MXDemo:
a Case Study about Audio, Video, and Score Synchronization

Adriano Baratè, Goffredo Haus, Luca A. Ludovico, Giancarlo Vercellesi
LIM (Laboratorio di Informatica Musicale) – DICo – Università degli Studi di Milano
{barate, haus, ludovico, vercellesi}@dico.unimi.it

Abstract

Information Technology provides many ways to describe music contents. An integrated and comprehensive representation is possible, as MX format demonstrates. This paper illustrates the basic principles of synchronization in a multi-layered XML-based environment aimed at music description, and provides an example of software implementation.

1. Introduction

Music language is made up of many different and complementary aspects. Music is the composition itself as well as the sound a listener hears, and is the score that a performer reads as well as the execution provided by a computer system.

The encoding formats commonly accepted and employed are often characterized by a partial perspective of the whole matter: they describe data or metadata for score, audio tracks, computer performances of music pieces, but they seldom encode all these aspects together. Nowadays we have at our disposal many encoding formats aimed at a precise characterization of only one (or few) music aspect(s). For example, MP3, AAC and PCM formats provide ways to encode audio recordings; MIDI represents – among other things – a well known standard for computer–driven performance; TIFF and JPEG files can result from a scanning process of scores; NIFF, Enigma, Finale formats are aimed at score typing and publishing.

The first problem to face is finding a comprehensive way to encode all these different aspects in a common framework, without repudiating the accepted formats (see [6] and [8]). An important advantage of such effort is keeping together all the information related to a single music piece, in order to appreciate the richness of heterogeneous representations of music (aural, visual, logical, structural descriptions). But this key advantage has an interesting consequence: the possibility to create a strongly interconnected and synchronized environment to enjoy music. The purpose of our MXDemo is illustrating the full potentialities of an integrated approach to music description.

This goal can be achieved thanks to two cooperating elements, both described in the paper: (i) a comprehensive XML-based format to encode music in all its aspects, and (ii) a software environment aimed to the integrated representation. The software application will provide a graphic interface to read, watch, and listen to music, keeping the different levels synchronized.

2. A short description of MX format

In order to provide a comprehensive music description, we are currently developing an XML-based format called MX. The reasons why XML is a suitable format to represent music are well summarized in [8]. The MX format is undergoing the IEEE standardization process, as described in [4]. Our approach is different from the aforementioned, partial perspectives in music description, in particular because we represent music information according to a multi-layer structure and to the concept of space-time construct.

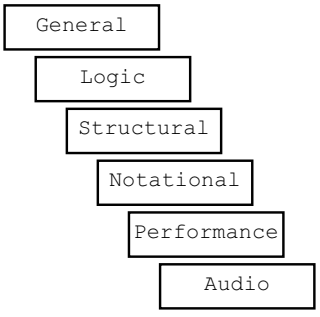

Figure 1. MX layers

Proceedings of the First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05)
0-7695-2348-X/05 $20.00 © 2005 IEEE

According to us, musical information should be structured by using a layer subdivision model, as shown in Figure 1. In our proposal for a common and exhaustive format, we distinguish among `General`, `Logic`, `Structural`, `Notational`, `Performance` and `Audio` layers (see Figure 1). Each layer is specific to a different degree of abstraction in music information.

The various levels are not independent; on the other hand, they reference (and are referenced by) a common structure that relates them from a spatial and temporal perspective. In fact, when considering music as a multi-layered information, we need a sort of glue to keep together the heterogeneous contributions music is made of. Accordingly, we introduced the concept of *spine*. Spine is a structure that relates time and space information (see Figure 2), using measurement units expressed in relative format.

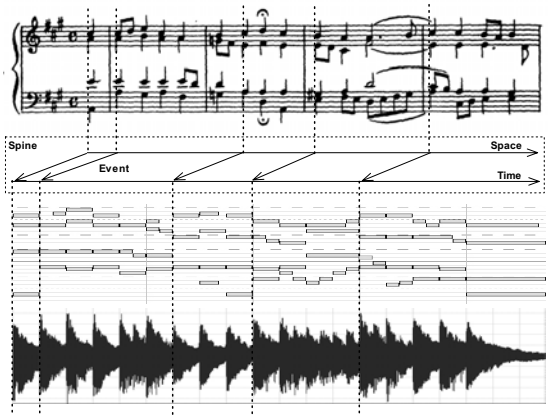

Figure 2. The role of MX spine

Figure 2 provides a graphical representation of the relationships between `Notational`, `Performance` and `Audio` layers, pointing out the role of the spine. From a logical and syntactical point of view, spine supplies a series of labels for music events; those labels will be used in other MX layers to reference the related object. If we concentrate on temporal characteristics of spine, it could resemble a timeline; likewise, from a spatial perspective, spine is a way to sort music events in the horizontal and vertical dimensions.

3. An overview about MX layers

After describing the main concepts about MX, we provide an overview about MX layers. Our discussion will not represent an in-depth study about the matter, it will just provide sufficient details to understand and appreciate how the MXDemo works. However, the complete DTD of MX 1.5 format is currently available at IEEE official Web site:

http://www.computer.org/standards/1599/par.htm

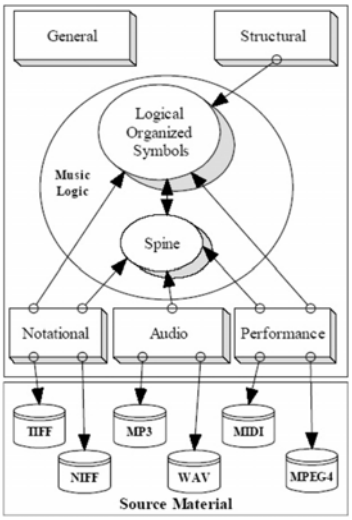

Figure 3. Connections among layers

The `General` layer is the only one not directly related to score's musical contents. This layer simply describes some fundamental alphanumeric information, namely metadata, about the coded music work. For instance, the title of the composition and its author are saved inside this MX layer.

In the following, the layers we will list are interconnected by the spine mechanism, as shown in Figure 3.

The `Logic` layer contains the information referenced by all other layers, and represents what the composer intended to put in the piece. It is composed of two elements: (i) the `Spine` description, used to mark the significant events in order to reference them from the other layers, and (ii) the `LOS` (Logically Organized Symbols) element, that describes the score from a symbolic point of view (e.g., chords, rests, horizontal symbols).

The `Structural` layer contains explicit descriptions of music objects together with their causal relationships, from both the compositional and musicological point of view, i.e. how music objects can be described as a transformation of previously described music objects. A common example of analysis hosted by this layer is the so-called "chord grid", but other types of structural descriptions are supported as well (for instance, Music Petri Nets).

The Notational layer links all possible visual instances of a music piece. Representations can be roughly grouped in two types: notational and graphical. A notational instance is often in a binary format, such as NIFF or Enigma, whereas a graphical instance contains images representing the score. Usually, the latter is in a binary format too (e.g., a JPEG image or a PDF file), but it can also be a vector image encoded in SVG. The information contained in this layer is tied to the spatial part of the Spine structure, allowing symbol localization.

The Performance layer lies between Notational and Audio layers. File formats supported by this level encode parameters of notes to be played and parameters of sounds to be created by a computer performance. This layer supports symbolic formats such as MIDI, Csound, and MPEG-4 SA files.

Finally, the Audio layer describes source material containing musical audio information, such as AAC, MP3, and PCM codings. This layer represents the lowest level of our layered structure.

4. Synchronization mechanisms

As we said before, in order to get synchronization among Audio, LOS, Notational, and Performance layers, we have developed a mechanism based on spine concept. Spine allows the interconnection of these layers on space and time domain through a relative measure in spine and an absolute measure in Audio, Performance, and Notational layers.

In MX spine, each music event is univocally defined by an identifier (*id*) and appropriate *timing* and *hpos* information. Timing is expressed in a relative way: the measurement unit is user-defined in function of time domain (e.g., 1 quaver may correspond to 1024 timing units) and its value is the timing distance from the preceding event. *Hpos* has a similar meaning, but in space domain. Now we provide a simplified example of spine.

```
<spine>
    <event id="e0" timing="NULL" hpos="NULL"/>
    <event id="e1" timing="0" hpos="3"/>
    <event id="e2" timing="1024" hpos="14"/>
    …
</spine>
```

At Audio level, every media linked to MX is defined as a clip which is mapped to spine events through the *clip_indexing* tag. Each single music event in spine can be physically indexed. This is the purpose of the *clip_spine_event* tag, which contains the spine identifier (*spine_ref*) and an absolute reference (*timing*) that specifies the time position of the event into the media. Unlike spine timing, at Audio level there are absolute time references, using different measurement units in function of the various kinds of media.

As regards the *timing_type* attribute, the default unit is the second, but it is possible to use bytes, samples, or frames as well. This solution supplies a more sophisticated time granularity according to the different media structures (PCM, MP3, AAC, etc.) supported by MX.

This paper does not address the techniques that can be employed to get synchronization among layers. It will be sufficient to know that synchronization in time domain can be obtained following three different approaches.

- Hand-made approach: synchronization and spine are completely created by human beings. For this purpose, it is necessary a good experience as a musician and as a trained listener.
- Semi–automatic approach: start beats are manually set by human beings, whereas intermediate notes are looked for by opportune algorithms based on interpolation techniques.
- Automatic approach: every music event is recognized automatically by means of audio–score synchronization algorithms.

Now we provide an example of Audio layer containing two clips, where the default time units are respectively the second and the sample-aligned frame.

```
<audio>
  <clip>
    <clip_indexing>
      <clip_spine timing_type="sec">
        <clip_spine_event timing="0.00" spine_ref="e1"/>
        <clip_spine_event timing="0.50" spine_ref="e2"/>
        …
    </clip_indexing>
  </clip>
  <clip>
    <frame_align measurement_unit = “sample”>
        1152
    </frame_align>
    <clip_indexing>
      <clip_spine timing_type="frame">
        <clip_spine_event timing="1" spine_ref="e1"/>
        <clip_spine_event timing="3" spine_ref="e2"/>
        ...
    </clip_indexing>
  </clip>
</audio>
```

It is also possible to synchronize video clip containing music that can be related to Logic/LOS (e.g. the soundtrack of a movie or the videoclip of a song). Once again, the synchronization is based on absolute timing values.

0-7695-2348-X/05 $20.00 © 2005 IEEE

`Performance` layer provides the same functionalities and synchronization mechanisms, now applied to computer-driven performances. Thus, also MIDI, Csound and MPEG files can be linked to MX.

Synchronization for `Notational` layer is somehow similar. Once again, it is based on `Logic/Spine` layer, where each event is marked in order to be referenced. However, at `Notational` level, MX format has to manage space relationships and not time references. We define a number of regions in score image(s) through two couples of coordinates. The purpose is associating spine markers to the corresponding bounding boxes. Here is an example of `LOS` and `Notational` levels.

```
<los>
    <chord event_ref="e2">
        <notehead>
            <pitch step="G" octave="5"/>
            <duration num="1" den="8"/>
        </notehead>
    </chord>
</los>
<notational>
    <graphic>
        <graphic_event spine_ref = “e1”
            upper_left_x = ”100”
            upper_left_y = “200”
            lower_right_x = “1000”
            lower_right_y = “1500”/>
        <graphic_event spine_ref = “e2”
            upper_left_x = ”500”
            upper_left_y = “200”
            lower_right_x = “1500”
            lower_right_y = “1500”/>
        …
    </graphic>
</notational>
```

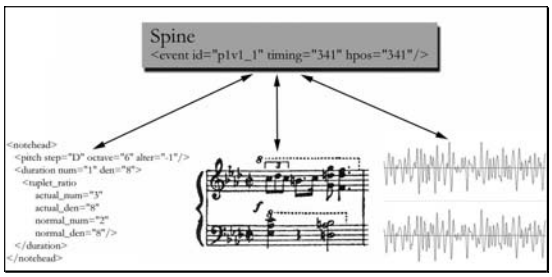

Figure 4. Possible mappings from and to spine

Therefore, each music event is logically mapped on spine, and it is accordingly referenced from the other layers. `Spine` and all the other layers (except `General` layer) are linked each other by event identifiers, acting like pointers. Identifiers are useful to jump from a kind of representation to another one, and to allow synchronized movement along layers both in time and in space domain.

Figure 4 illustrates some possible relationships within a MX file. Double arrows indicate two-way navigation. In other terms, spine mechanism allows not only to investigate all the encoded representations of the same spine event, but also to pass from a “peripheral” layer to another one through `Spine`.

5. The aim of MXDemo

The layered concept of the MX format is evidently visible in the main window of the MXDemo interface (see Figure 5): on the left there are seven radio buttons, one for each layer, whereas the main part of the window is devoted to the text box showing the entire MX file.

A toolbar is dedicated to service operations, such as opening, printing, editing the MX file. The rest of the window is devoted to interact with the currently opened MX file.

In the left part of the window there are also two lists: depending on the selected layer, the upper list shows parts/voices, and the lower one has a contextual role, listing the external files linked to the layer.

The aim of the application is to provide a tool to “navigate” a MX file and its associated media files, maintaining synchronization among all layers. From this point of view, in MXDemo we can load a MX file and open its heterogeneous linked material in distinct windows. MXDemo faces some interesting representational problems, such as structural descriptions for those pieces that can not be represented by traditional notation. For instance, harmonic grid is supported for Jazz improvisation. Besides, virtually any type of graphical representation for music can be mapped in a MX file, even if not explicitly supported by the original format, and this confers great flexibility and power to our descriptive possibilities.

A particularly interesting feature is the possibility to select a particular point of the music (a timing in a media file, a location in a score image, a tag in the MX window), and have all the other media automatically synchronized.

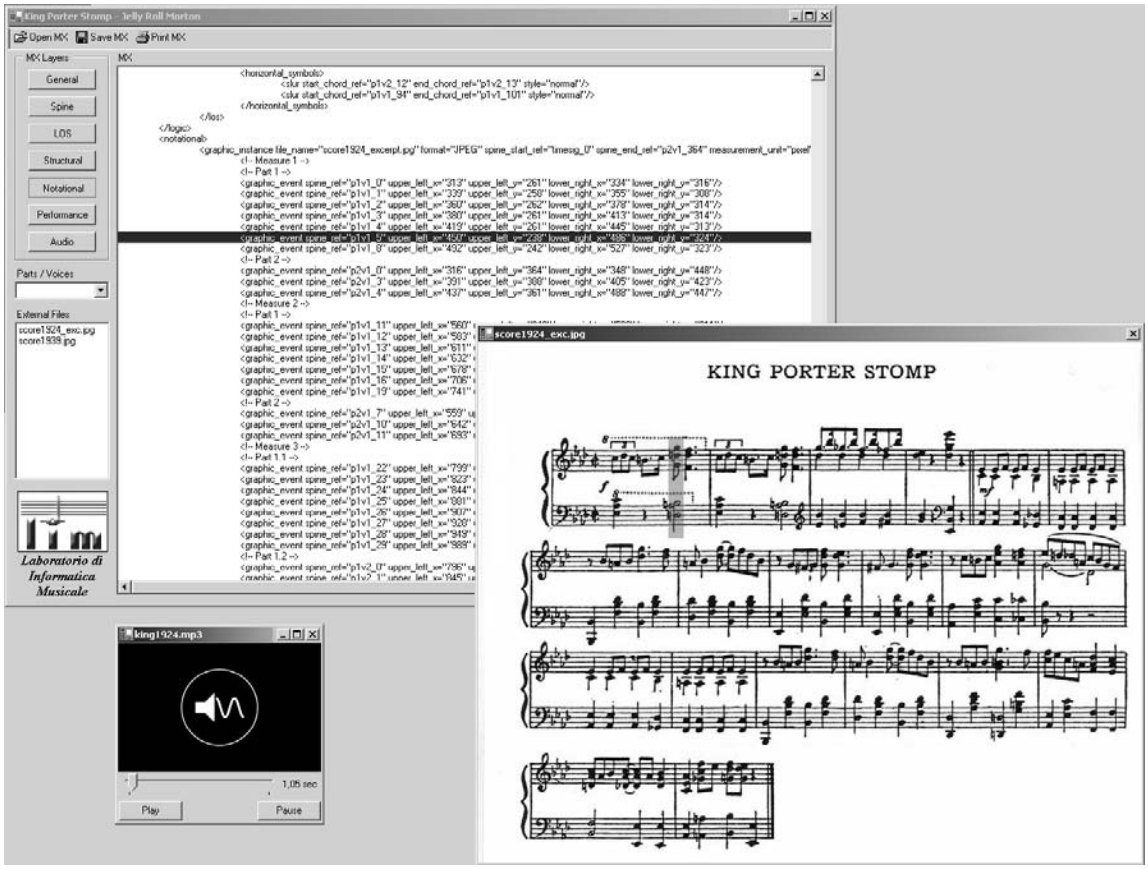

Figure 5. A MXDemo screenshot

6. MX real-time execution

Before describing the engine of MXDemo, a word on real-time MX-layers' synchronization must be spent.

Some MX layers contain only the concept of *relative time*; some others have also the idea of *absolute time*. Finally, recalling what was said before, the `General` layer does not have a time concept at all. The *relative time* concept resembles the execution of a music score: even if the score presents its own internal temporal relationships, in absolute terms music can be performed slowly or quickly. Similarly, the spine of a MX file contains an internal temporization, not in absolute terms (such as seconds) but in relative terms: each event is related to the previous one through a relative time and space distance.

The main layer presenting this form of relative time specification is `Logic/Spine` one. `Logic/LOS`, `Notational`, and `Structural` layers inherit this behavior, as their time parameters are specified simply by linking the spine.

On the other hand, the concept of *absolute time* is present in `Performance` and `Audio` layers. Here, in addition to spine links, the events must have an associated absolute timing (expressed in seconds, or ticks, or frames, depending on the file format). The absolute timing lets the application determine the precise occurrence of a given event within the clip. In this way, when an audio/video file is played, the relative time specifications of the related spine are translated to absolute timings. This mechanism allows to associate different performances to the same logical representation.

In summary, a score contains only relative timing indications (rhythmical values, vertical overlaps), but our mapping lets us associate to a single spine different performances (of the same piece) along with their different agogics.

As a consequence, the real-time execution of a MX file is possible only in two cases: (i) by specifying a relative/absolute time ratio, or (ii) by running one and only one associated performance/clip. Different performances/clips would actualize in different ways the relative time of the spine, and this would represent a problematic issue. In MXDemo, the former solution is not implemented yet, whereas our approach follows the latter method. As a consequence, when the *play* command is activated in a performance/clip window, previously running files are stopped.

This is the only case of non-concurrent execution: all linked objects from all other layers can be shown at the same time and kept synchronized. On the contrary, a performance file (e.g. a MIDI file) can not be played during an audio file (e.g. a MP3 file) execution.

One of the most relevant results of our implementation is showing that current computer performances allow the real-time synchronized execution of several music representations.

7. The MXDemo synchronization engine

How does MXDemo perform the described synchronization among media and between media and spine? When a MX file is opened, the left buttons that match the compiled layers are selectable. By choosing a particular layer:

1. The corresponding first line of MX code is selected in the central text box.
2. The parts/voices list is eventually compiled, and the contextual list is populated too. Double-clicking an item (i.e. a media file) in the contextual list opens the selected object (score, image, audio, video) in a separate window.

A number of user interactions to re-synchronize media are supported: clicking on the music score; scrolling the slider corresponding to the clip position; selecting a tag in the MX text box.

These actions cause a *search* in the MX file to find the selected element in the corresponding layer, and a *moving* of the other media pointers. We will soon discuss the meaning and the operation of *search* and *moving* procedures.

7.1. *Search* procedure in score

Since the music score is mapped by rectangular portions of space, only a click on a covered area of the image triggers a synchronization process (the required information to perform synchronization is already present in the MX file). When a click occurs in the score, the application scans the MX file to find a matching *graphic_event* tag. The corresponding *spine_ref* attribute is saved.

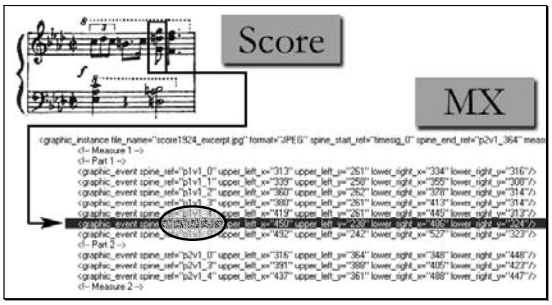

Figure 6. From score to the related MX line

7.2. *Search* procedure in audio/video clip

When scrolling the slider that represents the current position in an audio/video window, the engine searches for the absolute timing position in the corresponding audio/video layer. The nearest *clip_spine_event* element is selected and the corresponding *spine_ref* attribute is saved.

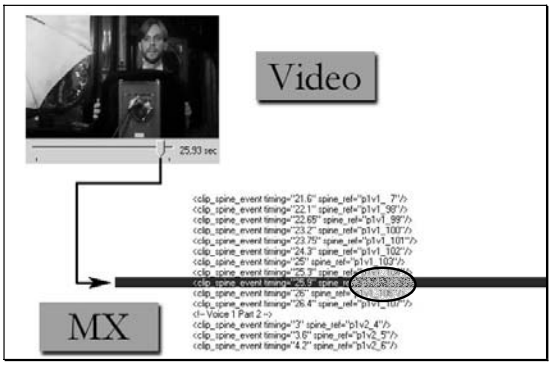

Figure 7. From video to the related MX line

7.3. *Search* procedure in MX

A selection in the MX file generates a synchronization process if the selected line incorporates a timing concept, i.e. has a link to a spine element. In this case the *spine_ref* attribute is saved. Otherwise, the application evaluates the possibility to reconduct the selected line to a related object with its timing information. For instance, by clicking on a *notehead* element, it is possible to reconstruct its timing thanks to the *event_ref* of the corresponding chord.

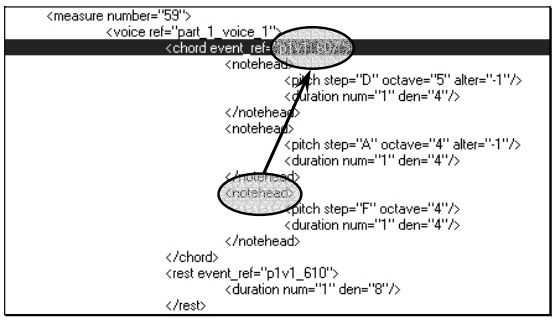

Figure 8. From *notehead* to spine reference

7.4. *Moving* procedure

When a spine reference is found in the *search* procedure, all opened media pointers must be accordingly re-synchronized. This is done by searching that spine identifier in the layers corresponding to the opened media.

As an example of the entire process (see Figure 9), suppose the user clicks on a score image: the engine searches for a *graphic_event* element in the corresponding *graphic_instance*, and saves the value of the *spine_ref* attribute. If an audio file and another score are also opened, that value is searched in their dedicated MX layers, and the two pointers (score and audio) in the corresponding windows are refreshed.

In the case the spine value is not present in the MX portion related to an opened media file, the pointer will be moved to the nearest position.

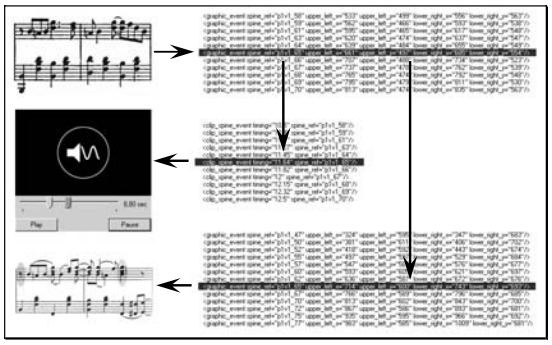

Figure 9. A re-synchronization example

7.5. The real-time execution in practice

All the operations so far described are performable also during a real-time execution of an associated audio/video clip. This is probably the most interesting use of the MXDemo application: when an audio/video file is playing, all the pointers of other opened media are re-synchronized in real-time, offering different (and configurable) points of view of the same material.

The engine implements this feature by using a timer. The timer samples the position of the clip every *t* milliseconds, refreshing all the other pointers. The value of *t* can be computed scanning the `Audio` layer.

Let *d* be the minimum time difference between two consecutive events, then

$$t = d / 2.$$

8. Related works

The MPEG-7 standard, formally called "Multimedia Content Description Interface", represents a way to describe multimedia information in all its aspects, providing a XML framework which allows applications to access and use efficiently audio and video codes. About multimedia audio [2], this standard provides a rich set of tools to describe the information related to physical media and their content from a structural and semantic perspective, dealing both the low (Audio Framework) and the high level. The latter provides the Melody Description Tool, which offers the possibility to describe music melody at symbolic level.

Unlike MX, MPEG-7 attention is focalised both on audio and on video information; at the moment, MPEG-7 is not particularly specialized on multi-layer music description, above all at symbolic level [5]. An effort towards a better music representation in a multi-layered system is getting into MPEG-4 standard thanks to SMR (Symbolic Music Representation) project [1]. The purpose is developing technologies for a complete description of music at symbolic level and for a comprehensive synchronization with audio and video objects provided by MPEG-4 (for example, MPEG-4 Structured Audio codes [3]).

Wedelmusic project is aimed at the development of systems to manage interactive multimedia music. Wedelmusic is particularly interested in copyright issues. Thanks to a proper XML language that describes music at different levels, the provided software allows – for example – the manipulation and the distribution of music. Another supported feature is the visualization of synchronized audio and score. Probably Wedelmusic project concerns mainly multimedia distribution and protection; on the contrary, in MX a central role is played by the music piece, and other multimedia objects are supported as particular aspects of its description. Further

information about Wedelmusic is available at the official Web site (http://www.wedelmusic.org).

9. Future works

The aforementioned approach is based on two key aspects: (i) a multi-layered XML-based format that supports heterogeneous music data and file types, allowing synchronization among layers; (ii) an application that can represent such relationships among layers, possibly in real-time. A great effort should (and will) be made to complete, improve and update both these aspects.

The MX format is currently under development: at the moment, it does not represent a standard encoding. Before the conclusion of the IEEE standardization process, many MX concepts will be further investigated: for instance, support and integration of MPEG files, MIDI-oriented mechanisms of synchronization, and – in general terms – relationships between MX `Logic/LOS` layer and the scores coded in other formats (MusicXML, Csound .SCO, MPEG-4 Structured Audio Score Language).

Besides, the synchronization process to compile a "rich" MX file is currently handmade. In other terms, the synchronization points (in time and in space domain) are determined by human intervention, e.g. performing an accurate waveform analysis or using rudimentary graphic applications. Attempts to perform automatic audio-to-score synchronization are currently under development, both working in the uncompressed domain and in the compressed one.

On the contrary, OMR (Optical Music Recognition) approaches to score mapping are still too weak. However, the final purpose is obtaining a number of software tools to reduce human intervention in the synchronization process. Unfortunately, at the moment a completely automated process is even difficult to foresee.

MXDemo software aspects will be improved, too. The concepts of accessibility and usability are very important in the management of interactive music, and MXDemo will be improved to reflect these ideas. In a future development, the application will be platform independent. Besides, as MX format can be employed to diffuse music on-line, a Web-accessible MXDemo implementation will be released.

10. Acknowledgments

This research has been partially funded by Italian MIUR - Italian Ministry of Education, University, and Research ("WEBMINDS - Wide-scalE, Broadband, MIddleware for Network Distributed Services" FIRB Project N. RBNE01WEJT_005). The authors want to acknowledge researchers and graduate students at LIM, and the members of the IEEE Standards Association WG on MX (PAR1599) for their cooperation and efforts.

11. References

[1] *Call for Proposals on Symbolic Music Representation*, International Organization for Standardization ISO/IEC JTC1/SC29/WG11 N6689, Redmond, USA, 2004.

[2] *MPEG-7 Overview*, International Organization for Standardization ISO/IEC JTC1/SC29/WG11 N5525, Pattaya, Thailand, March 2003.

[3] *Overview of the MPEG-4 Standard*, International Organization for Standardization ISO/IEC JTC1/SC29/WG11 N4668, March 2002.

[4] *Recommended Practice for the Definition of a Commonly Acceptable Musical Application Using the XML Language*, IEEE SA 1599, PAR approval date 09/27/2001.

[5] GOMEZ, Emilia, GOUYON, Fabien, HERRERA, Perfecto, and AMATRIAIN, Xavier, *Using and enhancing the current MPEG-7 standard for a music content processing tool*, Audio Engineering Society Convention Paper presented at the 114th Convention, Amsterdam, The Netherlands, 2003.

[6] HAUS, Goffredo, and LONGARI, Maurizio, *Towards a Symbolic/Time-Based Music Language Based on XML*, in Proceedings of the First International IEEE Conference on Musical Applications Using XML (MAX2002), Institute of Electrical and Electronics Engineers, New York, USA, 2002.

[7] NESI, Paolo, et al., *Music Notation Application Requirements and MPEG Technology*, ISO/IEC JTC1/SC29/WG11, Brisbane, Australia.

[8] ROLAND, Perry, *The Music Encoding Initiative (MEI)*, in Proceedings of the First International IEEE Conference on Musical Applications Using XML (MAX2002), Institute of Electrical and Electronics Engineers, pp. 55–59, New York, USA, 2002.