A new Impress extension for interactive MPEG-4 video conversion and a streaming architecture for E-learning

Bruno Carpentieri, Domenico Nunziata, Roberto Iannone.

***Abstract*—** MPEG-4 is not only a data compression standard algorithm, it gives also the opportunity to describe the scenes of a motion picture directly through XMT-A and XMT-O: two "high level languages" that are XML based. MPEG-4 therefore allows (and defines) the opportunities for video streaming. This makes this standard appealing for the conversion of Impress or PowerPoint presentations and for other e-learning applications because it offers the possibility of publishing the converted presentations on streaming servers. In this paper we exploit those possibilities by implementing an Impress extension that translates an electronic presentation to XMT-O, so to be finally "compiled" for MPEG-4 conversion, and we also describe a new streaming architecture for e-learning.

***Keywords*—**MPEG-4, Data Compression, XMT-O, Streaming, E-learning.

I. Introduction

THE MPEG-4 standard is not only a data compression algorithm. It gives also other opportunities to describe the scenes of a motion picture. In the MPEG-4 video standard (in particular the part 11 of this standard: the ISO / IEC 14496-11) we can describe the scenes of a motion picture directly through XMT-A and XMT-O: two high level languages XML based. MPEG-4 moreover allows and defines the opportunities for video streaming.

This makes this standard a good starting point for the conversion of multimedia presentations, such as Impress or PowerPoint presentations, and for other e-learning applications because it offers the possibility of publishing the converted presentations on streaming servers. This paper presents an Impress extension that translates an electronic presentation to XMT-O, so to be finally "compiled" for MPEG-4 conversion, and a streaming architecture for e-learning.

In a very simplified e-learning scenario, essentially two actors are present: a teacher, or content expertise, that produces the learning contents, and a student that takes advantage of those contents.

Our intention is to study the similarities between XMT-O and the XML based OpenDocument specifications. To do so we have implemented a converter that translates Impress multimedia presentations to MPEG-4. Preliminary results on this subject were presented in Iannone, Carpentieri and Annunziata [8] and Carpentieri and Iannone [9].

B. Carpentieri and R. Iannone are with the Dipartimento di Informatica, Universsità di Salerno, 84084 Fisciano (SA), tel. +39 089 969500, E-mail bc@dia.unisa.it , roberto.iannone@gmail.com. D. Nunziata is with STM Italia C. & O, 00128 Roma, E-mail : domenico.nunziata@gmail.it.

Our idea was inspired by Chih-Chun Lai and others [1] where the authors depict how MPEG-4 technologies can be helpful in authoring interactive e-learning contents. In particular the authors show how to exploit XMT-O and XMT-A to build-up three different applications of learning contents. The first is a navigation program for an historic monument. The second is a documentary that introduces the underwater creatures, such as fishes, turtles or cuttlefishes (a simple application where each fish is viewed as a moving object with a hyperlink that points to a predefined URL explaining about it). The last is an interactive English course, in which java script functionalities are utilized to evaluate the test results of a user.

XMT-A is a XML based language used by MPEG-4 to describe a scene of the BIFS where audio-visual object lies. XMT-O is a "friendly" version of XMT-A that was created to maintain compatibility with the W3C SMIL (see http://www.w3.org/AudioVideo/).

In this work we introduce an e-learning web based architecture that exploits the similarities between OpenDocument format, SMIL and XMT-O to build-up e-learning contents represented by MPEG-4 interactive videos that can be deployed through a Darwin streaming server.

II. MPEG-4 authoring and ODF

Many authors have studied the interactive authoring of MPEG-4 scenes by exploiting XMT-O conversions and implementations. Kyungae Cha and Sangwook Kim [2] implement a comprehensive set of facilitative editing tools for composing multimedia scene. They also present a tool for automatic generation of XMT documents and MPEG-4 contents in which the users can create their MPEG-4 scenes by using a graphical editor to put objects in the MPEG-4 video and also a timeline interface for animations and transitions. Chih-Chun Lai and others [1] implement three typology of learning contents by using XMT-O authoring, as previously described.

MPEG-4 is an ISO/IEC 14496 multimedia standard that enables the composition of multiple audio-visual objects. Users can display, play, and even interact with the scene contents that can be downloaded or streamed.

The scene contents can be protected by using an IPMP (Intellectual Property Management and Protection) system: this can be important in a learning environment where exchanged educational material could be covered by intellectual property rights.

The heart of the MPEG-4 scene composition is the BIFS (Binary Format for Scene) that arranges the scenes in a logical, hierarchical structure represented by a directed acyclic graph that has as its root the whole scene and as its leaves the objects of the scene (this graph is an extension of the VRML scene graph). Each node of the graph has time-space coordinates that are included in the time-space coordinates of its father node. It is possible to specify temporal relationships between objects along with the expansion of the tolerance time on each object.

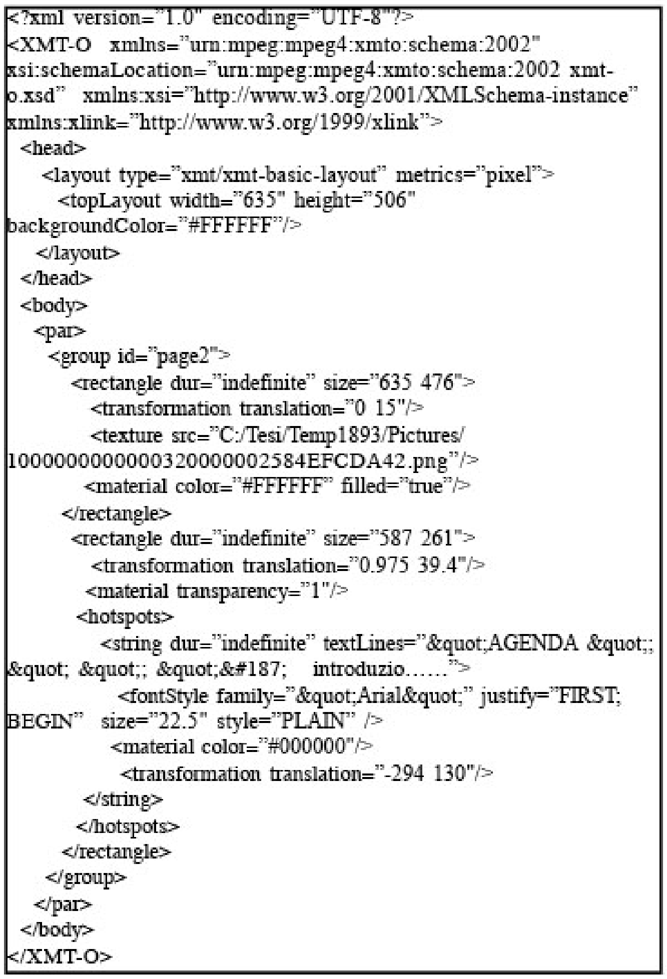
BIFS provides a very powerful event management system which allows the structure of the scene both to be changed dynamically and also to be driven by user interaction (e.g.: a user can move an object on the scene, he can click on an object to activate an event that permits a scene change - for example a “next” button on a visual presentation).

More interaction can be obtained by using MPEG-J: an MPEG-4 specification that defines a set of Java Application Programming Interfaces (APIs) to access and control the underlying MPEG-4 terminal that plays the MPEG-4 audio-visual session.

MPEG-4 gives the opportunity to describe the scenes of a motion picture directly through XMT-A (the “A” stands for Alpha) and XMT-O (the “O” stands for Omega), two “high level languages” XML based. XMT-A is an XML-based version of MPEG-4 BIFS content that provides a deterministic, one-to-one, mapping between the textual and the binary formats. XMT-O is a high-level abstraction of the MPEG-4 features based on the W3C SMIL language. It defines a subset of modules whose semantics are compatible. Therefore the XMT-O format can be parsed and played directly by a SMIL player or it can be compiled to an MPEG-4 representation that can be played by an MPEG-4 player.

XMT-O defines a document structure that is similar, but not identical, to the SMIL structure. The <XMT-O> tag can contain a single <head> and <body>, in that order. The XMT-O <head> tag can contain <meta>, <customAttributes>, <metadata>, <layout>, <transition>, <defs> and <macros>, while the <body> tag is the content itself. A key feature of the content is the high-level constructs for the audio-visual objects. These constructs not only describe the objects but also include their behavior. Like in SMIL the <body> element has the timing semantics of a <seq> timing container.

XMT-O is based on SMIL, but it is not the same language nor all the construct of SMIL have been integrated into XMT-O. In Figure 1 there is an example of a XMT-O file generated by the extension we developed. Inside the <head> tag we find the settings for type, layout and dimensions of the scenes that are derived from the ODP file. The <body> tag contains the <par> tag timing definitions and their behaviors. The <group> contains all the elements of the scene, so that these elements can be simultaneously animated.



```
<?xml version="1.0" encoding="UTF-8"?>
<XMT-O  xmlns="urn:mpeg:mpeg4:xmto:schema:2002"
xsi:schemaLocation="urn:mpeg:mpeg4:xmto:schema:2002 xmt-
o.xsd"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
     <layout type="xmt/xmt-basic-layout" metrics="pixel">
       <topLayout width="635" height="506"
backgroundColor="#FFFFFF"/>
    </layout>
  </head>
  <body>
    <par>
      <group id="page2">
         <rectangle dur="indefinite" size="635 476">
           <transformation translation="0 15"/>
           <texture src="C:/Tesi/Temp1893/Pictures/
10000000000003200000002584EFCDA42.png"/>
          <material color="#FFFFFF" filled="true"/>
        </rectangle>
         <rectangle dur="indefinite" size="587 261">
           <transformation translation="0.975 39.4"/>
          <material transparency="1"/>
         <hotspots>
            <string dur="indefinite" textLines="&quot;AGENDA &quot;;
&quot; &quot;; &quot;&#187;    introduzio……">
              <fontStyle family="&quot;Arial&quot;" justify="FIRST;
BEGIN"  size="22.5" style="PLAIN" />
             <material color="#000000"/>
              <transformation translation="-294 130"/>
          </string>
         </hotspots>
        </rectangle>
      </group>
    </par>
  </body>
</XMT-O>
```

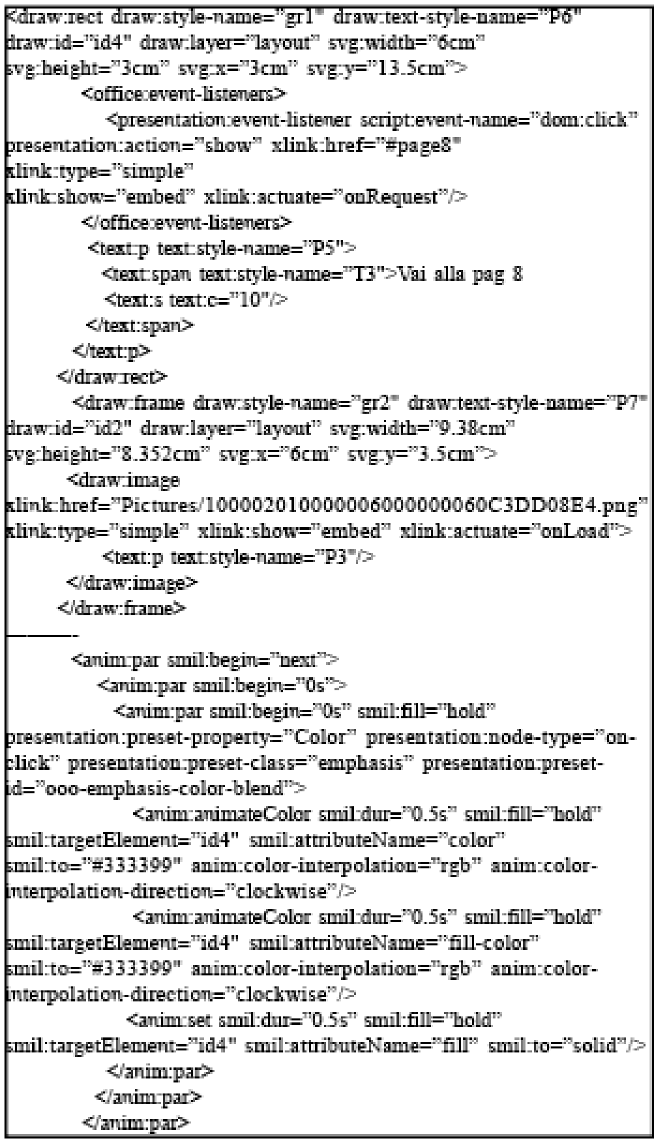
Fig. 1 An XMT-O file generated by our new extension

OpenDocument format (ODF, see [4]) is a file format for electronic office documents such as spreadsheets, charts, presentations and word processing. While the specifications were originally developed by Sun, the current standard was developed by the Open Office XML technical committee of the Organization for the Advancement of Structured Information Standards (OASIS consortium) and it is based on the XML format originally created and implemented by the OpenOffice.org office suite.

The structure of a ODF document can be represented via a ZIP compressed archive containing a files and directories; these can contain binary content and therefore it can use lossless compression to reduce file size. Furthermore OpenDocument benefits from separation of concerns by separating the content, styles, metadata and application settings into four separate XML files, so providing more flexibility:

• **content.xml**: it contains the actual content of the document except binary data (images for example).

• **styles.xml**: this file contains information about the styles used in the content. Paragraphs, page, characters, frame and lists styles will be found in this file.



```
<draw:rect draw:style-name="gr1" draw:text-style-name="P6"
draw:id="id4" draw:layer="layout" svg:width="6cm"
svg:height="3cm" svg:x="3cm" svg:y="13.5cm">
          <office:event-listeners>
             <presentation:event-listener script:event-name="dom:click"
presentation:action="show" xlink:href="#page8"
xlink:type="simple"
xlink:show="embed" xlink:actuate="onRequest"/>
          </office:event-listeners>
           <text:p text:style-name="P5">
             <text:span text:style-name="T3">Vai alla pag 8
             <text:s text:c="10"/>
           </text:span>
         </text:p>
       </draw:rect>
         <draw:frame draw:style-name="gr2" draw:text-style-name="P7"
draw:id="id2" draw:layer="layout" svg:width="9.38cm"
svg:height="8.352cm" svg:x="6cm" svg:y="3.5cm">
        <draw:image
xlink:href="Pictures/10000201000000600000060C3DD08E4.png"
xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad">
             <text:p text:style-name="P3"/>
        </draw:image>
       </draw:frame>
--------
         <anim:par smil:begin="next">
            <anim:par smil:begin="0s">
              <anim:par smil:begin="0s" smil:fill="hold"
presentation:preset-property="Color" presentation:node-type="on-
click" presentation:preset-class="emphasis" presentation:preset-
id="ooo-emphasis-color-blend">
                 <anim:animateColor smil:dur="0.5s" smil:fill="hold"
smil:targetElement="id4" smil:attributeName="color"
smil:to="#333399" anim:color-interpolation="rgb" anim:color-
interpolation-direction="clockwise"/>
                 <anim:animateColor smil:dur="0.5s" smil:fill="hold"
smil:targetElement="id4" smil:attributeName="fill-color"
smil:to="#333399" anim:color-interpolation="rgb" anim:color-
interpolation-direction="clockwise"/>
                <anim:set smil:dur="0.5s" smil:fill="hold"
smil:targetElement="id4" smil:attributeName="fill" smil:to="solid"/>
              </anim:par>
            </anim:par>
          </anim:par>
```

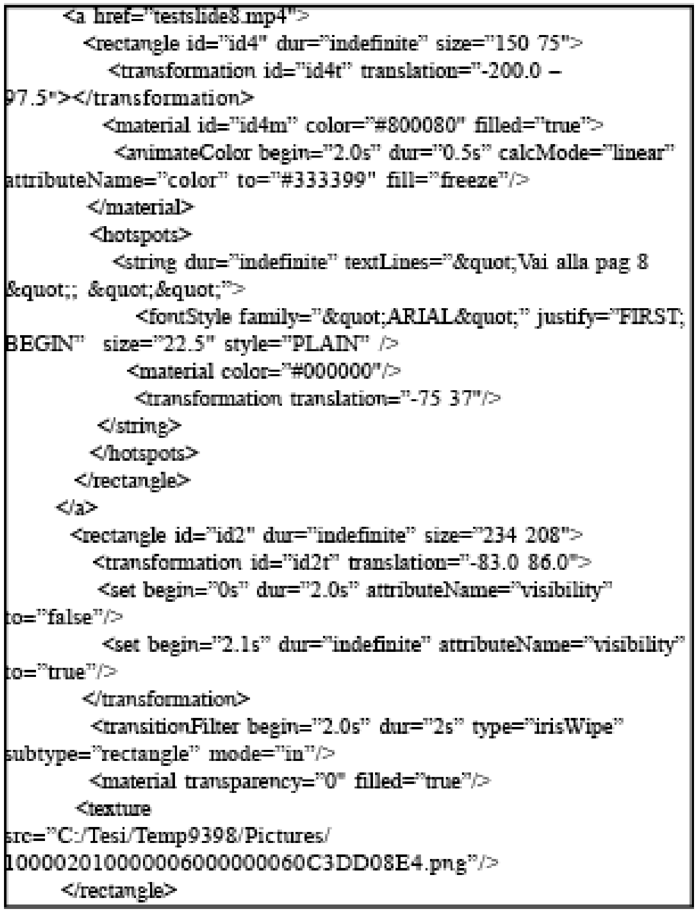
Fig. 2 ODF code that represents the description of two rectangles of different sizes, placed in two different parts of the scene.

• **meta.xml**: this file contains information about the document itself. These information could be divided into four categories: general document properties (creation and / or modification date, size, etc…), document description (title, description, author, etc..), user-defined information and document statistics (total number of page and words, number of pictures in the document, etc.). Data are described by the OpenDocument meta namespace and by the Dublin Core definitions.

• **settings.xml**: this file contains application specific information as zoom factor, cursor position on the document, etc.

• **mimetype.xml**: this file contains MIME type of the document. Essentially this file determines the OpenDocument file type: application/vnd.oasis.opendocument.presentation is the MIME type for the ODP document files.

The four folders are:



```
        <a href="testslide8.mp4">
          <rectangle id="id4" dur="indefinite" size="150 75">
             <transformation id="id4t" translation="-200.0 –
97.5"></transformation>
            <material id="id4m" color="#800080" filled="true">
              <animateColor begin="2.0s" dur="0.5s" calcMode="linear"
attributeName="color" to="#333399" fill="freeze"/>
           </material>
           <hotspots>
              <string dur="indefinite" textLines="&quot;Vai alla pag 8
&quot;; &quot;&quot;">
                 <fontStyle family="&quot;ARIAL&quot;" justify="FIRST;
BEGIN"  size="22.5" style="PLAIN" />
                <material color="#000000"/>
                 <transformation translation="-75 37"/>
            </string>
           </hotspots>
          </rectangle>
        </a>
         <rectangle id="id2" dur="indefinite" size="234 208">
            <transformation id="id2t" translation="-83.0 86.0">
             <set begin="0s" dur="2.0s" attributeName="visibility"
to="false"/>
              <set begin="2.1s" dur="indefinite" attributeName="visibility"
to="true"/>
           </transformation>
            <transitionFilter begin="2.0s" dur="2s" type="irisWipe"
subtype="rectangle" mode="in"/>
            <material transparency="0" filled="true"/>
          <texture
src="C:/Tesi/Temp9398/Pictures/
10000201000000600000060C3DD08E4.png"/>
        </rectangle>
```

Fig. 3 .XMT-O code that represents a similar description.

• **META-INF**: this folder contains the manifest.xml file that describes the contents of the compressed file.

• **Configurations2**: this folder contains application specific and advanced configurations.

• **Pictures**: this folder contains all the document pictures

• **Thumbnails**: this folder contains the document thumbnails.

OpenOffice Impress is a presentation software that is part of a suite of programs from OpenOffice.org. OpenOffice Impress is available as a free download. Unlike other proprietary presentation software, Impress, and all the suite of OpenOffice, are licensed as open-source so they can be studied, modified, and enhanced freely. In educational environments this is the software that is frequently used to deliver knowledge to the audience.

The ODF code in Figure 2 represents the description of two rectangles of different sizes, placed in two different parts of the scene. The XMT-O code in Figure 3 represents a similar description.

The syntax and the language constructions (even the units of measurement) used in these two codes to describe the same objects are completely different.

In order to convert all the objects and in order to place them correctly in the scene while keeping the original size, a number of operations are carried out to transform the dimensions and locations of the objects from centimeters to pixels.

ODF has as absolute reference the upper left corner, while XMT-O has as absolute reference the center of the scene. If we want to translate from ODF to XMT-O we therefore need also to consider the correct item positioning.

In ODF the animations are grouped after the definition of the objects, before the closing tag of the page, and they are managed through the references to the IDs of the objects. In XMT-O the tags for animations and transitions are included in the tags that define the item to which they relate.

Another difference concerns hyperlinks. In ODF these are seen as the actions to be taken in the presence of an appropriate listener. In XMT-O the links are handled as in HTML: by using a tag in which there are other tags that define the item as "interactive".

Those differences make the conversion process very hard but possible. In the rest of this paper we will describe how to implement this conversion.

III. An Impress extension for MPEG-4 conversion

An Impress extension has been developed with the purpose of giving to the content creator an easy way to deal with the creation of the content. In fact the extension gives the opportunity to continue working with software that is well known and easy to use (Impress) and, after the work is finished, it allows the translation of the presentation into an MPEG-4 video that preserves most of the characteristics of the multimedia presentation.

This extension has been developed in Java by using the IDE NetBeans with the OpenOffice plug-in; so the extension can be perfectly integrated into the OpenOffice.org suite.

When the extension is launched, a window appears and the dimensions of the video can be selected (the dimensions are approximate because the slide dimensions in Impress are variable: they are expressed in centimeter instead of pixel); after this choice the conversion process starts.

The presentation is first converted into XMT-O and after into an MPEG-4 video, this is carried out in eight steps:

1. The path of the presentation file is retrieved by using a query to the *UNO* Component; into this path it is created a temporary folder with name *TempX* (where X is a progressively incremented integer); the files contained into the .odp archive are then extracted into this temporary folder.

2. The *styles.xml* file is parsed and the general information about the style of the presentation are retrieved; in particular the name of the default style, the height and the width of the slide (the dimension are scaled according to the value selected for the video format), and the information for the visualization of the default background (color or image) are extracted.

3. The file *content.xml* is parsed and an output file with name *[presentation_name]slideX.xml* is opened for each slide of the presentation (X will be incremented progressively). This file contains the instructions, in XMT-O, that define the aspect and the behavior of the slide.

4. The method *gettransition* is invoked. Its task is to recover the data on the transition to the current slide and to restore the necessary instructions to play the exact transition (or something similar).

5. The method *convertislide* is invoked. This method is responsible for processing, one by one, all the elements of the slide and for their translation into XMT-O. This method performs the conversion of all the graphics by first collecting all the information about size, location, attributes, etc. (scaling and translation operations might be needed for the translation of size and positioning information, depending on the size of the video). Then, in the final step, by using the information collected, the method generates XMT-O tags that are needed to create and place the slide items.

6. When all the graphics have been finally prepared and positioned, then an area is defined, as footnote to the scene, containing two interactive arrows that indicate two possible movement directions: to advance to the next slide or to return to previous one.

7. The XMT-O document is closed with its closing tag. It is called the constructor to create an object of type *XMTBatch*, which is invoked via the run method for the conversion of the XMT-O files generated into an MPEG-4 video. The name of the MPEG-4 file that is generated here is *[presentation_name]slideX.mp4*.

8. After the conversion of all the slides of the presentation, the temporary folder is deleted and the execution ends.

IV. Conversion limits and compromises

The conversion from ODF to MPEG-4 needs to be adjusted by accepting a few appearance compromises. These adjustments cover all the graphics aspects: there shall be areas where compromises are more pronounced and other in which the compromises are almost imperceptible. The following is a list of the accepted compromises (in our implementation) among the original and the output:

Background: When the background of the slide is formed by the repetition of a single image, since XMT-O does not foresee the possibility to repeat an image as the texture of an item, we scale the image to force an automatic repetition of the image to cover the area occupied by the rectangle that is the background for the slide.

Dimensions: the compromise in this case concerns the conversion from the original size, expressed in centimeters, to the output size in pixels. To get videos that have pixel size close to the standard screen resolution, we had to use multiplying factors to scale both the graphic objects (20 for small, 25 for medium and 35 for big ones), or to adjust the font size for the text (1 for small, 1.25 for medium and 1.75 for big one).

Ellipse: these graphics elements are not supported by XMT-O. The compromise in this case is inherent to the generation of these elements. To obtain this geometric figure, we started from a circle having as radius the size of the main axes, and then we proceeded scaling on one of the two axes, in a way that was proportionally to the size of the secondary axis.

Transitions: In this field the compromises have been necessary because many SMPTE transitions have not been implemented in the IBMToolkitforMpeg-4 [5], but all of them were implemented in Impress. This led us to use different transitions in place of those not yet supported. Whenever it was possible we have replaced the missing transitions with similar ones.

Animations: It was not possible to translate with accuracy some emphasis animations; particularly the animations that included the change of parameters such as contrast, brightness, etc., because these are not contemplated in XMT-O. For motion animations along paths, it was decided to replace all the paths with a rectangle, given the impossibility of a direct conversion of any path and the huge amount of time required by a manual conversion of the different possible paths. For entry and exit animations for which it was not possible a conversion it has been chosen to make the item to disappear or to appear at the specified time.

Text: This is the point were more compromises are needed. This is, on one side, because of the poor support given by XMT-O and, on the other side, because of the large capacity of text formatting provided by Impress. The first compromise here concerns the loss of various text styles that may be present in the same Impress paragraph: with Impress, within a paragraph, you can define different sections, called span, and for each span you can define a different text style. This is not feasible in XMT-O (unless we put by hand a tag <string …> for each different word, and this is not feasible). For simplicity we used the text style of the last block in the paragraph for all the text of the paragraph. The other needed compromise relates to text formatting, as the text sections have no information about the new line, given that the text in Impress is formatted at run time, so we had to use a trick based on the measurement of the length in pixels of each word by using an instance of the class *FontMetrics* that provide the method *stringWidth* that is able to return the length in pixels of the string passed as argument.

There are also a number of graphic elements that are not yet supported in our prototype (for each of them we indicate a few hints for a future implementation):

Curve line: this type of line, is described, into the OpenDocument standard, by the utilization of the syntax derived from the tag <path d="…"…> of SVG, in XMT-O there is a tag that is similar but unfortunately not identical. So to correctly convert this type of graphics element, it must be studied the correct parameter to obtain the exact conversion.

Not geometrical shapes: this class of elements contains arrows, callout, flow diagram, stars etc. Those types of shapes are defined in a way that is similar to the curve line.

3D shapes: these types of graphic elements are supported natively by XMT-O, with all the effect of light and shadow. These shapes have not yet been included into the software because they are not very frequent into the e-learning content.

Video: XMT-O allows the inclusion of many type of videos. Unfortunately this feature is not yet implemented into the IBMToolkitforMpeg-4 that supports only the videos encoded into MPEG-4 format. The toolkit creators have released a video converter (*AVGen*) that allows the conversion of many formats of digital video into the MPEG-4 format.

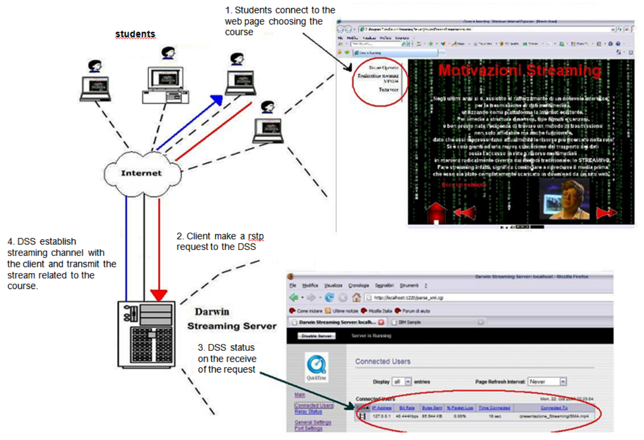

students
1. Students connect to the web page choosing the course
Motivazioni Streaming
Internet
4. DSS establish streaming channel with the client and transmit the stream related to the course.
2. Client make a rstp request to the DSS
Darwin Streaming Server
3. DSS status on the receive of the request
Connected Users

Fig. 4 . Our e-learning web based architecture (streaming contents are previously generated by our Impress mp4 conversion plugin).

This converter can be included into the software developed to allow the inclusion of video after the conversion process.

Audio: the situation is very similar to the Video files, but there is a format (MP3: MPEG-1 Layer 3) that is common to both standard. So this type can be used directly, a conversion must be performed for the other types of audio files.

V. A STREAMING AUDIO-VIDEO MPEG-4 APPLICATION

As depicted in Figure 4, once the contents have been converted to MPEG-4 files, they need to be published so to be delivered to the student audience (see Figure 5).

An efficient and fast way to deliver these contents is streaming. A streaming media is a multimedia that is constantly received by, and normally presented to, an end-user by a streaming provider. The delivery of streaming content is ruled by three protocols: the first is the Real-time Streaming Protocol (or RTSP) that defines how the client could control remotely the streaming media server, issuing VCR-like commands such as play and pause, and allowing time-based access to files on a server; the second is the Real-time Transport Protocol (or RTP) that defines a standardized packet format for delivering of audio and video over the Internet.

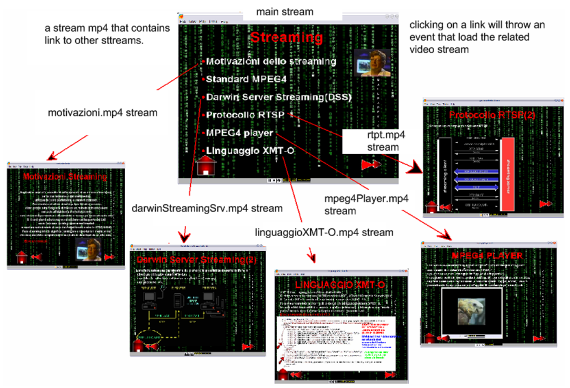

main stream
a stream mp4 that contains link to other sttreams.
clicking on a link will throw an event that load the related video stream
Streaming
Motivazioni dello streaming
Standard MPEG4
Darwin Server Streaming(DSS)
Protocollo RTSP
MPEG4 player
Linguaggio XMT-O
motivazioni.mp4 stream
rtpt.mp4 stream
darwinStreamingSrv.mp4 stream
mpeg4Player.mp4 stream
linguaggioXMT-O.mp4 stream
Motivazioni Streaming
Protocollo RTSP(2)
Darwin Server Streaming
LINGUAGGIO XMT-O
MPEG4 PLAYER

Fig. 5 . An example of learning course about streaming. The course is composed by different video stream that are loaded on user trigger events (the click on a link).

The last is the Real Data Transport (or RDT) that is a proprietary transport protocol for the actual audio/video data (developed by RealNetworks in the 1990s).

In our architecture we have chosen as streaming provider the freely available Apple Darwin Streaming Server[6] (DSS). The converted presentation is published on a DSS so students can take advantage of the learning content through a regular web browser with an MPEG-4 player. We have experimented with the IBM M4Applet[7] a Java applet developed by the Composite Media Technologies Group at the T. J. Watson IBM research center.

VI. Testing, results, and conclusions

Comparing the original slides with their translations reveals where the conversion compromises act.

The two screenshots in Figure 6 show that for the low impact compromises, the results obtained are very similar to the original (Figure 6, Figure 7).

For the text, the compromises have a relevant visual impact (Figure 8, Figure 9).

Anyway the overall visual effect of the converted presentation is good. From the point of view of the content the slides express the same semantic concepts and maintain the same spirit that the authors have impress on them.

In this paper we have presented an ideal e-learning architecture were educational contents are built by using an OpenOffice Impress presentation to be converted through our Impress plug-in to different MPEG-4 videos that can be published on a DSS to be streamed to a student audience.

We therefore showed how, by exploiting the similarities between SMIL, XMT-O and OpenDocument format it was possible to build-up an OpenDocument to XMT-O converter.

Future work will be devoted to enhance the converter and to reduce the compromises by introducing better conversion rules. Further studies are needed to incorporate MPEG-J functionalities for a richer interactive experience and to blend together the streaming architecture and the impress plug-in, obtaining a system, web services based, to send directly from Impress the converted publication to the DSS.

We are also considering the usage of the streaming architecture for applications related to e-learning and data compression as for instance interactive data compression (see Carpentieri [10] and Carpentieri [11]) and image or layered documents compression (see Carpentieri [12] and Ansalone and Carpentieri [13]).

References

[1] Chih-Chun Lai, Chihwei Pan, Chia-Hung Tsai, Yu-Chen Tsai; ‘Authoring and Presentation of Interactive eLearning Content by Using MPEG-4 BIFS Technologies’ , 16th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP 2003), (2003).

[2] Young, Kyungae Cha, Sangwook Kim; ‘Interactive Authoring Tool for Extensible MPEG-4 Textual Format (XMT)’ , Workshop Notes of Semantic Authoring, Annotation and Knowledge Markup, 15th European Conference on Artificial Intelligence (ECAI 2002), Lyon, France, July 22-26, 71-75 (2002).

[3] Mikaël Bourges-Sévenier, Euee S.Jang; IEEE Transaction on Circuits and Systems for Video Technology, 14(7), July (2004). W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[4] Oasis Consortium; ‘Open Document Format for Office Applications Specification v1.1’ , http://www.oasisopen.org/specs/index.php#opendocumentv1.1.

[5] AlphaWorks; ‘IBM Toolkit for MPEG-4’ , http://www.alphaworks.ibm.com/tech/tk4mpeg4; May (2003).

[6] Apple Computer Inc.; ‘QuickTime Streaming Server and Darwin Streaming Server Administrator’ s Guide’ , May (2002).

[7] IBM Research; ‘IBM MPEG-4 Multimedia Applet Player’ , http://www.research.ibm.com/mpeg4/Projects/player.htm; January (2006).

[8] Roberto Iannone, Bruno Carpentieri, Domenico Nunziata; ‘Interactive MPEG-4 Videos: An Impress Extension for MPEG-4 Conversion and a Streaming Architecture for e-Learning’ . IEEE ISCIS 2009, 29-34 (2009).

[9] Bruno Carpentieri, Roberto Iannone, “Building an Impress Extension for Interactive MPEG-4 Video Conversion”, in *Latest trends in information technology*, Proceedings of WSEAS ICTN 2012, (2012).

[10] Bruno Carpentieri: “Interactive Compression of Digital Data”. Algorithms 3(1): 63-75 (2010).

[11] Bruno Carpentieri: “Interactive Compression of Books”. WSEAS Transactions on Computers 9 (3) , pp. 278-287 (2010).

[12] Bruno Carpentieri: “Image compression via textual substitution”. WSEAS Transactions on Information Science and Applications 6 (5) , pp. 768-777 (2009).

[13] Anna Ansalone and Bruno Carpentieri: “How to set "don't care" pixels when lossless compressing layered documents”. WSEAS Transactions on Information Science and Applications 4 (1) , pp. 220-225 (2007).

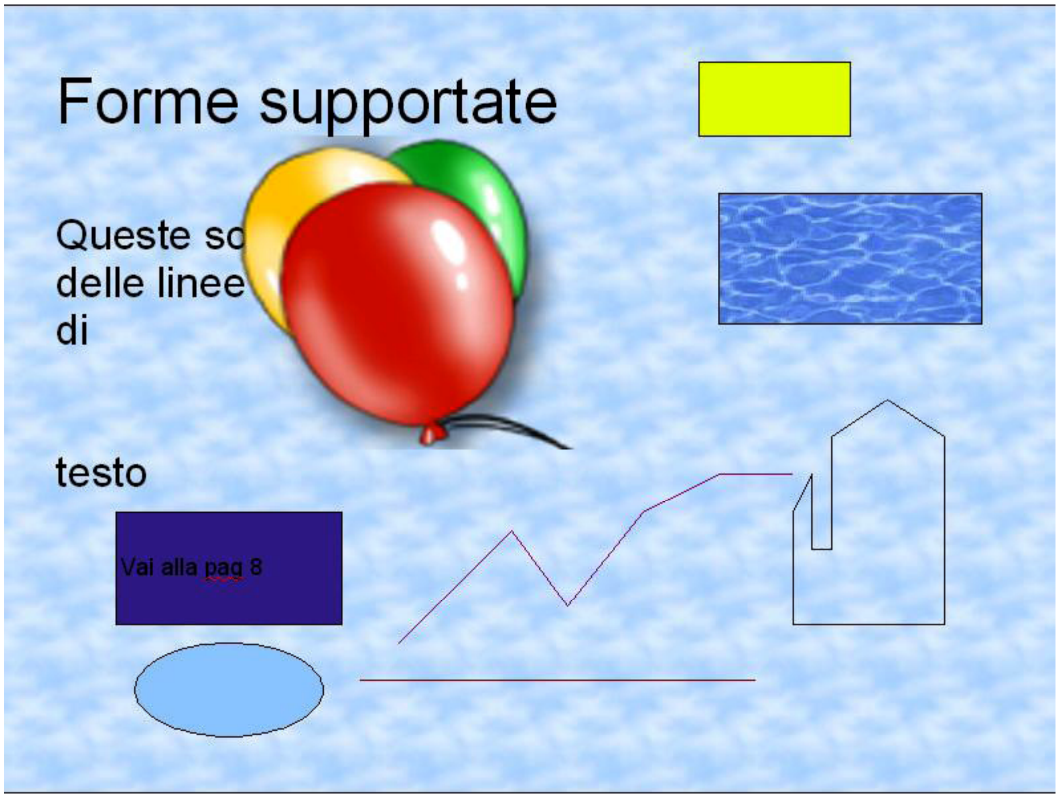


Fig. 6 . Original ODP slide

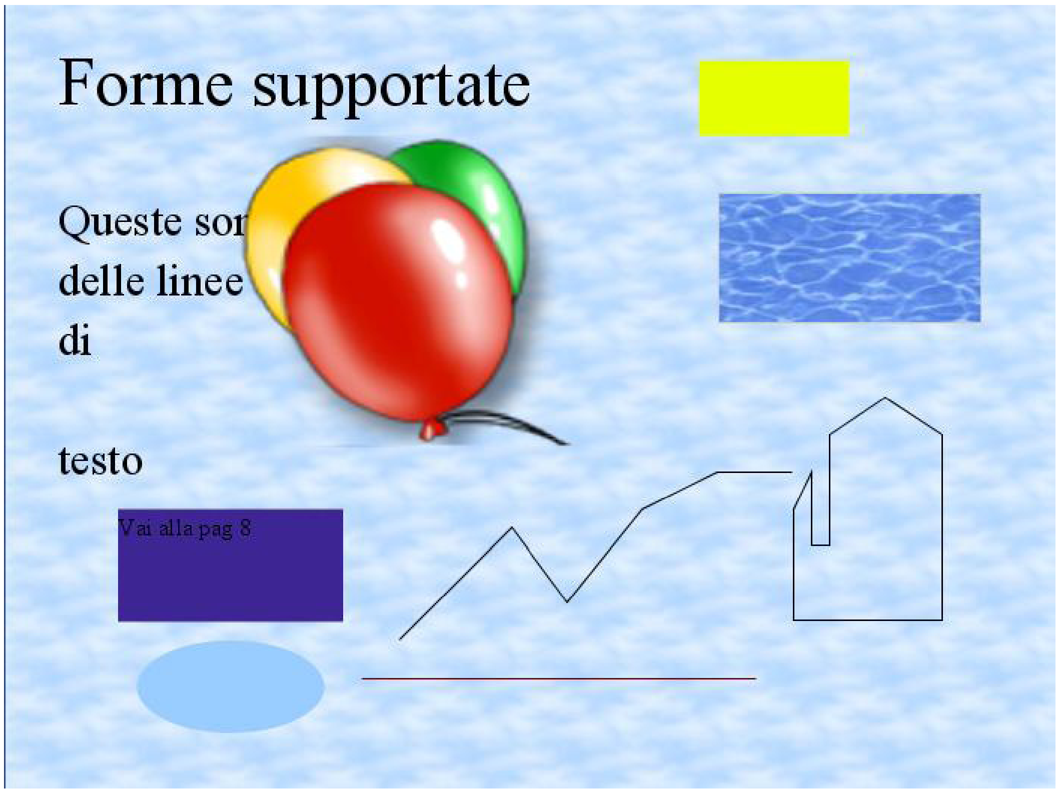
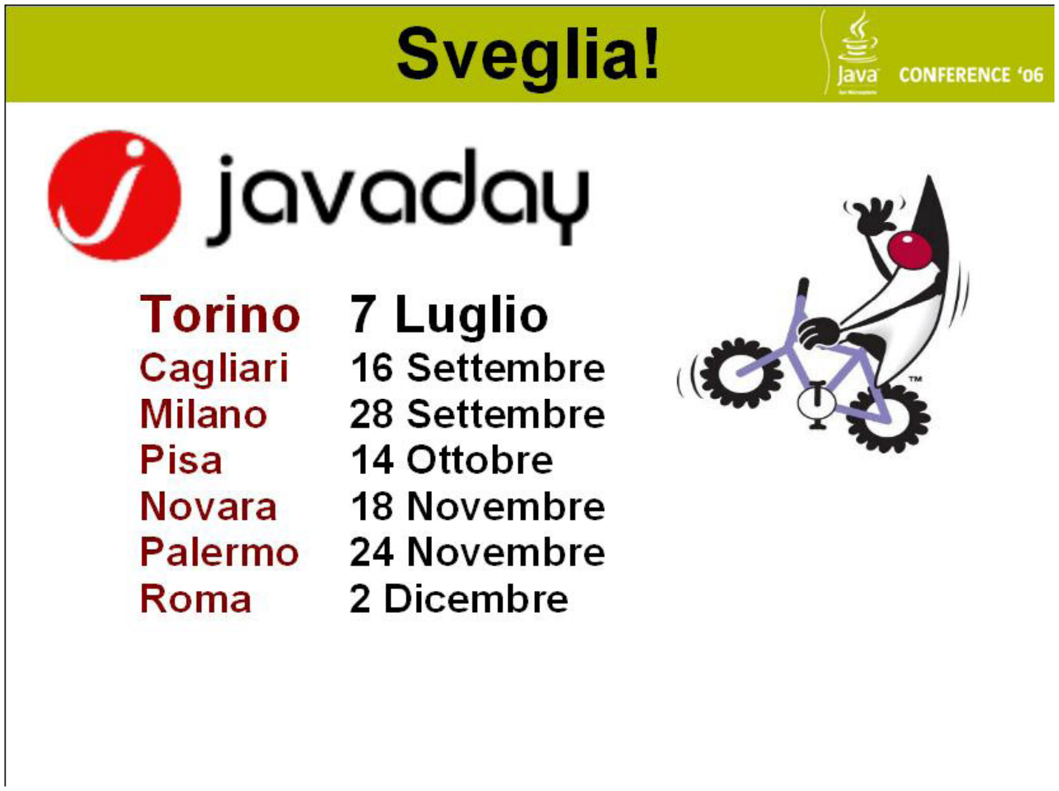


Fig. 7 . MPEG-4 converted version (some compromises about text layout and formatting are visible)

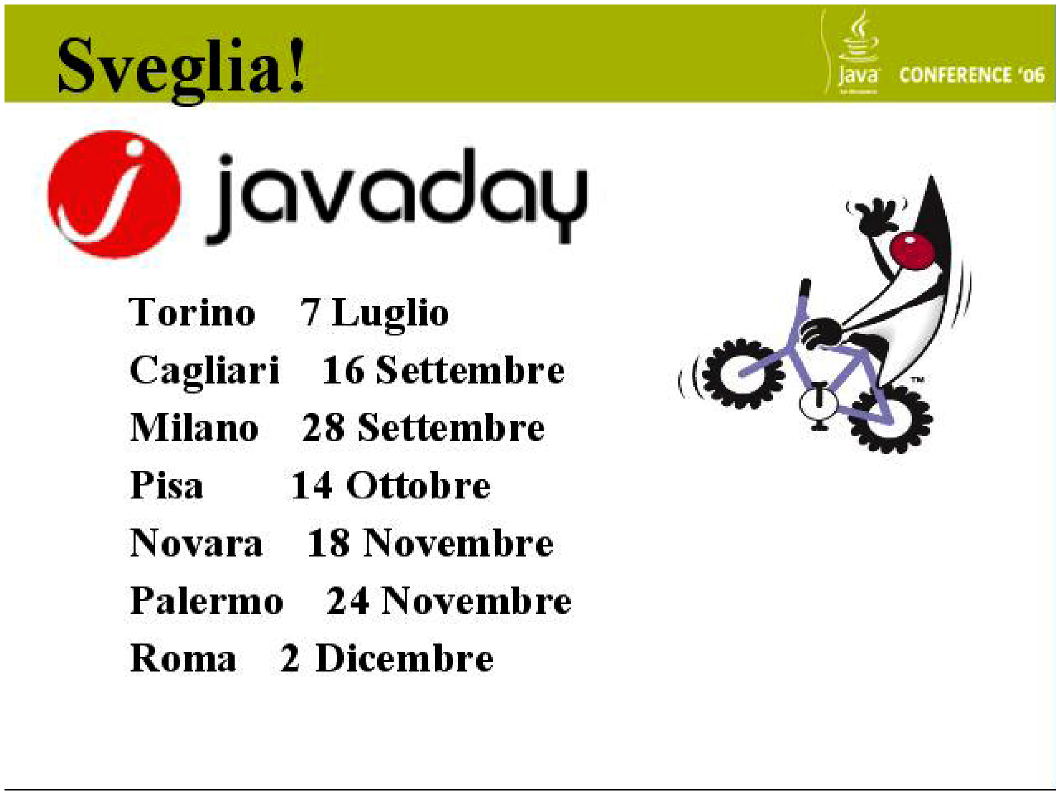


Sveglia!

javaday

Torino	7 Luglio
Cagliari	16 Settembre
Milano	28 Settembre
Pisa	14 Ottobre
Novara	18 Novembre
Palermo	24 Novembre
Roma	2 Dicembre

Fig. 8 . Original ODP slide



Sveglia!

javaday

Torino	7 Luglio
Cagliari	16 Settembre
Milano	28 Settembre
Pisa	14 Ottobre
Novara	18 Novembre
Palermo	24 Novembre
Roma	2 Dicembre

Fig. 9 . MPEG-4 converted version (some compromises about text layout and formatting are visible)