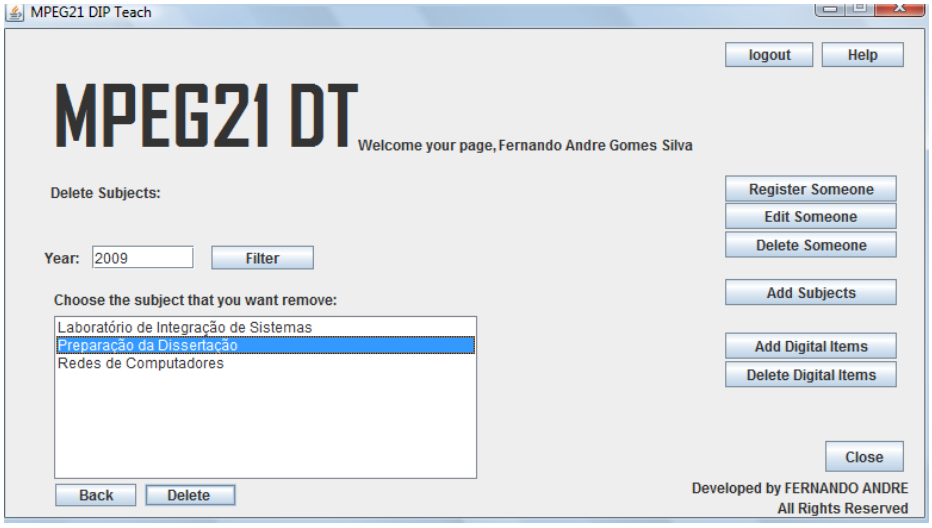


Figura 8.8 – Interface Delete Subjects.

- Delete Someone (Eliminar alguém) – Ao aceder a esta interface são listados os nomes de todas as pessoas que existem na base de dados com base na tabela 8.1. Se o utilizador preferir pode filtrar a pessoa com base no seu nome ou ano de entrada, ou a sua categoria (figura 8.9). Ou então pode filtrar com mais de uma opção activada, ou mesmo todas, sendo que a resposta da filtragem é apresentada na lista. Para remover a pessoa basta então seleccionar a pessoa requerida e carregar no botão ***Delete***, a consequência desta acção é que todos os dados relativos à pessoa serão apagados das tabelas da base de dados.

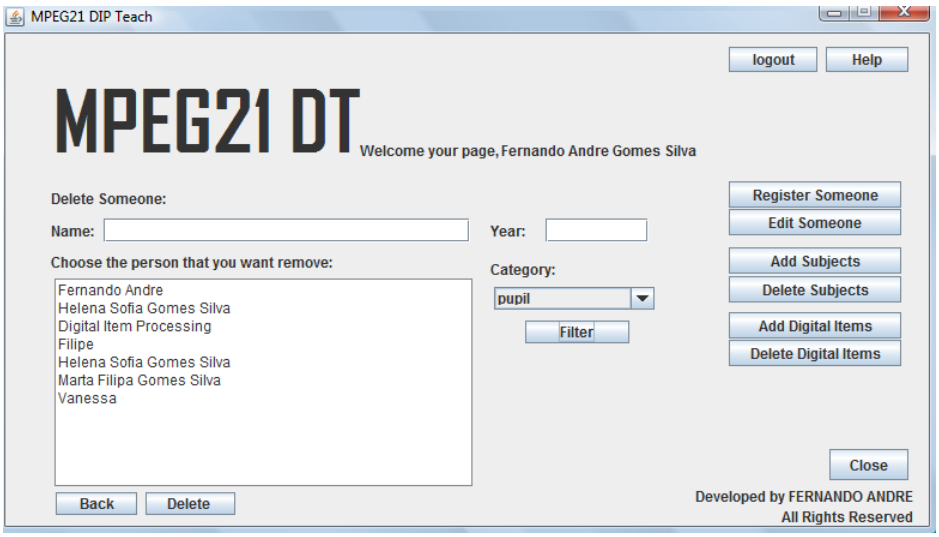


Figura 8.9 – Interface Delete Someone.

- Filter Someone (Filtrar Alguém) – Esta interface é acedida quando o administrador pretende editar os dados relativos a alguma pessoa, como os seus dados pessoais, as credenciais de acesso ou /e as disciplinas a que estão inscritos, carregando no botão ***Edit Someone***. O objectivo desta interface, Filter Someone, é poder filtrar os nomes que existem na base de dados para a possível escolha da pessoa que se pretende editar. Para isso o programa começa por listar todas as pessoas existentes no sistema na lista que se encontra disponível na respectiva interface (figura 8.10). Como já foi referido na interface Delete Someone o administrador poderá filtrar o nome da pessoa requerida com base no mesmo procedimento que foi aplicado a essa interface. Para editar a pessoa que pretende selecciona-a na lista e carrega no botão ***Edit*** que acederá á interface de edição propriamente dita.

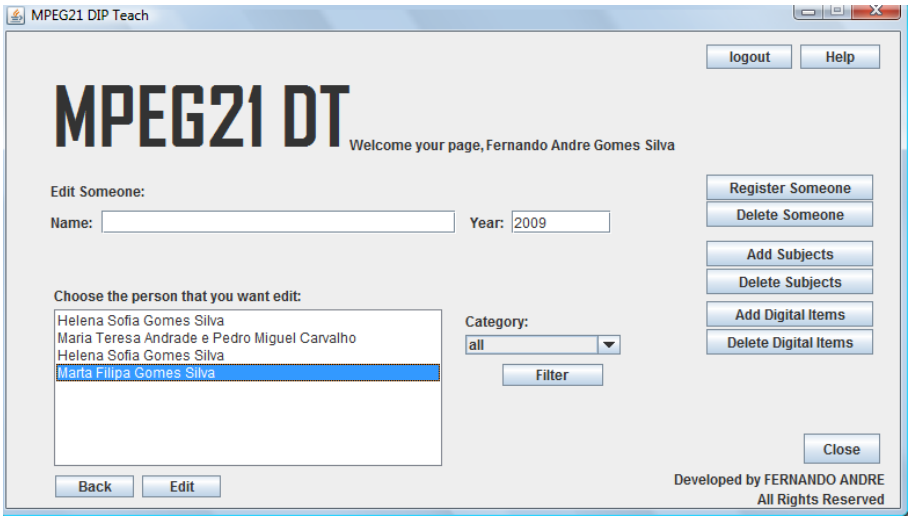


Figura 8.10 – Interface Edit Someone.

- Edit Someone (Editar Alguém) – Após ter escolhido a pessoa na interface anterior que se quer editar, o administrador acede à interface Edit Someone, que permite editar os dados da pessoa escolhida. Esta interface tem um funcionamento parecido em termos de implementação prática à interface Register Someone, à excepção de que os campos são preenchidos com os dados requeridos a esse mesmo de determinado utilizador como ilustra a figura 8.11. O administrador com base na informação disponibilizada acerca de determinada pessoa, pode alterar aquilo que pretende, quer seja o *username, password*, anular inscrição em disciplinas, inscrever-se em outras disciplinas, mudar de categoria, etc, desde que siga as regras relativas a essa interface tal como os campos obrigatórios a preencher, o *username* ser diferente de todos os que se encontram na base de dados, ter inserido um ano válido no respectivo campo, etc. Após ter editado alguém com sucesso, é enviado um email ao utilizador com as novas credenciais, se o campo de email estiver preenchido.

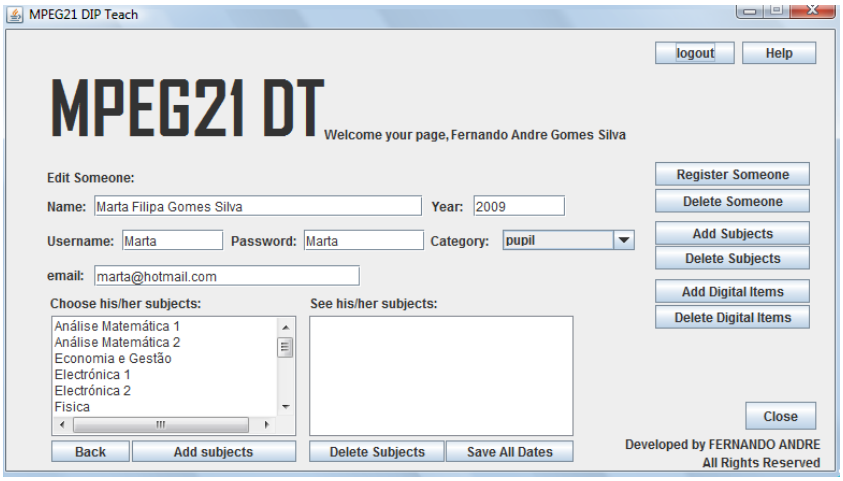


Figura 8.11 – Interface Edit Someone.

- Add Digital Items (Adicionar DIs) – Ao entrar nesta interface o administrador pode adicionar *Digital Items* ás disciplinas que pretende. Como tal as disciplinas são listadas na lista que se vê no lado esquerdo (figura 8.12). Para adicionar a uma disciplina o DI, administrador pode começar por procurar o DI no seu computador que quer adicionar carregando na tecla ***Browse*** e adicioná-lo; após esse passo selecciona a disciplina que pretende inserir o DI para que os alunos inscritos a ela possam visualizá-lo; e depois carrega na tecla ***Save*** para enviar para o servidor o respectivo DI. Quando a operação acabar será avisado disso mesmo, avisando se teve sucesso na operação que executou.

Figura 8.12 – Interface Add Digital Item.

- Delete Digital Items (Eliminar Digital Items) – O administrador pode nesta interface eliminar DIs às disciplinas que pretende (figura 8.13). Para isso o professor selecciona a disciplina que quer aceder na lista da esquerda, carrega em ***Browse*** que listará todos os DIs que se encontram na pasta da disciplina que seleccionou, selecciona um e carrega em ***Delete***. Após ter seguido estes passos será avisado se tal operação foi concretizada com sucesso.

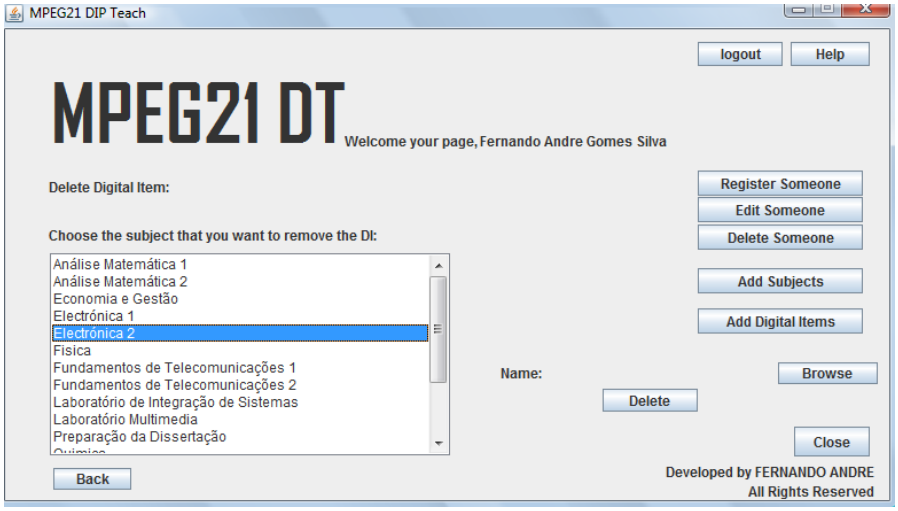


Figura 8.13 – Interface Delete Digital Item.

Após autenticação válida de um professor o professor será conduzido para a interface que está ilustrada na figura 8.3, poderá também visualizar os conteúdos das disciplinas que lecciona com o mesmo tipo de procedimento já referido para a interface alunos, além de poder adicionar e apagar *Digital Items* às disciplinas que lecciona. As duas interfaces que são apresentadas ao utilizador quando ele carrega no botão respectivo na interface da figura 8.3, são explicadas de seguida:

- Add Digital Items (Adicionar DIs) – O procedimento é o mesmo que a interface Add Digital Items do administrador, à excepção de que o professor só pode adicionar *Digital Items* apenas às disciplinas que lecciona ou que se encontra inscrito (figura 8.14).

Figura 8.14 – Interface Add DI Professor.

- Delete Digital Items (Eliminar DIs) – O procedimento é o mesmo que a interface Delete Digital Items do administrador, à excepção de que o professor só pode nesta interface eliminar DIs apenas às disciplinas que lecciona (figura 8.15).

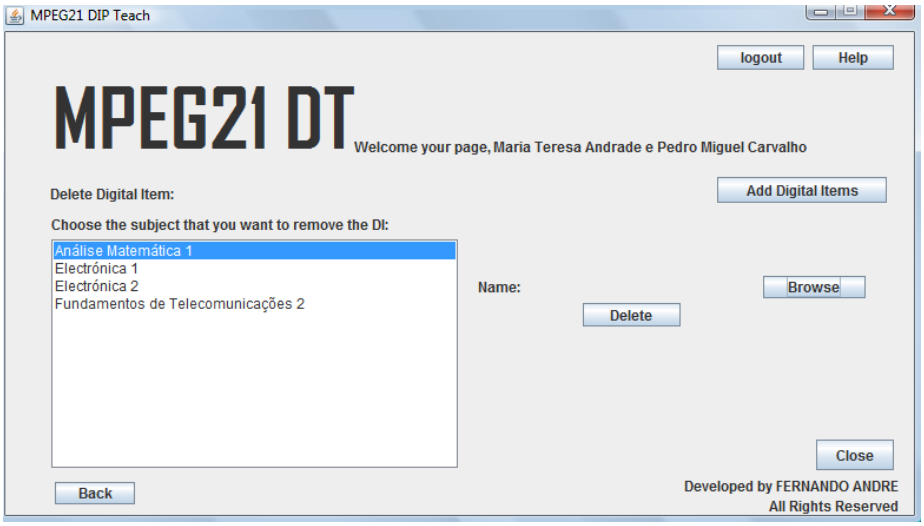


Figura 8.15 – Interface Delete DI Professor.

De salientar que quando referi ao servidor ao longo deste capítulo, estou a considerar o servidor na própria máquina onde reside a aplicação, tal como a base de dados está referenciada no endereço localhost do computador onde se encontra a aplicação.

Botões Gerais:

- Botão ***Logout*** – Permite aceder a interface de autenticação.
- Botão ***Back*** – Permite voltar à página principal de cada utilizador (menu de opções) de acordo com a categoria a que a pessoa se autentificou.
- Botão ***Close*** – Permite sair da aplicação a qualquer momento.

8.3 – Conclusão

Ao longo deste capítulo foram mostradas as interfaces e a sua explicação em termos de funcionamento, que podem vir eventualmente no futuro, funcionar em ligação com a interface principal e com um servidor remoto, tendo assim uma "verdadeira" aplicação de Ensino à Distância a funcionar em pleno.

O próximo e último capítulo abordará a conclusão final de todo o trabalho desenvolvido ao longo desta Dissertação.

Capítulo 9

Conclusão

ISO e IEC são duas organizações, que a par com outras organizações governamentais e não governamentais, trabalham em conjunto no desenvolvimento de normas internacionais, através de comités técnicos estabelecidos pelas mesmas. Através do grupo técnico SC29-WG11, conhecido sob a designação de MPEG *(Moving Pictures Expert Group)*, este organismo desenvolve especificações na área das aplicações multimédia. A norma MPEG-21 é um exemplo de uma especificação em desenvolvimento no seio deste grupo.

MPEG-21 está dividido em várias partes, dentro das quais destaco a parte 2 *(Digital Item Declaration)* e a parte 10 *(Digital Item Processing)* como relevantes no desenvolvimento da dissertação. A norma é baseada em dois conceitos essenciais: a definição da unidade fundamental de distribuição e transacção de informação multimédia *(Digital Items)* e o conceito da interacção com os *Digital Items* por parte do utilizador.

Digital Items podem ser entendidos como por exemplo, uma colecção de vídeo, um álbum de música ou outros tipos de conteúdos multimédia. Os *Digital Items* (DIs) são declarados numa estrutura estática DID através de uma linguagem própria (DIDL), num documento em formato XML como revela a parte 2 da norma. Um dos principais objectivos da norma é manipular os *Digital Items* de forma eficiente, transparente e de modo interoperável através das tecnologias adequadas para o seu acesso, consumo ou até mesmo comércio.

O MPEG-21 visa a utilização de recursos multimédia de forma transparente e crescente, através de uma larga gama de redes e dispositivos, usados por diferentes comunidades. Para manipular estes *Digital Items* a parte 10 da norma (DIP), disponibiliza um conjunto de métodos designados por DIMs que permitem a interacção do utilizador com DI. Desta forma permitem estender a definição estática do *Digital Item* fornecendo mecanismos que permitem manipular o DI de uma forma dinâmica. Esses métodos podem ser exprimidos através da linguagem definida em DIML, e são declarados juntamente com os recursos e metadados na estrutura DID referida anteriormente.

As DIMs podem conter chamadas para DIBOs e DIXOs, que são operações que permitem a interacção do utilizador com o DI. A diferença entre as DIBOs e DIXOs reside no facto de que as DIBOs estão normalizadas, ou seja, o programador tem que implementar o código de acordo com a norma, não podendo modificar a sua interface, havendo assim restrições. Nas

DIXOs, o programador tem mais liberdade para programar, podendo definir a sua interface e a sua implementação, devendo no entanto seguir algumas restrições impostas pela norma.

Durante estes meses em que trabalhamos para esta norma, chegámos à conclusão que é possível concretizar aquilo que se refere na norma, nomeadamente na parte 10, através de testes realizados. É possível implementar DIBOs e DIXOs que permitem fazer com que o autor permita adicionar interactividade aos conteúdos do DI de acordo com a sua criatividade, bastando ter apenas alguns conhecimentos de programação. É possível através do uso da DIP executar recursos de forma personalizada, proteger recursos com licenças, imprimir determinados recursos, filtrar determinados recursos que o autor pretende, ou seja, é possível visualizar tudo o que a DID contém em termos de metadados e recursos recorrendo à DIP, de outra forma não é possível garantir esta transparência e flexibilidade no acesso aos conteúdos da DID. Por exemplo com a DIP podemos escolher, por exemplo, se queremos visualizar um determinado vídeo através da aplicação desenvolvida ou então pelo *windows media player*, ou um VLC, de acordo com o que o autor estabelecer, ou seja, podemos adicionar funcionalidades que a aplicação não tem através do desenvolvimento de DIXOs. Sendo esta uma das principais vantagens em relação às DIBOs é que na DIBO a implementação é "fixa", se bem que pode ser alterada, por exemplo, no caso da DIBO play o recurso seleccionado será reproduzido sempre da mesma forma das outras vezes que for seleccionado. Enquanto com uma DIXO o autor pode mudar a implementação constantemente, hoje podemos reproduzir num *player* adequado ao tipo de DID a ser visualizado, amanhã podemos visualizar noutro. E é claro, com estas vantagens podemos construir DIXOs para colmatar DIBOs que não existem e se calhar seriam necessárias para a norma, além de podermos melhorar a acção de alguma delas em aspectos menos eficientes das DIBOs sendo que agora através de DIXOs, permitindo deste modo estender as funcionalidades das DIBOs. Outra coisa a salientar é que com a DIP é possível reproduzir diferentes tipos de recursos ao mesmo tempo, ex: Powerpoint, vídeo, áudio, imagem, texto/plain.

Um dos objectivos da dissertação e que foi mostrado com a aplicação, é que é possível ter duas máquinas diferentes, com a mesma aplicação, a aceder ao mesmo *Digital Item* ao mesmo tempo e ter diferentes formas de visualização do mesmo conteúdo através da DIP, consoante a "pilha" DIP que cada máquina contém. Também o facto de ter ficado mostrado que é possível usar DIXOs para navegar na DID e para reproduzir recursos da maneira que o autor da DID pretender, fornecendo um maior dinamismo à interpretação do *Digital Item*. Outros dos aspectos que tivemos em conta no desenvolvimento da dissertação, foi o desenvolvimento de uma DIXO que permite ao autor definir como os *Components* são reproduzidos, fornecendo assim uma alternativa aos problemas encontrados pelo grupo AXMEDIS e Enthrone.

Para concluir, os problemas mencionados das outras aplicações desenvolvidas no capitulo 3 e que de certa forma conduziram ao desenvolvimento desta dissertação, ficaram mostrados durante esta dissertação que são possíveis de se resolver usando *Digital Item Processing*.

Trabalho futuro incidirá eventualmente no desenvolvimento adicional de um conjunto de APIs para as DIBOs e na continuação do estudo e desenvolvimento de DIXOs, para que consiga ultrapassar as limitações com que nos deparamos ao longo do trabalho, bem como contribuir com novas soluções para problemas encontrados ao nível da interpretação do DI. Também gostariamos de desenvolver uma aplicação deste tipo e/ou usar a que desenvolvemos, para

dispositivos móveis, bem como usar outro tipo de linguagem para implementar DIP, para que tenha uma aplicação com efeitos e animações, como actualmente as aplicações e as páginas web estão providas.

Anexos

Lista de DIBOs da norma MPEG-21:

DIA *(Digital Item Adaptation)*:

- adapt(component,metadata);

DID *(Digital Item Declaration)*:

- areConditionsSatisfied(element);
- configureChoice(choice);
- setSelection(selection, state);

DII *(Digital Item Identification)*:

- getElementsByIdentifier(sourceDID, value);
- getElementsByRelatedIdentifier(sourceDID, value);
- getElementsByType(sourceDID,vaule);

DIP *(Digital Item Processing)*:

- alert(message,messageType);
- execute(element);
- getExternalData(mimeType, requestMessages);
- getObjectMap(document);
- getObjects(objectTypes, requestMessages);

- getValues(dataTypes, requestMessages);
- play(element, assync);
- print(element);
- release(playStatus);
- runDIM(itemIdType, itemId, componentIdType, componentId, arguments);
- wait(timeInterval);
- runJDIXO(itemIdType, itemId, componentIdType, componentId, arguments);

REL *(Rights Expression Language)*:

- getLicense(resource);
- queryLicenseAuthorization(license, resource, rightNs, rightLocal, additionalInfo);

DIP Error:

- getDIPErrorCode();

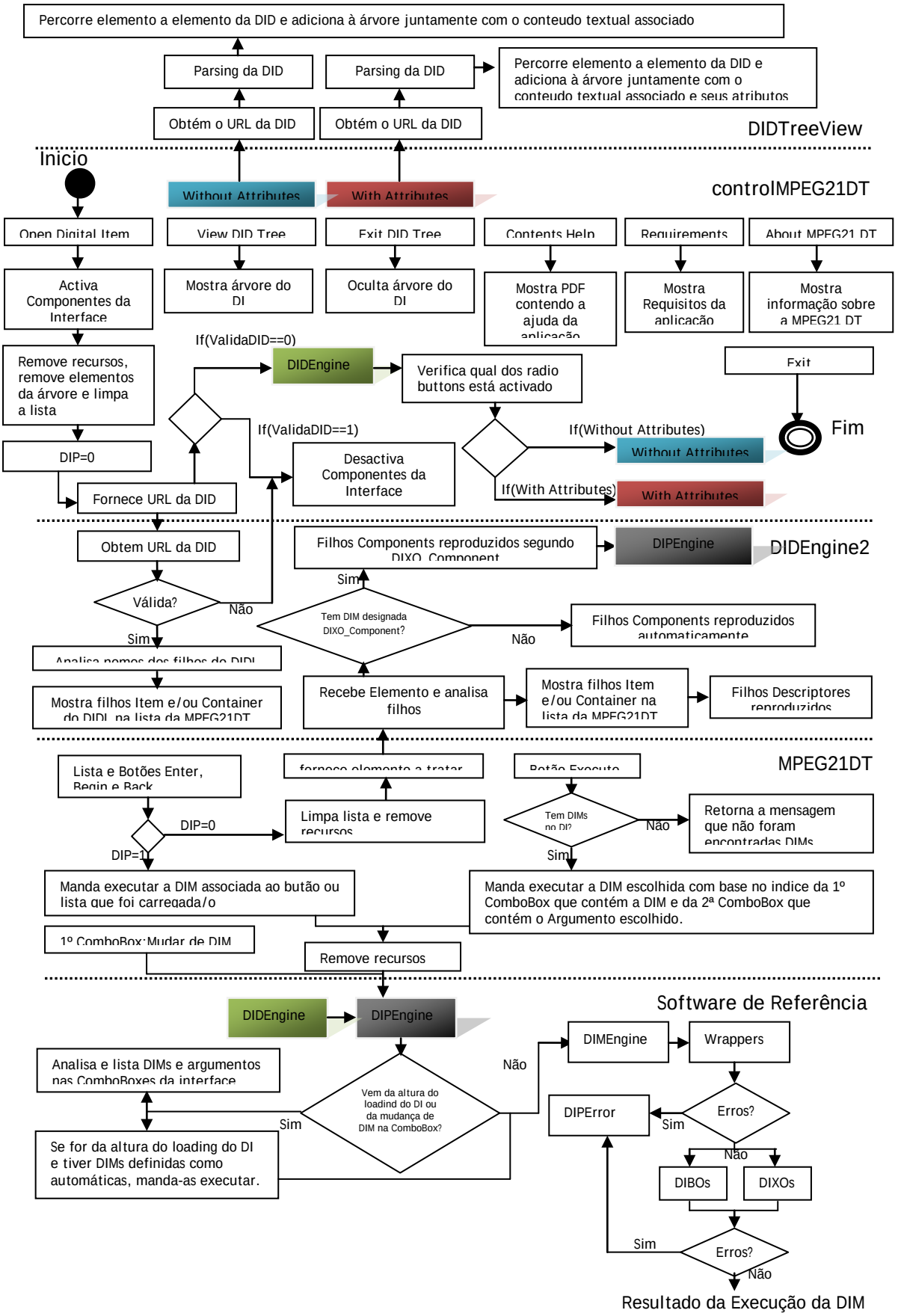
ObjectMap:

- getArgumentList(index);
- getArgumentListCount();
- getMethodCount(argumentList);
- getMethodWithArgs(argumentList);
- getMethodWithArgs(argumentList, index);
- getObjectOfType(typeName, index);
- getObjectsOfType(typeName);
- getObjectsOfTypeCount(typeName);
- getObjectTypeCount();
- getObjectTypeName(index);

PlayStatus:

- getStatus();

Fluxograma completo da aplicação MPEG 21 DIP Teach:



As cores existentes (excluindo o branco) em algumas caixas no diagrama significam que a caixa que tiver uma determinada cor, é como se chama-se a outra caixa que tem a mesma cor.

Referências

[1] ISO/IEC FDIS 21000-2:2005, *Information technology – Multimedia framework (MPEG-21) – Part 2: Digital Item Declaration*

[2] ISO/IEC FDIS 21000-10:2005, *Information technology – Multimedia framework (MPEG-21) – Part 10: Digital Item Processing*

[3] ISO/IEC TR 21000-1:2004, *Information technology – Multimedia framework(MPEG-21) – Part 1: Vision, Technology and Strategy*

[4] Rui Filipe Santos Rocha*, Digital Item Processing,* Dissertação de MIEEC na FEUP

[5] Giorgiana Ciobanu, *MPEG21 DI Browser, an MPEG-21 based architecture for the consumption of Digital Items*, Dissertação em Tecnologia Multimédia na FEUP

[6] ISO/IEC 21000-10, Introduction MPEG-21 DIP an Overview, *Information technology – Multimedia framework (MPEG-21) – Part 10: Digital Item Processing,* July 2005, Poznan, PL

[7] MDS Group, *Introducing MPEG-21 DIP – an Overview*, April 2005, Busan, KR

[8] IEEE, *Introduction to the Special Section on MPEG-21,* Transactions on Multimedia, VOL. 7,NO. 3, JUNE 2005

[9] Ghent University, Elis Department, Multimedia Lab, *Digital Item Processing Architecture,* October 2002, Shanghai, China

[10]IEEE, *MPEG-21 Digital Item Processing*, Frederik De Keukelaere, Saar De Zutter, and Rik Van de Walle

[11]INESC Porto, *An MPEG-21 Web Peer for the consumption of Digital Items*, Giorgiana Ciobanu, Maria Teresa Andrade, Pedro Carvalho, Eurico Carrapatoso, 22 junho de 2007

[12]http://www.adactus.no/ Adactus, Abril de 2009-04-22

[13]http://www.ist-world.org/ProjectDetails.aspx?ProjectId=734cf5c8a2f54268838d3cebd8b8ae40 Muffins, Abril de 2009-04-22

[14]http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm MPEG, Abril de 2009-04-22

[15]http://multimedialab.elis.ugent.be/demo/DIPImplementation/terminal/MPEG-21_DIP_terminal.html Demo da Multimedia Lab, Abril de 2009-04-22

[16]http://www.enikos.com ENIKOS, Abril de 2009-04-22

[17]http://www.axmedis.org/ AXMEDIS, Abril de 2009-04-22

[18]http://www.axmedis.org/tiki/tiki-index.php?page=AXMEDIS+Player#AXMEDIS_players Players AXMEDIS, Abril de 2009-04-22

[19]http://danae.rd.francetelecom.com/project-technology.php DANAE, Abril de 2009-04-22

[20]http://danae.rd.francetelecom.com/index.php DANAE, Abril de 2009-04-22

[21]http://danae.rd.francetelecom.com/technology-mpeg21.php DANAE, Abril de 2009-04-22

[22]http://multimedialab.elis.ugent.be/demo.asp Multimedia Lab, Abril de 2009-04-22

[23]http://www.prweb.com/releases/2003/11/prweb89412.htm ENIKOS, Abril de 2009-04-22

[24]http://www.ist-enthrone.org/ Enthrone 1, Junho de 2009-06-19

[25]http://www2.inescporto.pt/utm/projecto/em-curso/projecto-enthrone/?searchterm=terminals Enthrone 2, Junho de 2009-06-19

[26]*A Multimedia Terminal For Adaptation And End-To-End QOS Control*, Beilu Shao, Marco Mattavelli, Daniele Renzi, Maria T. Andrade, Stefano Battista,Samuel Keller, Giorgiana Ciobanu, Pedro Carvalho

[27]*Broadcasting and Communication Convergent Network Based on MPEG-21: Design and Implementation of Multimedia Service Framework*, Yongju Cho, Jae-Gon Kim, and Jin Woo Hong

[28]*MPEG-21 Digital Item Declaration (ISO/IEC 21000-2): an overview*, Jeroen Bekaert and Herbert Van de Sompel, Research Library, Prototyping Team, Los Alamos National Laboratory

[29]*Dynamic and Distributed Adaptation of Scalable Multimedia Content in a Context-Aware Environment*, Michael Ransburg, Hermann Hellwagner, Renaud Cazoulat, Benoit Pellan, Cyril Concolato, Saar De Zutter, Chris Poppe, Rik Van de Walle, Andreas Hutter

[30]*Advanced Multimedia Systems using MPEG-21 Digital Item Processing*, Chris Poppe, Frederik De Keukelaere, Saar De Zutter, Rik Van de Walle

[31]Standard ECMA-357, *ECMAScript for XML (E4X) Specification*, 2ª Edição / Dezembro de 2005

[32]http://java.sun.com/ Java Sun, Abril de 2009-04-20

[33]JavaOne, *JDOM Makes XML Easy*, Sun's 2002 Worldwide Java Developer Conference, by Jason Hunter,Co-Creator JDOM Project

[34]JDOM Makes XML manipulation in Java easier than ever, *JDOM and XML Parsing*, by Jason Hunter

[35]O'Reilly Open Source Convention 2001, *JDOM: How it Works, and How It Opened the Java Process*, by Jason Hunter, July 2001

[36]http://www.jdom.org/ JDOM, Abril de 2009-04-20

[37]W3C, *Document Object Model (DOM) Level 3 Core Specification*, Version 1.0, W3C Recommendation 07 Abril de 2004

[38]http://java.sun.com/javase/technologies/desktop/media/jmf/reference/api/index.html Java Media Framework, Abril de 2009-04-20

[39]http://java.sun.com/docs/books/tutorial/ Java Tutoriais, Abril de 2009-04-20

[40]http://java.sun.com/products/javamail/javadocs/index.html API JavaMail, Abril de 2009-04-20

[41]http://java.sun.com/products/javamail/ API JavaMail, Abril de 2009-04-20

[42]https://pdf-renderer.dev.java.net/ PDF Renderer, Abril de 2009-04-20

[43]http://www.w3schools.com/media/media_mimeref.asp Lista de MimeTypes, Abril de 2009-04-20

[44]http://www.itscj.ipsj.or.jp/sc29/29w42911.htm#MPEG-21 Conteúdos da norma MPEG 21, Abril de 2009-04-20

[45]http://commons.apache.org/email/ Commons Mail, Abril de 2009-04-20

[46]http://www.wampserver.com/en/ Servidor WampServer, Abril de 2009-04-20

[47]http://www.eclipse.org/ Eclipse, Abril de 2009-04-20

[48]http://www.ffmpeg.org/ FFMPEG, Abril de 2009-04-20

[49]http://www.w3.org/XML/ Extensible Markup language, Abril de 2009-04-20

[50]W3C, *XSL Transformations (XSLT)*, W3C Recommendation 16 November 1999

[51]W3C, *Synchronized Multimedia Integration Language (SMIL 3.0)*, W3C Recommendation 1 December 2008

[52]http://mpeg-21.itec.uni-klu.ac.at/cocoon/mpeg21/_mpeg21Demo.html Klagenfurt University, Junho 2009-6-25