Chapter XIV
Music Representation of Score, Sound, MIDI, Structure and Metadata All Integrated in A Single Multilayer Environment Based on XML

Adriano Baratè
Università degli Studi di Milano, Italy

Goffredo Haus
Università degli Studi di Milano, Italy

Luca A. Ludovico
Università degli Studi di Milano, Italy

ABSTRACT

In this chapter, we will analyze the heterogeneous contents involved in a comprehensive description of music, organizing them according to a multilayer structure. Each layer we can identify corresponds to a different degree of abstraction in music information. In particular, our approach arranges music contents in six layers: General, Music Logic, Structural, Notational, Performance, and Audio. In order to reflect such organization, we will introduce a new XML-based format, called MX, which is currently undergoing the IEEE standardization process (IEEE SA PAR1599). In an MX file, music symbols, printed scores, audio tracks, computer-driven performances, catalogue metadata, and graphic contents related to a single music piece can be linked and mutually synchronized within the same encoding. The aforementioned multilayer structure allows us to gather and organize heterogeneous contents, leaving them encoded in well-known and commonly used formats aimed at music description.

INTRODUCTION

Nowadays, music is considered an important matter in information and communication technology. Production, reproduction and representation of music by computer are different facets of the same problem: all these terms should be taken into account for a comprehensive description of music. In fact, if we want to provide an accurate and detailed description of a music piece, we cannot consider only its score. On the contrary, our approach requires a deep knowledge about the processes that bring to the final result, including compositional ideas, music structures and relationships, notated signs, sound generation, recording and reproduction.

Today it is possible to produce music and sound also by computer. In this context, the term *production* can mean generation, composition, but also transformation and manipulation of existent audio material. Computer based production is not only allowed, but even made easier and richer as regards expressive possibilities. We can cite the examples of computer aided composition, notation and editing software, digital instruments and effects, and so on.

Not only music production, but also *music reproduction* has gained benefit from digital knowledge and techniques. As regards digital music availability for its consumers, information technology has recently reached good results: let us cite portable and wearable audio devices or the phenomenon of online music sharing. In other words, music availability is now achieved both in space and in time, and music can be enjoyed both by remote and by future recipients.

About music reproduction on digital devices, a significant example is constituted by digital media supports and by file formats that make music available on computers. If we were able to include audio information coming from one or more media files as well as score symbolic representation, the overall description of the music piece would be more complete, opening a number of new scenarios that will be discussed later.

A directly related aspect is represented by music digitalization, in its most comprehensive meaning: not only performances, but also scores, graphic material, and related physical objects. We know that digital information is not intrinsically eternal (see Rothenberg, 1995 for a throughout discussion about the longevity of digital information), however—thanks to digitalization campaigns—documents can be preserved from the wearing out due to time and physical phenomena. After digitalization process, storage and networking technologies allow the preservation, the transmission, and the worldwide diffusion of music.

As regards information and communication technology, the state of the art of music production and reproduction by computer-based systems is noticeably advanced. But what about *music description*?

Before jumping into an in-depth discussion of the matter, let us claim the relevance of computer-based music description. In our opinion, the audience of potential recipients is very wide: music producers (publishers, editors, composers, major, and indie record labels, and multimedia entertainment industries), music consumers (both educated listeners and keens), and finally researchers (analysts, musicologists, etc.) Each of the aforementioned actors faces the problem of music description from a different point of view, and expects an answer to the actor's requirements and demands.

Needless to say, the locution *music description* can embrace a number of different meanings, and understanding its exact sense is the first key problem. When we describe a music work, we usually list the metadata about its title, its author(s), its performers, its instrumental ensemble, and so on; but we could also want to catch the symbols that compose the piece, or give a description of physical objects related to music itself and to music performance; finally, also audio/video recordings

should be considered music descriptions. We will face the matter of a comprehensive description of music in the next section; at the moment, for our purposes it is sufficient to realize that music communication is very rich, thus also the number of its possible descriptions will be high.

Any meaning we assign to the term "description", the problem of describing music in a computer-based system is not a technological one. In general, we can choose the most appropriate between already existing file formats aimed at music description, according to the interpretation we consider. The evidence comes from the large number of available file formats addressed to music enjoyment. For example, AAC, MP3, and PCM formats are commonly used to encode audio recordings; MIDI represents a well known standard for computer-driven performance; GIF, JPEG and TIFF files can contain the results of a scanning process of scores; DARMS, NIFF, and MusicXML formats are aimed at score typing and publishing, and so forth.

So, the problem of music description on computer systems is not technological, rather there exists a theoretical problem. The latter can be summarized through the following key questions: Which facets of music information should be described? Are heterogeneous communication levels involved? Is an encoding format currently available to catch the different aspects of music information? In the next section we will try to answer these questions.

A COMPREHENSIVE DESCRIPTION OF MUSIC

In our opinion, it is necessary to conceive music description in a comprehensive way. Before investigating this concept, let us recall that music communication is made up of many different and complementary aspects: music can be (and actually is) the idea that the composer translates to symbols as well as their performance, the printed score that musicians read as well as a number of other related contents. Let us cite the example of opera houses (Haus, 1998) or music publishers (Haus, & Ludovico, 2005), where a great amount of heterogeneous music documents are available: scores, audio/video recordings, fliers, playbills, posters, photos, stage maps, sketches, and so forth. Our definition of comprehensive description of music embraces all the aspects we have cited.

As we have affirmed before, specific encoding formats to represent peculiar music features are already commonly accepted and used. But such formats are characterized by an intrinsic limitation: they can describe music data or metadata for score, audio tracks, computer performances of music pieces, but they are not intended to encode all these aspects together. On the contrary, we are interested in a "comprehensive" representation and enjoyment of music, addressed to musicologists as well as to performers, to music students as well as to untrained people simply interested in music. The key characteristics that a comprehensive format should support can be summarized as follows:

- Richness in the multimedia descriptions related to the same music piece (graphical, audio, and video contents).
- Possibility to link and perform a number of media objects of the same type (for instance, many performances of the same piece or many score scans coming from different editions).

In addition, we want this comprehensive format to support complete synchronization among time-based contents, meaning that audio and video contents are kept synchronized with score advancing, even when the user switches from a particular performance to another, or from a particular score edition to another.

Interaction should be supported as well. Of course, this aspect cannot be realized by a format, rather by applications working on the

format. The matter will be developed in Section 5, devoted to MX applications. At the moment, in order to provide a broad picture, we underline the possibility for the users to click any point of the score and jump there also in the audio content currently performed, as well as the possibility to navigate the audio track moving a slider control and highlight the related portion of the score.

Achieving the aforementioned goals implies: (1) designing and adopting a suitable format to represent music, (2) encoding music pieces and all the related material in such format, and (3) implementing software applications to achieve synchronization and user interaction.

The first step is represented by the design of a comprehensive representation format for music. A formal and complete analysis of music richness and complexity is provided by Haus and Longari (2005), where six different levels of music description are identified—namely General, Structural, Logical, Notational, Audio, and Performance layers. This multilayer structure, suitable for our concept of comprehensive description of music, is reflected by the encoding format we will describe in Section 4.

XML-BASED REPRESENTATION OF MUSIC DATA AND METADATA

In the previous section, we have provided our definition of "comprehensive description" applied to music. In this sense, the inadequacy of current file formats should be evident. As a satisfactory standard is still unavailable, we propose some guidelines to face the matter, introducing a new approach for computer-based music description.

A possible solution to the problem could reside in the adoption of an XML-based format. There are many advantages in choosing XML[1] to describe information in general, and music information in particular. Many texts and scientific papers have already described the advantages of XML-based formats applied to information representation. Here we are more interested in the reason why XML and music make a good couple.

First, XML is a formal meta-language suitable for declarative representations. It is capable to describe entities (in our case music objects) as they are, without unnecessary information overload. This kind of representation is modular and hierarchical; nevertheless it allows explicit interaction among modules. Internal relationships are formal and explicit. All these aspects have an important counterpart in music language, which is formal and hierarchical too.

XML modularity can solve the problem of a well-organized comprehensive description: each part constitutes a separate plug of the overall description, still maintaining its own identity. However, interdependencies are allowed and made explicit by an XML-based format.

Languages derived from XML are extensible, as they support extensions and external entities. This aspect is fundamental to take into account further developments of music language or still unforeseen uses of the description format.

Finally, XML is open to user contributions, easy to be read, decoded, edited, and understood. In principle, anyone can give suggestions and implement specific parts of the format. Even if easier than binary formats, XML in general is not intended to be managed directly: rather, it should be read and written by computer-based systems. For instance, music XML cannot be printed in its text form in order to be played at first sight by a human performer; on the contrary, we need specific software to decode the format and represent it as a sequence of music symbols. Fortunately, applications to edit XML files can be easily found on the marketplace and usually basic editors are free. Besides, specific software able to work on a peculiar XML-based format can be implemented without paying royalties or licences, as most formats are free.

In conclusion, an XML-based language allows inherent readability, extensibility, and durability. A specific coverage of the matter is presented by

Steyn (2002). These are the main reasons why the format we propose relies on XML in order to represent and organize music information in a comprehensive way.

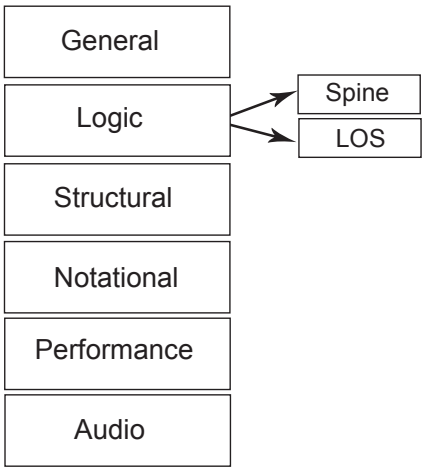
AN OVERVIEW ON MX OVERVIEW AND BASIC CONCEPTS

MX is the code name for a new file format officially called IEEE SA PAR1599[2] (XML Musical Application Working Group, 2001). This project is aimed at a comprehensive symbolic music representation, and opens up new possibilities to make both the message and the internal structure of music available to musicologists, performers and non-practitioners. Its ultimate goal is providing a highly integrated representation of music, where score, audio, video, and related graphical contents can be enjoyed together.

MX inherits all the peculiarities of an XML-based format: for example, it is open, free, easily readable by humans and computers and editable by common software applications. Moreover, some of the typical features of an XML format well suit to music: XML is strongly structured, as well as most music scores; an XML format can be extended, supporting new notations and new music symbols; XML provides a means of interchange for music over the Net.

All the characteristics we have mentioned before represent themselves valid reasons to adopt an XML-based encoding for music. But our own XML format, namely MX, presents further advantages. First, the music description provided by an MX file is flexible and potentially very rich, as regards both the number and the type of media involved. In fact, thanks to our approach a single file could contain one or more descriptions of the same music piece in each layer. For example, if we had to provide a description of an operatic aria, the MX file could house: the catalogue metadata about the piece, its author(s) and genre; the corresponding portion of the libretto; the scans of the autographical version and of a number of printed versions; many audio files containing different performances; related iconographic contents such as sketches, on-stage photographs, and playbills. Thanks to the rich information provided by MX, software applications based on such format allow an integrated fruition of music in all its aspects.

Figure 1. The six-layer MX structure



Another key feature from both a theoretical and an applicative point of view is the full synchronization among layers, involving both time and space dimensions. This characteristic lets the user switch from a representation to another in real-time. For instance, he/she can compare the vocal performance of Enrico Caruso to the one of Luciano Pavarotti, or the signs notated on the autographical score to their translation on the printed version. At the same time, the user can view the structure of the piece he/she is listening to and open a number of iconographic related files.

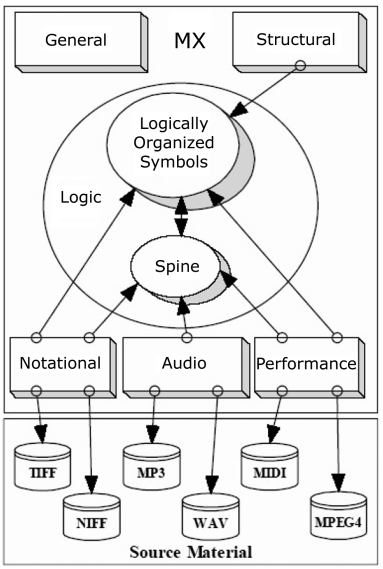
As mentioned before, a thorough description of music must encompass different layers to represent information (see Figure 1):

- **General:** Catalog information about the piece

- **Logic:** The symbols that compose the score as intended by its author(s)
- **Structural:** Music objects and their relationship
- **Notational:** Graphical representations of the score
- **Performance:** Computer-based descriptions and executions of music
- **Audio:** Digital or digitized recordings of the piece

Figure 2 shows the relationship between the group constituted by General, Structural, and Logic layers and the one encompassing Notational (e.g., GIF, JPEG, TIFF for a score), Audio (e.g., AAC, MP3, WAV), and Performance (e.g., Csound, MIDI, MPEG) layers. It should be evident that our approach consists in keeping intrinsic music descriptions inside the MX file (the upper block in Figure 2), whereas media objects are external (the lower block in Figure 2). Let us cite some clarifying examples. The symbols that belong to the score, such as chords and notes, are directly described in XML format, in the so-called Logic layer; on the contrary, MP3 files and other audio descriptions are not translated into XML format,

Figure 2. Relationships among MX layers and external media objects



rather they are linked and mapped inside the corresponding MX layer, namely Audio layer.

Needless to say, not all layers must or can be present for a given music piece. Of course, the higher their number, the richer the piece's description is. Besides, each layer could contain one or more object instances: for example, Audio layer could link 3 audio tracks and Structural layer could provide two different analyses for the same piece. The concept of multilayer description (i.e., potentially many different types of descriptions, all correlated and synchronized) together with the concept of multiinstance support (i.e., potentially many different media objects for each layer) provide a very rich and flexible means of encoding music in all its aspects. Section 5 will illustrate possible applications of these characteristics.

MX layers will be soon analyzed in greater detail and made clearer by some example, but first we have to introduce the main concept MX is based on: the idea of spine.

Spine

According to our comprehensive approach to music description, the MX standard provides different layers to represent information. Of course, these levels cannot be independent: in order to achieve richness in information description, each music event can have one or more descriptions in different layers. Thus, we should be able to create a common data structure in order to link the different representations of each music event within the MX file. For instance, let us refer to a hypothetical event named e12, corresponding to a B quaver somehow described in Logical layer: we should be able to refer to its representation also in audio, video, score parts of the MX file.

Accordingly, we have introduced the concept of spine, which is an overall structure that relates time and space information. Spine contains no more than an ordered list of event identifiers, whereas the events are later described in Logic layer as regards their musical meaning. Events

Box 1. MX spine

```
<spine>
   <event id="timesig _ 0" timing="0" hpos="0" />
   <event id="keysig _ 0" timing="0" hpos="0" />
   <event id="clef _ 0" timing="0" hpos="0" />
   <event id="clef _ 1" timing="0" hpos="0" />
   <event id="p1 _ 0" timing="0" hpos="0" />
   <event id="p2 _ 0" timing="0" hpos="0" />
   <event id="p1 _ 1" timing="256" hpos="256" />
   <event id="p1 _ 2" timing="256" hpos="256" />
   <event id="p1 _ 3" timing="256" hpos="256" />
   <event id="p1 _ 4" timing="256" hpos="256" />
   …
</spine>
```

correspond to the music entities of the piece we want to describe. This definition is intentionally vague, as our approach is aimed at allowing a high degree of flexibility. In general, events are all the notes, rests, time signatures, key signatures, clefs, and symbols of the score; but the aforementioned definition allows also to map only the C notes in the trumpet part or the dotted crotchets of string instruments. The choice of what should be referenced by the spine list of events is very subjective, and depends on the characteristics of the piece and on the purpose of the description. For instance, in jazz improvisation only the harmonic grid can be mapped; in this case, spine contains identifiers for chords instead of including identifiers for single notes.

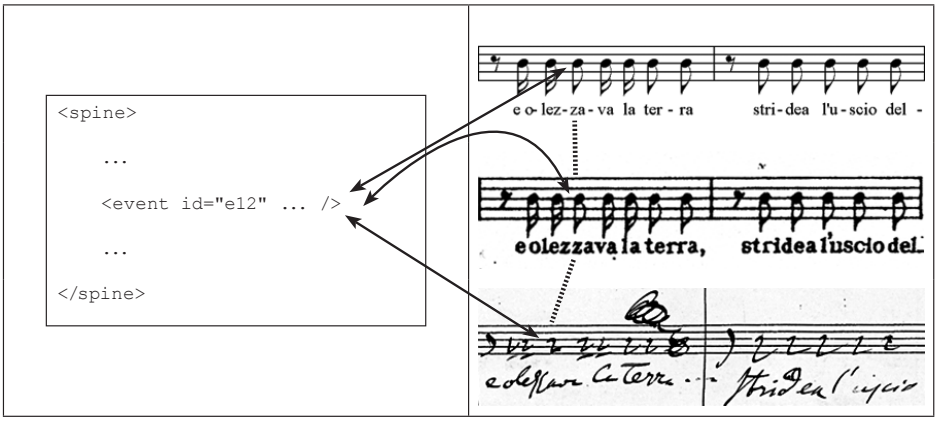
In Box 1, the first 10 events of an MX spine are shown.

A first problem is: how can we put in a one-dimensional ordered list a two-dimensional sequence of events? Notes, rests and other symbols in a score are disposed according to a vertical juxtaposition (related to harmony) and to a horizontal development (corresponding to temporal evolution of music). Our solution is referring each event in spine to the previous one, following a path that scans music in vertical from upper staves to lower staves; when all the simultaneous events have been considered, we move to the next event on the right, and so on. In a certain sense, the list of events achieved by this process "meanders" through the score, from top to bottom and from left to right. Timing and hpos attributes (shown in the example) are calculated in function of previous events: timing = "0" means simultaneous occurrence in time, and hpos = "0" means vertical alignment. Values are expressed in virtual timing units and in virtual positioning units, in order to achieve a high degree of abstraction. In this layer, music is equally spaced in time and space by applying a concept similar to beat spacing. The actual occurrences of music events inside digital objects such as scores and clips will be mapped inside Notational, Performance and Audio layers.

Now we have described what spine contains. But how does spine provide synchronization to the whole multilayer structure? Each music event, later described in any MX layer, is listed inside the spine and associated to a unique identifier. This identifier represents the key that other layers use in order to refer to such particular event. In this way, all the layers reference (and are referenced by) a common structure that relates them from a spatial and temporal point of view. Figure 3 illustrates the principle spine is based on: by a number of links between spine and other layers, it is possible to create a global net of relationships. For the sake of greater clearness, Figure 3 shows the relationships among three graphical representations of the same score[3]: in this case, synchronization takes place among instances of the same layer.

The arrows represent the mapping from and towards Notational layer. Even if this is a simpli-

Figure 3. Mappings of the notational layer and synchronization of the spine event named `e12`



fied example, please note that Figure 3 does not represent a trivial star schema: if we consider the vertical dotted lines, they are the graphical counterparts of a structure intrinsically and automatically produced by spine mappings. Thus, the resulting layout is a star-ring scheme, where the star is given by the relationships among the identifiers listed in spine and their occurrence in other layers, whereas the ring represents synchronization among instances. This process can be generalized to heterogeneous representations aimed at music description, as Figure 4 illustrates.

MX Layers

After presenting the idea of multilayered structure to represent music and the concept of time-space construct to achieve synchronization, let us briefly define the meaning and the contents of each MX layer.

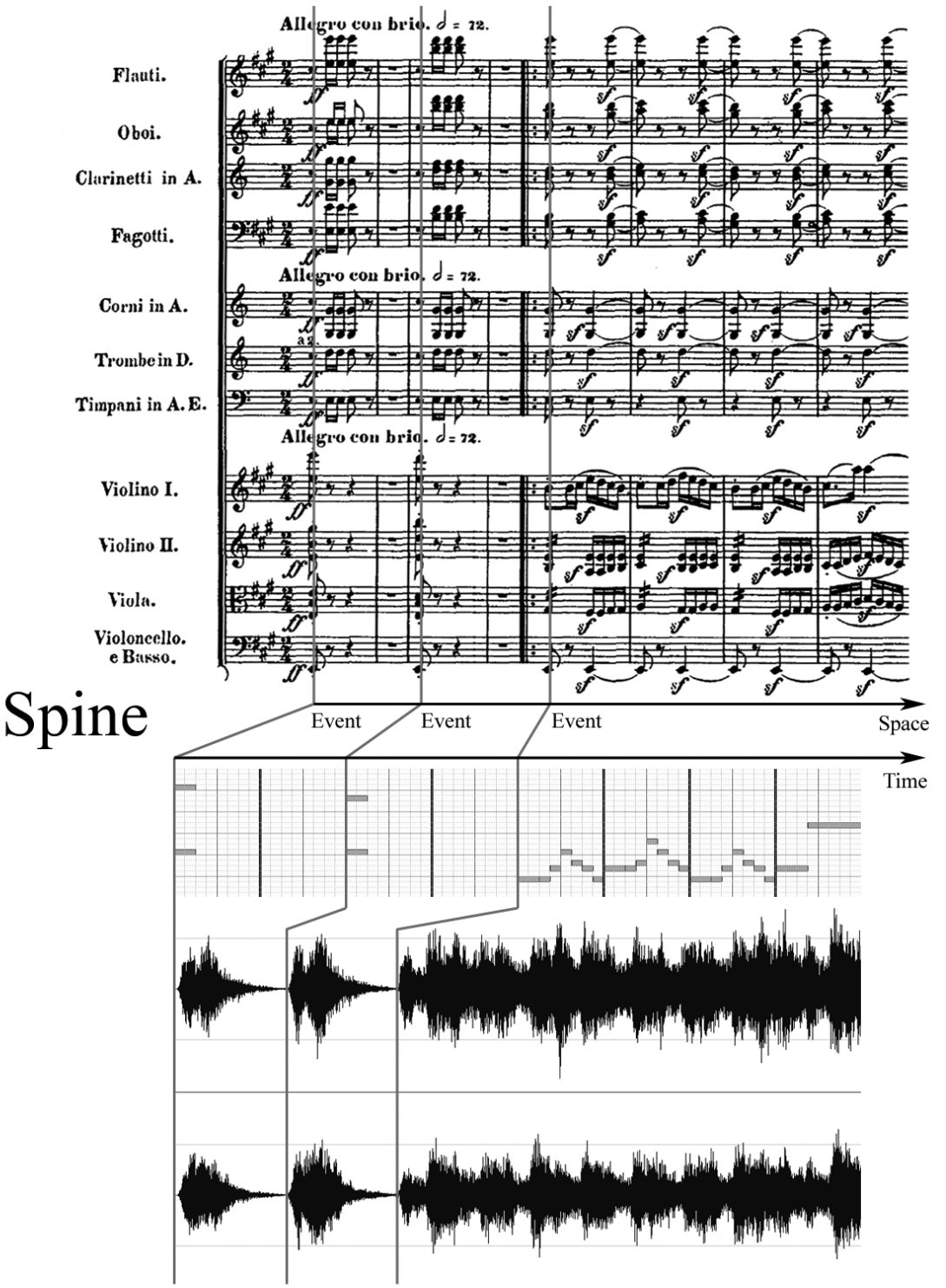
The first MX layer, namely General layer, is not directly related to score and audio contents. It describes some fundamental catalog information about the encoded music work. The situation is reflected by Figure 2, where General block appears disconnected from other levels, even if it clearly belongs to the overall description as well. A particular importance is given to its subelement Description, devoted to author, genre and title information. Such music metadata are not music events, so they are not directly related to music symbols, scores or audio performances; nevertheless, these labels are particularly interesting for music classification and retrieval. The XML block displayed in Box 2 provides an example of catalog metadata.

In addition, General layer can contain links towards files that do not describe directly the piece

Box 2. General layer: Catalog metadata

```
<general>
   <description>
      <work _ title>Tosca</work _ title>
      <movement _ title>E lucevan le stelle</work _ title>
      <author type="composer">Giacomo Puccini</author>
      <author type="librettist">Luigi Illica</author>
      <author type="librettist">Giuseppe Giacosa</author>
   </description>
</general>
```

Figure 4. Synchronization among notational, performance and audio layers provided by spine



Box 3. General layer: Other data

```
<general>
   <description>
   …
   </description>
   <related _ files>
         <related _ file file _ name=”Iconografico\manifesto _ 1.jpg”
                     file _ format=”image _ jpeg”
                     encoding _ format=”image _ jpeg”
                     description=”Manifesto”>
      </related _ file>
      <related _ file>…</related _ file>
      …
   </related _ files>
</general>
```

and its music events, but are strongly related and can provide further information. For instance, related files can contain iconographic material and text contents. Needless to say, such objects cannot be synchronized with music execution, and they do not present any reference to spine structure; nevertheless they belong to our concept of comprehensive description of music.

As regards the aforementioned Puccini's aria, our MX description could provide link towards sketches, on stage photos, libretto, and score covers. Box 3 illustrates this case.

Logic layer contains what the author(s) intended to put in the piece. If we look at a score as a sequence of music symbols, we should refer to its "abstract" layout. For instance, characteristics of a score such as the number of measures per system, page margins, fonts, paper type, and format are described in Notational layer, as they derive from a graphical implementation of the score. In Logic layer we find the same information, but unformatted and disposed on a sort of virtual score. The difference is illustrated by Figure 5, where we can compare an abstract description of a chord to its corresponding printed version: the former is the logical description of the chord, where—among a number of possible descriptions—we have chosen an XML-based one; on the contrary, the latter is the kind of description referred by Notational layer, as we will explain later. Please note that the XML logic description is encoded inside MX Logic layer, whereas the digital image of a printed score remains external and it is only linked and mapped within Notational layer (see Figure 2).

Figure 5. A logic representation of a chord vs. a graphical one

```
<chord event _ ref="p7v1 _ 69">
    <notehead>
        <pitch step="E" octave="5"/>
        <duration num="1" den="1"/>
    </notehead>
    <notehead>
        <pitch step="G" octave="5"/>
        <duration num="1" den="1"/>
    </notehead>
    <notehead>
        <pitch step="C" octave="6"/>
        <duration num="1" den="1"/>
    </notehead>
</chord>
```

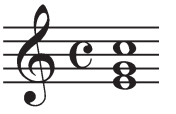
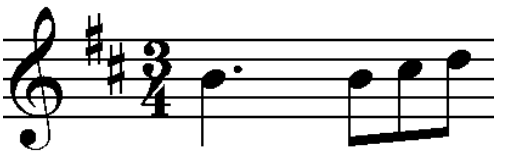


Figure 6. The measure encoded in MX format

```
<measure number="1">
   <voice ref="Clarinetto _ I _ voice _ 1">
      <chord event _ ref="p5v1 _ 0">
         <notehead>
            <pitch step="B" octave="5" />
            <duration num="3" den="8" />
            <aug _ dot />
         </notehead>
      </chord>
      <chord event _ ref="p5v1 _ 1">
         <notehead>
            <pitch step="B" octave="5" />
            <duration num="1" den="8" />
         </notehead>
      </chord>
      <chord event _ ref="p5v1 _ 2">
         <notehead>
            <pitch step="C" octave="6" alter="1" />
            <duration num="1" den="8" />
         </notehead>
      </chord>
      <chord event _ ref="p5v1 _ 3">
         <notehead>
            <pitch step="D" octave="6" />
            <duration num="1" den="8" />
         </notehead>
      </chord>
   </voice>
</measure>
```



Logic layer is composed of two sub-elements: (1) spine, which lists the significant events referable by other layers and (2) logically organized symbols (LOS), that describes the score from a symbolic point of view.

Spine has been illustrated and explained in the previous sub-section. As regards LOS sub-layer, it is based on the hierarchical structure typical of XML formats. Generally speaking, music can be considered strongly structured in a hierarchical fashion: a complete score is made of pages, each page of systems, each system of staves, each staff of measures, and so on. Unfortunately, this tree structure suffers from problems when applied both to staves and to instrumental parts. In fact, the standard situation is having a biunique relationship between staves and parts: a violinist reads the violin staff, as well as a trumpeter performs the notes written on the trumpet staff. But pianists, harpists, and organists have their part usually split into two or even three staves[4]. These examples could suggest that staves are at a lower level of instrumental parts in the hierarchy; in other words, a score seems to be made of parts, and each part takes one or more staves. But all the scores represent a counterexample where many instrumental parts are written on the same staff: for instance, flute and piccolo parts, or three different melodic lines for as many horns. The solution is providing two different structures and a device of crossmapping: an instrumental hierarchy for parts and voices and a layout hierarchy for systems and staves. A part/voice will be assigned to a standard staff, with the possibility to change this setting with single-notehead granularity.

Figure 6 shows a graphical representation of a measure encoded in MX format.

At the end of LOS subelement, after describing symbols such as clefs, notes and rests, there are the sections to represent horizontal symbols (slurs, hairpins, embellishments) and lyrics (Box 4).

Box 4. horizontal symbol representation

```
<lyric font _ type=”Times New Roman” font _ size=”10”
                  part _ ref=”Cavaradossi” voice _ ref=”Voice1”>
   …
   <syllable spine _ start _ ref=”e10” spine _ end _ ref=”e10” hyphen=”yes”>
      <sbtext>e o</sbtext>
   </syllable>
   <syllable spine _ start _ ref=”e11” spine _ end _ ref=”e11” hyphen=»yes»>
      <sbtext>lez</sbtext>
   </syllable>
   <syllable spine _ start _ ref=”e12” spine _ end _ ref=”e12” hyphen=»yes»>
      <sbtext>za</sbtext>
   </syllable>
   <syllable spine _ start _ ref=”e13” spine _ end _ ref=”e13”>
      <sbtext>va</sbtext>
   </syllable>
   …
</lyric>
```

Box 5. Structural layer: Music analyses

```
<structural>
   <chord _ grid description=”Harmony”>
      <chord _ name root _ id=”e1”>I _ 3 _ 5</chord _ name>
      <chord _ name root _ id=”e2”>IV _ 3 _ 6</chord _ name>
      <chord _ name root _ id=”e3”>V _ 3 _ 5</chord _ name>
      <chord _ name root _ id=”e4”>V _ 7</chord _ name>
      <chord _ name root _ id=”e5”>VI _ 3 _ 5</chord _ name>
      …
   </chord _ grid>
   …
</structural>
```

Structural layer contains explicit descriptions of music objects together with their causal relationships, from both a compositional and a musicological point of view. In this context, we can define “music object” as every type of structured information in any musical language. Given this definition, Structural layer identifies a number of music objects and describes how they interact in the overall structure and how they can be described as a transformation of previous music objects.

Thanks to the meaning we assigned at Structural layer, this is the right place to host music analyses. A simple application of music analysis is the identification of chords, whose concatenation gives the harmonic path. In this case, music objects are constituted by simultaneous sets of notes, namely chords (Box 5).

Other applications are the identification of the themes of a sonata form, of the subject of fugues, of particularly important melodic or rhythmic patterns.

Structural layer, thanks to MX multiinstance approach, can contain one or more analyses. In addition, many forms of analysis and segmentation are supported.

Notational layer links all the possible visual instances of a music piece, by organizing single digital objects in collections called graphic or notational instance groups. In this way, for a given piece, MX can link and map not only *n* files belonging to a particular version, but also the files coming from *m* different versions. The *m* x *n* graphic instances are grouped and numbered, to keep trace of their original position in the collection.

Box 6. Notational layer: Spine event mapping

```
<notational>
   <graphic _ instance _ group description=”Partitura autografa”>
      <graphic _ instance file _ name=”Immagini\autografo _ 1.tif”
            format=”image _ tiff” position _ in _ group=”1”
            spine _ start _ ref=”intro _ 0” spine _ end _ ref=”p5v1 _ 10”
            measurement _ unit=”pixel”>
         <graphic _ event spine _ ref=”e0”
               upper _ left _ x=”977” upper _ left _ y=”451”
               lower _ right _ x=”999”lower _ right _ y=”483” />
         <graphic _ event spine _ ref=”e1”
               upper _ left _ x=”999” upper _ left _ y=”458”
               lower _ right _ x=”1011” lower _ right _ y=”483” />
         <graphic _ event spine _ ref=”e2”
               upper _ left _ x=”1026” upper _ left _ y=”462”
               lower _ right _ x=”1042” lower _ right _ y=”493” />
         …
      </graphic _ instance>
      <graphic _ instance file _ name=”Immagini\autografo _ 2.tif”
            format=”image _ tiff” position _ in _ group=”2”
            spine _ start _ ref=”p5v1 _ 11” spine _ end _ ref=”p5v1 _ 44”
            measurement _ unit=”pixel”>
         <graphic _ event spine _ ref=”e30”
               upper _ left _ x=”710” upper _ left _ y=”123”
               lower _ right _ x=”732” lower _ right _ y=”170” />
         <graphic _ event spine _ ref=”e31”
               upper _ left _ x=”738” upper _ left _ y=”90”
               lower _ right _ x=”775” lower _ right _ y=”149” />
         <graphic _ event spine _ ref=”e32”
               upper _ left _ x=”813” upper _ left _ y=”105”
               lower _ right _ x=”836” lower _ right _ y=”154” />
         …
      </graphic _ instance>
   </graphic _ instance _ group>
   …
</notational>
```

Representations mainly belong to two classes: notational and graphical. A notational instance is often in a binary format, such as NIFF[5]; nevertheless, there exist also text-based and XML-based formats. In this case, the occurrence of music events is identified by the offset between the event-related binary encoding and the beginning of file. On the contrary, a graphical instance contains images representing the score, usually in a binary format like JPEG or TIFF. The occurrence of events is expressed in term of space measurement units (e.g., pixels, inches, centimetres) and is directly related to virtual space coordinates in the Spine structure. The way to map spine events in digital images is creating a sort of bounding box for each event, using corner coordinates in pixels or other absolute measurement units. Box 6 illustrates this technique.

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 computer performances. This layer supports symbolic formats such as Csound, MIDI and SASL/SAOL files.

Finally, Audio layer describes the properties of the audio material related to the piece. Once again, many audio/video clips in a number of formats are supported. Here the device used to map music events is based on timing values, as shown in Box 7.

The complete DTD of MX[6] is downloadable at http://www.lim.dico.unimi.it/mx/mx.zip, to-

Box 7. Audio layer: Mapping music events based on timing values

```
<audio>
   <clip>
      <clip _ format file _ name="Audio\Tosca1984.wav"
            file _ format="audio _ wav" encoding _ format="audio _ wav">
         <frequency type="constant" avg _ value _ Hz="44100" />
         <bitrate type="CBR" avg _ value _ bitsec="1536"></bitrate>
         <quantization/>
         <channel channel _ number="2" LFE="yes"></channel>
      </clip _ format>
      <clip _ indexing>
         <clip _ spine timing _ type="sec">
            <clip _ spine _ event timing="0.00" spine _ ref="timesig _ in"/>
            <clip _ spine _ event timing="0.00" spine _ ref="intro _ 0"/>
            <clip _ spine _ event timing="0.00" spine _ ref="intro _ 1"/>
            <clip _ spine _ event timing="1.05" spine _ ref="intro _ 2"/>
            <clip _ spine _ event timing="2.07" spine _ ref="intro _ 3"/>
            …
         </clip _ spine>
      </clip _ indexing>
   </clip>
   …
</audio>
```

gether with complete music examples encoded in MX format.

MX APPLICATIONS

As stated before, MX format allows a rich and comprehensive description of music contents. MX technology represents a base for the development of applications that allow a full musical experience. Richness is just a possibility: a piece could be described only in terms of its music symbols, without multimedia objects attached, and the MX file would be validated in any case. However, if the file has as many related material as possible, a more comprehensive description of music is achieved.

Before enjoying music contents, the MX file has to be created. Like any other XML-based encoding, files in this format can be written and edited even by a simple text editor. It is virtually possible to encode all the score symbols and all the synchronization information by hand and in plain text format. Nevertheless, this approach would be very cumbersome and time-consuming. First, for a musician the task of writing notes and rests according to XML formal rules (that are substantially different from score notation) is unacceptable. Besides, after obtaining a well-formed, valid, complete, and semantically correct encoding of the piece, the author of the MX file should face other complex tasks: for example, the author should manually find the symbols to map, synchronize heterogeneous media objects, and enter those values in the MX file.

Since the central part, and in a certain sense the skeleton, of an MX file is represented by Logic layer, this is the first part to be produced. A practical way to perform this task is implementing conversion tools that take in input widely used formats of music representation and translate them to a basic MX file. At the moment, we have developed an application that converts ETF[7] and MIDI files into MX format. The simple interface of this tool is shown in Figure 7.

This tool prepares a basic MX file as regards only Logic layer, creating a spine structure and LOS sub-layer.

After describing symbolic information from a logic perspective, we have to face the problems related to heterogeneous media linking and syn-

Figure 7. The ETF to MX conversion tool

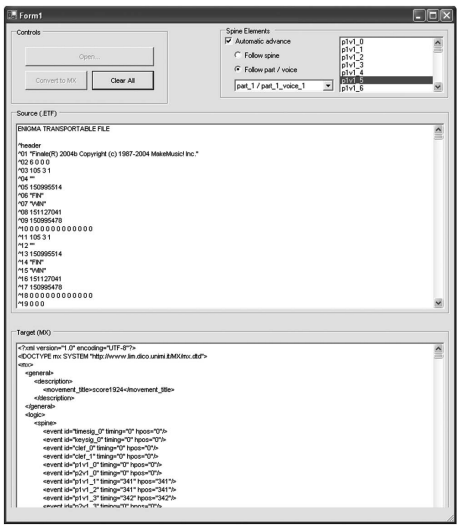
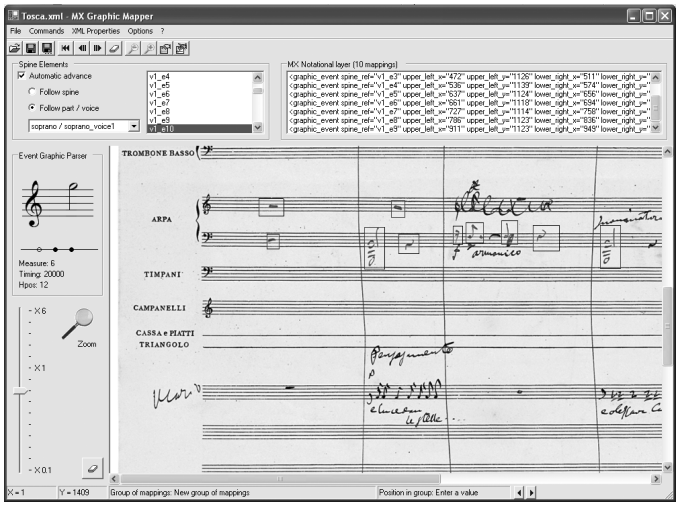


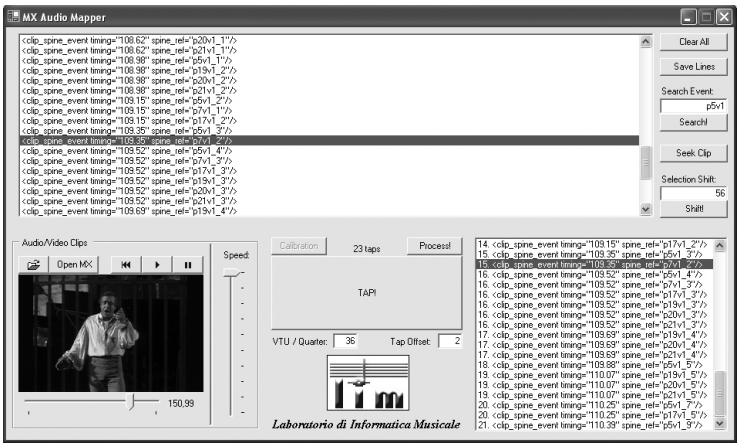
Figure 8. The MX graphic mapper



chronization. As a solution, we have implemented two of utilities to support the creation and management of rich MX files, namely MX Graphic Mapper and MX Audio Mapper. Their general purpose is simplifying the assemblage process of heterogeneous contents within a single music description.

Since a key concept of the MX format is the mutual synchronization of all associated media, the process of adding a certain kind of material cannot be viewed merely as linking a file, but it needs an automatic, semiautomatic, or manual synchronization procedure. Unfortunately, the process required to map and synchronize heterogeneous media at the moment cannot be performed automatically with a high precision degree. As regards notational contents, this would require a

Figure 9. The MX audio mapper



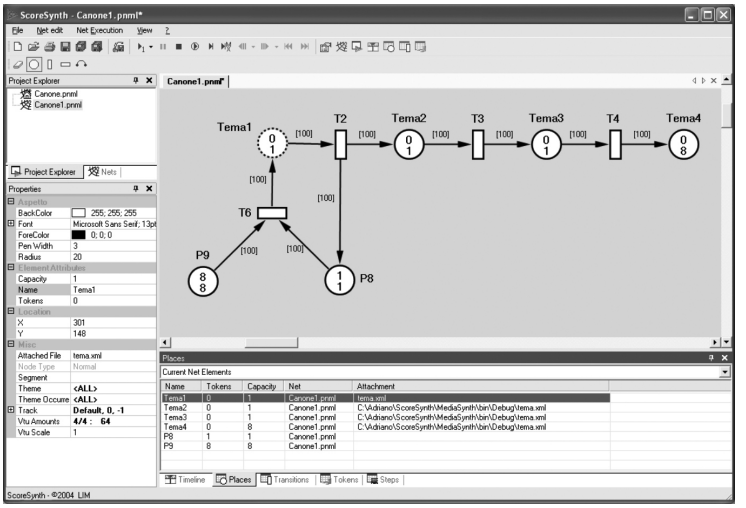
reliable OMR[8] system in order to recognize musical symbols in scores, even when autographical. Besides, such symbols should be automatically put in relationship with the spine structure. As regards audio information, an effective application to extract automatically music events from a complex audio track should be employed, but some well-known limits in automated music analysis techniques prevent the extensive applicability of this approach.

On the other hand, mapping music events by a completely hand-made process would imply a terrible waste of time and energy: it would require, for instance, an accurate listening of the audio tracks in a sound editing environment, or the precise computation of the "bounding box" around each music event in a digital imaging software. Calculating milliseconds and pixels by hand is neither effective nor efficient. A better solution is designing and implementing some aiding applications to speed up the mapping process.

MX Graphic Mapper is the application devoted to link the logic events of an MX file to their corresponding graphic counterparts within one or more score images (see Figure 8); in this way, the final MX file will contain a new block of information defining where a particular event—a note, a rest, a clef, and so forth—is graphically located in the score representation. Through this tool, an existing MX file that needs to be mapped is opened and scanned, and a list of all note/rest events is created. The user can open one or more images that contain the score in order to map all the events. The mapping procedure is done by dragging rectangles on the score image loaded in the central window, while the representation of the note/rest event to be mapped is graphically shown in the "Event Graphic Parser" and the XML fragment to be added is visible in the upper part of the interface. When a rectangle is created, the application generates the XML line by reading the current event in the "Spine Elements" list box, and by computing the coordinates of the rectangle. After adding a new line, the current element indicator is moved forward and the mapping process continues. When this procedure comes to an end, each logic event in the LOS part of the original MX file has at least one element associated in Notational layer, indicating where it is represented in the score image.

On the other side, MX Audio Mapper is the application used to map the logic events in audio/

Figure 10. The ScoreSynth application

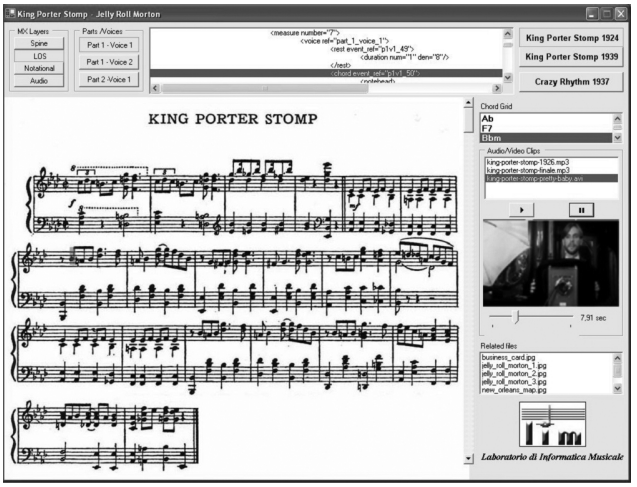


video clips (see Figure 9). This software is similar to the previous one, but—instead of mapping the graphic representations of scores—it maps audio/video performances. The mapping process is achieved by "tapping" the rhythm on a button. To map a clip, some parameters must be defined in advance: the timing unit per quarter used in Logic layer and the rhythm figure to be processed (crotchet, quaver, etc.). The user opens the MX file (already containing a Logic layer) and the audio/video clip and starts hitting the "TAP!" button (with the mouse or the space bar): the result is the hand-made synchronization of the selected rhythmic figures. When this process is completed, the MX spine is processed and all the timings of the events are computed (in seconds), interpolating events between two consecutive taps. Finally, the computed points of synchronization can be fine-tuned by hearing the selected position in the clip and adjusting the wrong timings.

As stated in Section 4, the MX format has a layer devoted to structural information. The software tool, which we have developed to compile this layer, is ScoreSynth (see Figure 10). This application works with Petri Nets (Petri, 1976), a formalism used at LIM since 1980 as the basic tool for music description and processing. In Music Petri Nets we manage music objects (MOs), that is anything that could have a musical meaning and that we consider an entity, either simple or complex, either abstract or detailed. Such entities, identified by unique names, can present some relationship with other music objects. We can treat music objects at various abstraction levels within a hierarchical context of description, applying to them transformation algorithms in order to create new modified objects (Haus, 1997). Through ScoreSynth, a user can describe the structure of a music piece coded in MX format simply drawing the corresponding Music Petri Nets. The model created by ScoreSynth is saved in MX too, and it is linked to Structural layer of the file itself.

An important advantage of the MX standard is represented by its usability in a multimodal environment. Thanks to the multilayer approach in music description, we have developed a number of applications devoted to MX enjoyment, that is, to process an MX file in order to offer an integrated environment where the user can appreciate the advantages of the MX format. In the following,

Figure 11. The MX jazz demo



two applications will be presented: one dedicated to jazz compositions and another one based on an aria from Puccini's *Tosca*.

MX Jazz Demo (Baratè, Haus, Ludovico, & Vercellesi, 2005), shown in Figure 11, features two jazz pieces: *King Porter Stomp*, with two versions of scores and three mapped audio/video clips, and *Crazy Rhythm*, constituted by some improvisations on a fixed harmonic grid. The aim of the application is providing a tool to "navigate" an MX file and its associated media files, maintaining synchronization among all layers. From this point of view, in MX Jazz Demo we can load an MX file and open its heterogeneous linked material. When a linked clip is played, other media are shown synchronously: there is a small rectangle which highlights the note being played in the score part of the window, while the active part of the MX file is shown, and the Chord grid advances. The main part of the window is dedicated to represent the current score. On the top the currently playing MX element is shown, according to the selected layer.

MX Jazz Demo faces some interesting representational problems, such as structural descriptions for those pieces that cannot be represented by traditional notation. For instance, harmonic grid is supported for jazz improvisation. Besides, virtually any type of graphical representation of music can be mapped in an MX file, even if not explicitly supported by the original format (as in *Crazy Rhythm*); this confers great flexibility and power to MX descriptive possibilities. An interesting consequence of our approach is the possibility to select a particular point of 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 in real time, thus allowing an integrated and evolved enjoyment of music contents.

Before describing the second application aimed at user interaction with music contents, some words on real-time synchronization among MX layers must be spent. In MX, some layers contain only the concept of *relative time*, some others have also the idea of *absolute time*, and layers such as General do 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 an 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 time specification is Logic layer. LOS, Notational, and Structural layers inherit this behavior, as their time parameters are specified only in function of 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 absolute timing associated (expressed either in seconds, ticks, or frames, depending on the file format). The absolute timing lets the application determine the precise time occurrence of a given event within the clip. In this way, when an audio/video file is played, the relative time specifications of the spine are translated to absolute timings. This mechanism allows association of different performances (with different absolute time values) to the same logical representation (where events are characterized by relative time relationships).

In summary, a symbolic score contains only relative timing indications (rhythmical values, vertical overlaps), but our mapping lets us associate different performances of a piece to a single spine along with their different agogics. As a consequence, the real-time execution of an MX file is possible only in two cases: (1) by specifying a relative/absolute time ratio, or (2) by running one and only one associated performance/clip. In fact, multiple simultaneous performances/clips would actualize in different ways the concept of relative time, and this would represent a problematic issue.

In MX Jazz Demo, our approach follows the latter method. As a consequence, when *play* command is activated in a performance/clip window, previously running files are stopped. This is the only case of non-concurrent execution: all the linked objects from all the other layers can be shown at the same time and kept synchronized. On the contrary, a file containing a temporized performance (e.g., a MIDI file) cannot be played during the execution of another audio or video file (e.g., an MP3 file).

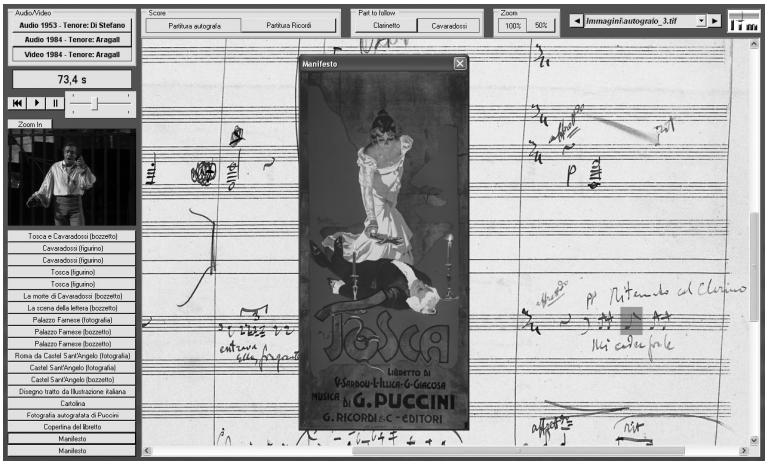
The second MX demo aimed at music enjoyment was installed at the exhibition "Tema con Variazioni. Musica e innovazione tecnologica" (Theme with variations. Music and technological innovation), a voyage into the Italian musical heritage through the rooms held at Rome's Music Park Auditorium in December 2005. All the images shown are courtesy of Italian publishing house Ricordi[9], which owns the manuscripts of the most important Italian composers of the last two centuries.

One of the purposes of the exhibition was making music tangible and visible bringing together the five senses, not just hearing. In this context we have designed a simple user interface, conceived for untrained people, in order to listen to and visualize a track alongside variously interpreted scores. This application, entitled MX Navigator, represents the natural evolution of MX Jazz Demo. The main differences between MX Jazz Demo and MX Navigator are the following two:

- The latter is a generalized version of the former, as MX Jazz Demo was designed only to demonstrate MX format characteristics and worked on the limited number of pieces consequently chosen, whereas MX Navigator is virtually able to open any MX file. In fact, this new version is able to adapt itself in real time to the different kind and number of media contained in the MX file[10].
- MX Navigator is necessarily characterized by an improved usability and by a simpler user interface. In fact, MX Jazz Demo had to be presented to a scientific audience by its authors, whereas MX Navigator is at untrained users' disposal in public exhibitions.

MX Navigator allows an integrated enjoyment of different media contents related to the same

Figure 12. The MX Navigator



music piece. For the exhibition held in Rome, we translated into MX the aria "E lucevan le stelle…" from Puccini's *Tosca* – III Act. The choice of the piece was imposed by the *leitmotiv* of the exhibition "Tema con Variazioni", nevertheless MX Navigator could open and play any MX file.

Apart from a number of multimedia objects such as greeting cards, playbills, historical photos and sketches, the software mainly shows two versions of the score (an autographical and a printed one), a video and two audio performances of the aforementioned aria.

The central part of the interface contains the score in one of its versions, since this is the main media in terms of interaction and visual extent.

The upper left part of the window hosts the controls related to audio/video interaction. In this application, three different clips are shown: an audio clip of a 1953 version performed by the Italian tenor Di Stefano and both an audio and a video clip of the 1984 version performed by Aragall at Verona's Arena. When a particular version is selected, the music and/or video clip is played, and in the score window a red rectangle highlights the current event (thanks to the synchronization achieved by MX Audio Mapper and MX Graphic Mapper).

Since there are many instrumental parts in the score, and they all have been coded into the MX file, it should be possible to choose any part to follow. But this application is intended to be used even by nonmusicians, so the choice is simplified and only two parts can be selected. Thus, in the top of the interface there are two buttons that control which part can be followed by the red rectangle in the score: either the Clarinet or the Tenor (Cavaradossi). Please note that this limitation has been discarded in more recent versions of the application.

In the upper area of the interface, the controls are devoted to manage score visualization. This MX file includes two score instances: the autographical version by Puccini and the Ricordi printed version. The upper left buttons switch (even when running an audio/video clip) between this two scores. The upper right buttons control the zoom level (50% or 100%) and the current page to be displayed. Of course, this takes place only in the Pause state; otherwise the current page is automatically selected by the audio/video clip execution.

The bottom left part of the interface is simply used to open some iconographic objects related to the piece.

MX Navigator provides a visual interface that supports different ways to enjoy music. First, it is possible to select a score version, an audio track, and a leading instrument and simply follow the instrument part evolution. This is a basic level of music enjoyment, and yet music can be listened and watched in a synchronized fashion. But a second way to enjoy music through MX Navigator is even more interesting: it consists in switching from an aural/visual representation to another. In other words, it is possible to compare in real time different versions of the score (the hand-made and the printed one) and different performances. When the user decides to switch from a representation to another, MX Navigator continues just from the point previously reached. Finally, the application suggests a third way to enjoy music, by the possibility to alter the original sequence of music events. For instance, it is possible to jump – forward or back – from a point to another point of the score, both in its visual and aural representations, maintaining the overall synchronization.

RELATED AND FUTURE WORKS

The idea of representing music for computer applications goes back several decades, as shown by Plaine-And-Easie Code (Brook, 1970) and DARMS (Erickson, 1975).

As regards recent attempts to face the problem of a comprehensive description of music, we can cite the Standard Music Description Language (SMDL) (Newcomb, 1991) and the Music Encoding Initiative (MEI, by Perry Roland) (Roland, 2002). The former provides a markup language for any kind of music, of any time, of any geographical region, of any style; the latter is primarily concerned with the markup of all the written forms of musical expression, and in particular those referable to common music notation. Though well defined, SMDL probably failed to attract much attention because of lack of applications. On the other hand, MEI strives to be a sort of framework for music description languages, and MX was deeply influenced in its main peculiarities by Roland's suggestions.

Presently, there are some de-facto standards based on XML. MusicXML is a proprietary standard by Recordare[11], used in dozens of existing applications on the market, including the aforementioned MakeMusic Finale. This standard has been in existence for over four years, and new Version 1.1 has been recently released. Even if MusicXML provides an XML-based description of music, its approach is different from MX's one as the former is aimed at a complete description of the logic and the notational aspects of music, disregarding other features such as structural and audio contents. However, it is interesting to note that most current attempts are based on the markup concept, which confirms that XML-based languages can provide an effective representation of music.

Currently, the most prominent international initiatives to encode music in all its facets are referable to IEEE Standard Association Working Group on Music Application of XML and to ISO/IEC JTC1/SC29/WG11 Audio Subgroup.

Since 1992, the IEEE Computer Society has supported the establishment of a Technical Committee on Computer-Generated Music. The purpose was investigating the vast interdisciplinary area of computer science, electrical engineering and music: a field that ranges from artistic music —composed or played with computers—to audio signal processing (Baggi, 1995). Nowadays the efforts of the working group are mainly aimed at creating the MX standard. The MX project, officially named "Definition of a Commonly Acceptable Musical Application Using the XML Language", was accepted by IEEE and labelled PAR1599.

We can find a similar target in the Call for Proposals on Symbolic Music Representation, submitted by ISO/IEC JTC1/SC29/WG11 Audio Subgroup and already approved. This workgroup

solicits technology responses that propose solutions for the specification of a Symbolic Music Representation. The resulting format should be capable of coding symbolic representations of music in a manner that allows its integration in MPEG-4 (International Organisation for Standardisation, 2004). Even if "MPEG-oriented", such declaration of intents is aimed at the same purpose of IEEE: an approach to overcome the limits of partial descriptions of music by establishing relationships with other data and metadata.

Some differences between the two approaches should be evident. ISO/IEC considers music as one media in a multimedia context, and establishes links towards other multimedia levels. On the other hand, IEEE is working on a standard uniquely aimed at music description, and in such encoding music in treated in a multilayer environment. Also in IEEE format audio, video, and text are supported, but the role played by the music piece is central. IEEE proposal is based on links towards other file formats: audio, video, and text will be treated as a part of music information; whereas ISO/IEC approach is aimed at the integration of a symbolic music layer in the already existing MPEG standards.

The two perspectives are different concerning implementation, but quite similar about theoretical motivations. The fact that both IEEE and ISO/IEC are working on a comprehensive music description makes us sense the importance of this matter in computer music field.

As regards future developments of our research, some aspects of our work should be further investigated, even if the overall approach can be considered stable and consolidated.

First, supported music notation should be completed and integrated with symbols coming from cultures different from Common Western Notation. A detailed test phase is required to validate the completeness and the effectiveness of the adopted data structures. Tests should embrace music from all historical periods, styles, and geographical regions. Needless to say, jazz music grammar is different from early neumes, tablatures, or 20th century experimental notations. In our ambitious attempt to describe all the aspects of music in a comprehensive way, each type of music should be supported.

Second, a number of contributions coming from other XML-based formats should be integrated: we can cite formalisms to represent music in Braille language, or descriptions of intrinsic music structures through Petri Nets Markup Language (PNML).

Finally, some MX layers are not completely developed. A particular attention will be devoted to the problem of genre classification. In particular, General layer will be soon enriched to achieve a flexible and Semantic Web compatible representation of the metadata associated with MX resources.

Another research field will be the investigation of music structures, their relationships and the ways to describe them. The results of this study will enrich MX Structural layer, taking into account the opinions of experts in information technology as well as in musicology.

ACKNOWLEDGMENT

The MX standard, on which this work is based, has been sponsored by the IEEE CS Standards Activity Board and financially supported by the global fund Intelligent Manufacturing Systems (www.ims.org). The authors want to acknowledge researchers and graduate students at LIM, and the members of the IEEE Standards Association Working Group on Music Application of XML (PAR1599) for their cooperation and efforts. Special acknowledgments are due to: Denis Baggi for his invaluable work as working group chair of the IEEE Standard Association WG on MX (PAR1599); Mariapia Ferraris and Cristiano Ostinelli (Ricordi) for their fundamental contributions as regards the original material (autographic and printed scores, sketches, pictures, photos) used in MX Navigator.

REFERENCES

Baggi, D. (1995, November). Presentation of the Technical Committee on Computer-Generated Music. Computer.

Baratè, A., Haus, G., Ludovico, L. A., & Vercellesi, G. (2005). MXDemo: A case study about audio, video, and score synchronization. In *Proceedings of Axmedis 2005*.

Brook, B. S. (1970). The Plaine and Easie code. *Musicology and the computer* (pp 53-56). New York: City University of New York Press.

Erickson, R. F. (1975). The Darms project: A status report. *Computers and the Humanities, 9*(6), 291-298.

Haus, G. (1997). Describing and processing multimedia objects by petri nets. In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE Computer Society Press, 3906-3911.

Haus, G. (1998). Rescuing La Scala's music archives. *Computer, 31*(3), 88-89.

Haus, G., & Longari, M. (2005). A multilayered time-based music description approach based on XML. *Computer Music Journal*. MIT Press.

Haus, G., & Ludovico, L. A. (2005). The digital opera house: An architecture for multimedia databases. *Journal of Cultural Heritage, 7*(2), 92-97.

International Organisation for Standardisation (2004). ISO/IEC JTC1/SC29/WG11, Call for Proposals on Symbolic Music Representation. Redmond, USA.

Newcomb, S. R. (1991). Standard music description language complies with hypermedia standard. *IEEE Computer*, 76-79.

Petri, C. A. (1976). General net theory. In *Proceedings of the Joint IBM & Newcastle upon Tyne Seminar on Computer Systems Design*.

Roland, P. (2002). The music encoding initiative (MEI). In *Proceedings of IEEE 1st International Conference on Musical Application Using XML (MAX 2002)*. Milano, Italy.

Rothenberg, J. (1995). Ensuring the longevity of digital documents. *Scientific American, 272*(1), 42-47.

Steyn, J. (2002). Framework for a music markup language. In *Proceedings of IEEE 1st International Conference on Musical Applications Using XML (MAX2002)*. Milano, Italy.

XML Musical Application Working Group (2001). Recommended practice for the definition of a commonly acceptable musical application using the XML language. IEEE SA 1599, PAR approval date 09/27/2001.

ENDNOTES

1. XML stands for eXtensible Markup Language. XML is a W3C-recommended general-purpose markup language for creating special-purpose markup languages.
2. Institute of Electrical and Electronics Engineers Standards Association, Project Approval Request 1599.
3. Here and in the following examples, we refer to the aria "E lucean le stelle…" from Giacomo Puccini's *Tosca*. The iconographic material shown in this paper and in the software applications we will describe is property of Archivio Storico Ricordi (Milan, Italy) and is copyrighted.
4. In piano pieces by Claude Debussy, piano part is sometimes arranged in three staves (e.g., in the piece entitled *D'un cahier d'esquisses*).
5. NIFF stands for Notation Interchange File Format. It is a file format primarily designed for music typing and publishing. Another key element is transferring music notation among different score writers.

[6] Current MX release, namely version 1.6, was updated on September 2006.

[7] ETF stands for Enigma Transportable File, a format produced by the well-known notation software MakeMusic Finale. An ETF file mainly contains score information.

[8] OMR stands for Optical Music Recognition.

[9] http://www.ricordi.it.

[10] As a matter of fact, MX Navigator at the moment has been employed in three other exhibitions, with different music contents: "Antonio Vivaldi e il suo tempo" (National Library of Turin, February – June 2006), "Ottobre: Piovono Libri" (Bari, October 2006), and "Celeste Aida" (Teatro alla Scala, Milan, December 2006 – January 2007).

[11] http://www.recordare.com/xml.html.