There are six aspects into which music information can be divided: general, structural, music logic, notation, performance, and audio. We call these aspects "layers," because each represents a different level of abstraction of the music information. However, these layers can be viewed as a single symbolic music information (SMI) entity. The purpose of SMI is to relate all existing representations in the notation, performance, and audio layers using the music logic and structural layers.

In this article, we present an Extensible Markup Language (XML) instance intended for the integration of the different aspects of music representation. It tries to bring together ideas and concepts developed in the past. Because this format is still under development by the IEEE-SA Working Group on Music Application of XML (IEEE-SA MAX WG; see www.lim.dico.unimi.it/IEEE/XML.html), and because in an article it is impossible to give a detailed description of a whole format, we present only the main concepts that most probably will be affected only by minor changes by future developments of the format itself. These concepts are Layered Symbolic Music Information (Layered SMI) and the Spine structure. Owing to these two concepts, we consider the contribution of our format to be a more complete integration of previous concepts and formats in a framework usable by diverse music applications, especially those that are based on different concurrent music layers (for example, the automatic synchronization of audio, MIDI, and score).

Layered SMI is important owing to the manifold nature of music representation. Downie (2003) explains this concept in a very meaningful way. According to Downie, music is composed of seven facets: pitch, temporal, harmonic, timbral, editorial, textual, and bibliographic. Moreover, each facet can interact with the others, increasing the complexity of the representational challenge. In addition, music representation also poses multi-representational, multi-cultural, multi-experiential, and multi-disciplinary challenges.

Among the several existing formats for music representation, there are only a small number of them that can be said to be a de facto standard. If we compare these few music formats, we can observe that each of them is designed to represent mainly a particular aspect or only a limited number of aspects of music information. We can subdivide these formats in four big clusters: audio, sub-symbolic, notational, and compositional, similar to the domains of Standard Music Description Language (SMDL; Sloan 1993; see also ftp.orl.gov/pub/sgml/WG8/SMDL/10743.ps).

Audio formats encode signal information, that is, only the purely aural aspect (the “Gestural” domain of SMDL). Sub-symbolic formats like MIDI [Musical Instrument Digital Interface; MIDI Manufacturers Association 2001] or Csound (Boulanger 1999) encode information about how to produce or reproduce music electronically (the “Logical” domain of SMDL). Many music notation file formats have been developed by the different music editing software producers. Some of them are rich enough, like NIFF [Notation Interchange File Format; see www.musique.umontreal.ca/personnel/Belkin/NIFF.doc.html] or Enigma, to generate a MIDI rendering of notational content (the “Visual” domain of SMDL). However, new software composition tools (e.g., Haus and Sametti 1991; Assayag et al. 1999) need to formalize and exchange information and structures that are not represented in these formats (the “Analytical” domain of SMDL). Moreover, the emerging possibility of wide dissemination of music via the Internet increases the urgency of dealing with the problems of cataloging and protecting intellectual rights of these items.

In the past, many researchers have addressed the problem of representing different aspects of music. We think that the most conceptually meaningful of these attempts was SMDL (Sloan 1993; see also ftp.orl.gov/pub/sgml/WG8/SMDL/10743.ps). Although it is not explicit in many existing representations, there is an intrinsic space–time relationship...
in music that can be seen as a bi-directional mapping function between the space and time domains (e.g., disposition of notes on the staff versus timing of notes in an audio file). This relationship is an underlying structure that holds the layers together like glue. We call it a “spine,” because it functions as a backbone for the music work. This concept, as far as we know, was first used by Gomberg (1975), who based a system for electronic music publishing on a similar structure called Spine. Our conception of a spine is composed of events, each of which has a reference in the time domain and a reference in the space domain. Using this concept of the spine, we have devised a method of joining each of the different file formats into a single structural entity.

We think that there are many benefits that can emerge from this unified conception of music representation. Musicians benefit from this work because they can handle each representation of a piece of music as a unique entity. It is possible to build software that uses information stored in the spine to visualize the score (information from the notational layer) while playing a particular compact disc recording of that piece (information from audio layer). Music analysts benefit from this work because they can have the complete control over the notational information. For example, they can browse the score (information from notational layer) and easily find the point in different recording of that piece (information from audio layer). Music vendors can have a database where each piece of music is stored along with different recordings, each one with their own properties, but all related to one product. Music producers can benefit from this work because they can set up a music database in which all the material produced (e.g., scores, MIDI parts, and different tracks of individual instruments) can be handled in a standard and structured way. For example, they may create a web service to which the user connects, selects a piece of music, and receives a MIDI rendering of the piece (information from performance layer) to practice along with and a visualization of the score from which to play (information from notational layer). Surely, there are many more different applications in which this unified conception of the layers is useful, and many other implementations can be imagined.

As we said before, this work is partially under development within the framework of the IEEE SA Working Group on Music Application of XML (www.lim.dico.unimi.it/IEEE/XML.html). This group is devoted to developing an XML application that defines a standard language for symbolic music representation. The language will be a meta-representation of music information for describing and processing music information within a multi-layered environment. It will achieve integration among structural, score, MIDI, and digital audio levels of representation. Furthermore, the main goal is to define a recommended practice for the definition of a commonly acceptable music application using the XML language that integrates music representation with already defined and accepted common standards. The standard should be accepted by any kind of software dealing with music information, e.g., score editing, optical music recognition (OMR) systems, music performance, musical databases, and composition and musicological applications.

In the following sections, we first illustrate some important related work, and we explain some benefits of the representation of music information by means of XML. Then, we briefly survey our prototype XML Document Type Declaration (DTD), called MX, designed for the representation of music information. (Some implementations of MX are outlined in Longari 2004.) We describe the concepts of layered SMI and its representation in XML syntax (see also www.w3.org/TR/REC-xml). Later, some related topics and open problems are discussed.

**Related Works**

We have already mentioned SMDL as perhaps the closest work to our format. There are, however, three important technologies very similar to the MX format: MusicXML, developed by Good (2000); the Music Encoding Initiative (MEI), developed by Roland (2002); and WEDELMUSIC (Bellini and Nesi 2001), which is the proposal currently under development by the MusicNetwork European Project submitted to the MPEG committee. MusicXML is an XML instance intended as an interchange format sufficient for notation, analysis,
trieval, and performance applications, and, in fact, it is the most widely implemented in music notation programs. The content represented by the format is score-oriented, i.e., notes are represented as symbolic and graphical objects. MusicXML addresses the integration of the performance and notational aspects of music, but it does not address the integration of other layers such as audio or structural.

MEI is an attempt to create a TEI-like [Text Encoding Initiative; TEI-C] format for music [see www.tei-c.org]. The main goal of TEI was the creation of a comprehensive yet extensible standard for the encoding and transmission of textual documents in electronic form. Roland is also the developer of MusiCat [Roland 2000], a format for the representation of music catalog information. MEI is a format for representing symbolic music information. It is a rich format with a large number of elements and attributes and designed to be comprehensive, declarative, explicit, interpreted, hierarchical, formal, flexible, and extensible. Attributes are subdivided in domains like those of SMDL (i.e., logical, visual, gestural, and analytical), and the DTD allows the definition of new ones. It addresses the representation of different aspects of music but not the integration in a way that can be useful for applications.

Layered information is partially covered by the WEDELMUSIC [Web Delivering of Music] format that explicitly encodes information on links to score images, notation, timed symbolic data, audio files, and video files. The synchronization aspect is left to the application implementing the language. This work is now considered in the context of the “MusicNetwork” European Project. This project is also in the process of submitting a proposal to the MPEG committee for the integration of music notation with MPEG-7 [Nesi et al. 2003].

Many other works for the representation of music information in XML have been developed; among them we cite MusiXML [see www.music-notation.info/en/musixml/MusiXML.html] and Music Markup Language [MML; Steyn 2002]. They contain good ideas in the representation of and interaction between notational and performance information.

Other projects having similar data sources include the VARIATIONS project [Dunn and Mayer 1999] at Indiana University and the Online Music Recognition and Searching (OMRAS) project [see www.omras.org]. OMRAS addresses the need to represent symbolic information in the context of interactive musical libraries. The VARIATIONS project focuses on interaction among score images, audio recordings, and notational representation.

**XML Benefits**

The representation of musical information by means of XML provides some formalization facility. As stated by Huron [1992], there are at least twelve design principles of good representation, and Roland [2002] notes that XML guarantees the representations that are unique, mnemonic, non-cryptic, structurally isomorphic, explicit, optional, and extendable.

As demonstrated by MusicXML [Good 2002], representation of music information with XML can be a successful approach to data standardization and interchange. In this case, the adoption of MusicXML by commercial software has taken place very quickly. This has led it to be the first XML music format widely used in the music notation software market.

In the XML world, there are two ways to define an XML instance [i.e., the structure of the documents]. The first is by means of a grammar-like language called Document Type Definition (DTD) inherited from SGML [Standard Generalized Markup Language]. The second is the newest and richer XMLSchema, which declares the structure of the documents by means of another XML document. We have decided to encode the format with DTD technology instead of the XMLSchema, because DTD leads to some benefits noted by Roland [2002]. First, DTD technology is stable and widely adopted by software implementations, whereas XMLSchema is not. Another important feature of DTDs is the possibility of including external entities. This feature brings flexibility and extensibility to the definition of the format, since a user can modify the declaration of an included entity without affecting the entire document structure.

A problem we encountered in the definition of the DTD regards the inclusion of other XML stan-
Haus and Longari

Standards defined with XMLSchema technology. It is not possible to directly include an XMLSchema definition in a DTD or vice versa.

Layered SMI

Before proceeding, we want to define some terms frequently used in this article. First, we define **music content information** as basic music note properties such as duration, pitch, timbre, loudness, and their sequential or concurrent organization (e.g., staves). **Symbolic music information** (SMI) is defined to include general, structural, and music content information. Next, we use the term **source material** to denote graphic score images, notation files, digitized audio and video recordings, and performance files. Finally, we define a **music work** as the set of source materials and related symbolic representations.

Because XML organizes information in a hierarchical structure, each layer is represented in a subtree of the root element. The conceptual structure of layers is depicted in Figure 1, which should be read in this way: the Audio layer is the layer closer to “what we hear” (i.e., the physical aspects of music). Going up one level (to the Performance layer), we make an abstraction step. This way, we can think about the General layer as the more abstract; in fact, it is composed only by meta-data.

The root element of MX (<mx> element) shown below contains all the layers, each of which will be discussed shortly.

```xml
<!ELEMENT mx (general, structural?,
              music_logic, notational?,
              performance?, audio?)>
```

Note that elements that have a question mark appended are not mandatory. This means that an MX instance could be constituted only by the General and Music Logic layers; in fact they are the essential layers of the format.

General

Information regarding the music work as a whole should be stored in this layer. It gives a general description of the music work and groups information about all related instances.

The General information layer is defined this way:

```xml
<!ELEMENT general (description, casting?,
                   related_files?, analog_media?,
                   notes?, rights?)>
<!ELEMENT description (work_title?,
                       work_number?, movement_title,
                       movement_number?, genre?, author+)>
<!ELEMENT genre (#PCDATA)>
<!ATTLIST author type CDATA #IMPLIED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT work_title (#PCDATA)>
<!ELEMENT work_number (#PCDATA)>
<!ELEMENT movement_title (#PCDATA)>
<!ELEMENT movement_number (#PCDATA)>
<!ELEMENT casting EMPTY>
<!ELEMENT related_files EMPTY>
<!ELEMENT analog_media EMPTY>
<!ELEMENT notes EMPTY>
```

This layer contains the general description of the piece. The casting element, for example, represents information about actors or singers in an opera. Other general information includes the table of related music data files, referring to all layers, with one or more files for the summarization of each layer; the table of related multimedia data

Figure 1. MX layers.
files, such as images, videos, and the like; and the table of related analog media and technical information about related import/export/restoring/cataloguing/other operations.

The rights element is used to specify intellectual rights as well as security information about the whole piece. Rights and security information about source material must be specified for each source file. Thus, each sub-tree associated with source material [i.e., Audio, Performance, and Notational] contains another rights element.

**Structural**

The Structural information layer contains the explicit description of music objects and their causal relationships from both compositional and musical points of view [i.e., how music objects can be described as transformations of previously described musical objects]. Thus, it is neither a timed nor a physically ordered description, but it is a causal, ordered one. The information within this layer does not contain explicit descriptions of time ordering and absolute time instances of music events. Rather, it describes causal relationships among music object transformations and positioning within the music score as they happen in the frame of compositional/de-compositional [i.e., synthesis/analysis] processes.

There is no currently available standard that could be considered for this layer, but we can draw from three in particular. The first, SMDL [Sloan 1993; see also ftp.orl.gov/pub/sgml/WG8/SMDL/10743.ps], considers the topic without giving any concrete tool for describing it. The second standard includes high-level music programming languages and formal tools for modeling [e.g., Haus and Pighi 1996]. Finally, we can draw from a descriptive approach based on Music Petri Nets [Haus and Rodriguez 1993; Haus and Sametti 1994, 1995; De Matteis and Haus 1996], because there is an ongoing effort at the Laboratorio di Informatica Musicale [LIM] to improve ScoreSynth [Haus and Sametti 1991], a software tool for symbolic music processing, to meet the needs of the IEEE MX project.

This new approach is based on the PNML [Petri Net Markup Language, the Petri Net XML dialect; Jüngel et al. 2000] for representing musical Petri Nets as the structural level of music encoding [Baraté 2004].

A Petri Net is usually defined by a quadruple \(<P, T, A, M>\), where \(P\) is the set of places, \(T\) is the set of transitions, \(A\) is the set of arcs connecting both places to transitions and transitions to places, and \(M\) is the marking of the net. Music Petri Nets extend the “classic” definition of Carl Adam Petri so that the musician can build hierarchical music models defining music objects [MOs] and music algorithms [MAs], i.e., MOs processing. In this way, a Music Petri Net is defined by means of the triplet \(<P, T, A>\) and the following three statements:

\[
P ' p <\text{identifier}, \text{tokens}, \text{capacity}, \text{MIDI channel}, \text{object}, \text{play}, \text{file} >
\]

\[
T ' t <\text{identifier}, \text{algorithm} >
\]

\[
A ' a <\text{node-from}, \text{node-to}, \text{multiplicity} >
\]

Here, \(P\) is the set of places; \(T\) is the set of transitions; \(A\) is the set of arcs; \text{identifier} is the label of the node [both place and transition]; \text{tokens} is the number of tokens within the place, i.e., the value of the marking function \(M\) [see below]; \text{capacity} is the maximum number of tokens allowed for that place; \text{MIDI channel} is the number of the MIDI channel where the MO should be played; \text{object} defines whether an MO can be associated to the place or not; \text{play} defines whether the object should be played or not; \text{file} defines whether the MO associated with the place, if one exists, is stored on file or not; \text{algorithm} defines whether the MA associated to the transition, if one exists, should to be executed or not; \text{node-from} is the identifier of the node that starts the arc; \text{node-to} is the identifier of the node that ends the arc; and finally, \text{multiplicity} is the numeric label of the arc. If a place has an associated MO that is stored on file, its identifier represents both the label of the node and the file identifier. Furthermore, \(P, T,\) and \(A\) are finite sets, and \(P\) cannot be empty.

The marking of a net is the distribution of tokens into the place nodes; their number and distribution can change during net execution owing to transi-
tions firings. A marking function is defined as the following array:

\[ M(\langle m_1, m_2, \ldots, m_i, \ldots, m_n\rangle) \]

where \( n \) is the number of places within the net, \( N \) is the set of positive integers, and every \( N \cdot m_i = M(p_i) \) with \( P \cdot p_i \).

Then, the hierarchy of the model is described by the set

\[ S \subseteq \langle \text{identifier, node-begin, node-end} \rangle \]

which is the set of hierarchy links, where \( \text{identifier} \) is the label of the net, \( \text{node-begin} \) is the identifier of the beginning node for the morphism application, and \( \text{node-end} \) is the identifier of the ending node for the morphism application.

The \( S \) set represents all the information about the hierarchy of the model, whereas the \( P, T, \) and \( A \) sets completely describe a particular net of the model. In other words, a model is fully defined by an \( S \) set and as many triplets \( \langle P, T, A \rangle \) as the number of the nets within the model.

Part of the DTD describing the Structural layer follows:

```xml
<!ELEMENT structural (analysis*, MPN*)>
<!ELEMENT analysis (theme*, segment*, transformation*, relationship*)>

<!ELEMENT MPN (reference*)>
<!ELEMENT reference EMPTY>

<!ATTLIST reference filename CDATA #REQUIRED>
```

The symbol * after a declaration of an element in the content of another element means that the contained element can be repeated zero or more times. Thus, the element structural could be empty or composed by an arbitrary number of analysis elements followed by an arbitrary number of MPN elements.

The reference element in the content of MPN has an attribute indicating the filename that contains the description of the Musical Petri Net in PNML language.

### Notational

In this layer, we group all possible visual instances of a piece of music. We have previously characterized two modalities in which the music can be written or read: notational and graphical. A notational instance is often in a binary format, such as NIFF or Engima, that represents symbolic information. Other interesting instances are representation of music information by means of XML: MusicXML (Good 2000), Music Encoding Initiative (MEI; Roland 2002), and Music Markup Language (Steyn 2002). There are several other formats for the representation of notational information, however. We refer the interested reader to Selfridge-Field (1997).

A graphical instance contains images that represent the score. It is usually in a binary format, for example, a JPEG image or a PDF file, but it can also be a vector image encoded in Support Vector Graphics (SVG; see the “Scalable Vector Graphics (SVG) 1.0 Specification,” available online at www.w3.org/TR/SVG)—and thus in a text format that has no explicit musical meaning. Information contained in this layer is tied to the spatial part of the spine structure, allowing any point in the image to be mapped to a corresponding point in time. Consider the following definition:

```xml
<!ELEMENT notational (notation_instance | graphic_instance)>  
<!ELEMENT notation_instance (desc?, part_ref+, rights)>  
<!ATTLIST notation_instance  
  file_name CDATA #REQUIRED  
  format CDATA #REQUIRED  
  spine_start_ref IDREF #REQUIRED  
  spine_end_ref IDREF #REQUIRED>  

<!ELEMENT graphic_instance (desc?,  
  part_ref+, rights)>  
<!ATTLIST graphic_instance  
  file_name CDATA #REQUIRED  
  format CDATA #REQUIRED  
  spine_start_ref IDREF #REQUIRED  
  spine_end_ref IDREF #REQUIRED>  
```

The notational element must be composed by at least one element between notational_
instance or a graphic_instance. In fact, the symbol | indicates an alternative between the elements, and the symbol + indicates the repetition on one or more times. An attribute declared as #REQUIRED must be present in the instance of the element, but if it has a default value, it can be omitted.

Performance

The Performance layer lies in the middle between the Notational and Audio layers. It encodes parameters of the notes to be played (like the Notational layer) and parameters of the sounds to be created (like the Audio layer).

The Performance layer embraces symbolic formats such as MIDI files, Csound, or SASL/SAOOL (see sound.media.mit.edu/mpeg4/audio/documents; Scheirer and Vercoe 1999). This information is encoded for the purpose of synthetic music rendering. Timing information is coded in relative time units.

Performance layer descriptors do not encode information about each note or event but only general information about the file and its relation to the SMI. For example, the file can address only one instrument, or it may span only a part of the whole piece. Consider the following definition:

```xml
<!ELEMENT performance (performance_instance+)>
<!ELEMENT performance_instance (desc?, (MIDI | CSOUND | MPEG4), rights)>
<!ATTLIST performance_instance
  file_name CDATA #REQUIRED
  spine_start_ref IDREF #REQUIRED
  spine_end_ref IDREF #REQUIRED>
<!ELEMENT MIDI (MIDI_part_ref+)>
<!ELEMENT MIDI_part_ref EMPTY>
<!ELEMENT CSOUND (part_ref)>  
<!ELEMENT MPEG4 (part_ref)>  

When attributes declared as #IMPLIED have no value specified and derive their values from the context (e.g., if spine_start_ref and spine_end_ref are omitted), it is assumed that they represent the beginning and ending of the piece, respectively. The attribute format can assume only one of the three declared values.

Audio

The Audio layer describes properties of the source material, which contains musical audio information, and it is the lowest level of the format. Formats representing audio information can be subdivided into two categories: compressed and uncompressed. Uncompressed audio can be PCM/WAV (Pulse Code Modulation in Microsoft WAV file format), Audio Interchange File Format (AIFF), and μ-Law. Compressed audio can be further subdivided into lossy (e.g., MPEG and Dolby AC3) and lossless (e.g., ADPCM, SHN) types. To automatically relate the audio to the time part of the spine structure, it is necessary to extract features indicating the actual temporization of musical events. This information is independent from the format in which the audio information is stored, because compressed formats are uncompressed before elaboration. It is described by means of the <index> element:

```xml
<!ELEMENT audio (clip+)>
<!ELEMENT clip (desc?, part_ref+, performers?, rights, COMMON_HEADER_INFO?, index+)>
<!ATTLIST clip
  file_name CDATA #REQUIRED
  format CDATA #REQUIRED
  duration CDATA #REQUIRED
  encoding CDATA #REQUIRED
  freq CDATA #REQUIRED
  n_bit CDATA #REQUIRED
  n_channel CDATA #REQUIRED
  bitrate CDATA #IMPLIED
  spine_start_ref IDREF #REQUIRED
  spine_end_ref IDREF #REQUIRED>
```

Computer Music Journal
The COMMON_HEADER_INFO element refers to a specific format representation (e.g., MP3) and is beyond the scope of this article.

Music Logic Layer

The core of MX, and the concept on which we focus our attention next, is the Music Logic layer. As we stated before, each of the aforementioned layers is not complete enough to represent logical aspects of music. The Music Logic layer contains information common to all other musical aspects and other musical information that the composer intended to include in the piece. It is composed of two mandatory elements—the spine and the Logical Organized Symbols (LOS)—and an optional container for specifications for the presentation of symbolic information:

```
<!ELEMENT music_logic (spine, los, layout?)>
```

The spine element is a structure that relates time and spatial information. With such a structure, it is possible to move from a point in a notational instance to the relative point in a performance or audio instance. Logical organized symbols are the common ground for the music content. Music symbols are represented in XML syntax, making music content explicit to applications and users. Below, we explain the concept of each layer in detail.

For the sake of clarity, we provide a simple but complete example of a piece. Figure 2 illustrates instances of the [a] Structural, [b] Notational, [c] Performance, and [d] Audio layers. Figure 3 shows the complete XML file.

Suppose we have only a MIDI file representation of this piece. A notation program that receives this MIDI file as input will interpret the first `<Note On, Note Off> couple as a dotted sixteenth note, because it spans the time of a sixteenth note plus the time of a thirty-second note. There is no way to express in MIDI syntax the fact that in the notation this event is a staccato quarter. Also, there is no way the legato symbol can be expressed using MIDI. This information can be extracted from the audio only if we have a timbre model of the legato execution behavior of this instrument.

The XML encoding, however, enables the integration of the peculiarities of the three source layers without the implementation of interpretative and timbre models. Later, when these models become available, they can easily be incorporated into the XML encoding. We will discuss timbre and interpretative models in the Open Problem section.

Finally, Figure 2a shows a structural representation of the piece in Musical Petri Nets format (Haus and Sametti 1991; Baraté 2004). We can see that the piece is composed of two identical sequences of notes. The first is said to be a generative segment [or theme Th1] of the piece, and the second is a repetition of the first segment translated by zero semitones. For the sake of brevity, we do not explain further details of the description, but it is possible to prove that the execution of the Musical Petri Net leads to the reconstruction of the sequences of notes of Figure 2b. As we said before, the PNML description of the Petri Net is contained in a separate file linked by the MPN element, and it is not detailed here owing to space limitations.

Spine

The spine element is the logical structure that implements the integration of Notational, Performance, and Audio layers with the Music Logic layer. Its purpose is to build an abstract structure to which all the layers that describe the properties of the source material and the LOS can refer. This structure is necessary because properties encoded in the source material layers are relative to the format they are describing, and we need a unique reference point for all the instances in the same or different layers. For example, in the Audio layer, we have a property describing the exact temporization (expressed in
hundredths of a second) of a note or group of notes. If we have two different recordings of the same piece, we will have two different timings for the same group of notes. Each of the timings belonging to different audio instances will refer to the same event in the spine. Moreover, in the Notational layer, we can have two different representations of the same score, i.e., a NIFF file and a group of Tagged Image File Format (TIFF) images. We have the same problem as in the Audio layer: two different reference points in two different instances refer to the same logical event. The difference between the event in the Audio layer and the event in Notational layer is that the former is a temporal event, and the latter is a spatial event. However, they are logically the same event. The spine is the solution to this problem.

Going into deeper detail, the spine is a space–time structure composed of uniquely identified events. Each event has two coordinates relative to the preceding event in its dimension, because some symbols can have meaning only in the space dimension or in the time dimension. The first coordinate is the temporal coordinate of the event. It describes how much time must pass after the actualization of the

---

Figure 2. Examples of source material instances: (a) Structural; (b) Notational; (c) Performance; (d) Audio.
Figure 3. Example of a simple music work.
preceding event. This time is expressed in virtual time units (VTU, as described in the SMDL draft; see ftp.ornl.gov/pub/sgml/WG8/SMDL/10743.ps).

The second coordinate is expressed in a relative spatial unit that we call a “virtual logic position unit” (VLPU). Its purpose is to represent the reference point for the vertical alignment of different symbols in different parts or staves [see Figure 4].

The spine contains all the events of the music work. It can be seen as the vertical collapse of all the graphical and aural (performance and audio) events. Because repetitions are not often fully expanded in music notation, creating a discrepancy between the visual and aural domains, we have chosen to explicitly encode all the events in the aural domain.

The spine must be seen as an abstract structure to...
Figure 4. A spine visualization of space–time relationships.
which all the source material can be linked. The co-
ordinates are expressed in relative measurement
units, allowing any of the sources to be projected
onto the spine. In Figure 3, in the Audio layer, index
elements represent note start times in the audio
file. These start times are expressed in hundredths
of a second and are individually related to events in
the spine.

Each event is accessible from any layer and can
be employed for purposes of content retrieval and
browsing of the music work. This is an essential
task when we have to deal with a large amount of
source material relative to different music works.
Such an environment is feasible if we think about
the broad diffusion of music over the Internet or
about the rapidly growing need for digital libraries.
For an overview of problems related to managing a
vast amount of digital music information and music
information retrieval, see Downie (2003).

The integration of all musical aspects also en-
ables the improvement of human-machine interac-
tion applications, expanding the possibilities in the
areas of digital music composition and realization.

The unit measure granularity of the spine depends
on the smallest unit granule of information described
within each music layer. Using the smallest value
allows access to information related to individual
music events, such as notes, beginnings of themes,
instrumentation changes, etc. This feature allows
musicians to efficiently interact together with mu-
sic information. For example, building applications
that use this feature for a synchronized rendering of
audio and related score visualization becomes a
matter of software implementation.

Logical Organized Symbols

This sub-layer exists to aid the separation of con-
tent from presentation. The encoding of symbolic
information in XML syntax is important, because it
is the common ground upon which semantic elabo-
ration of music content can be created. In fact, it
could be the starting point for the synthesis of the
other layers. It is not important in which of the sev-
eral XML languages for music notation it is ex-
pressed. It is necessary, however, that the elements
explicitly identify concepts that can be tied to the
spine structure. For example, we think that parts
are fundamental elements. This is because in most
of the formats in which source material is encoded,
it is possible to distinguish between instrumental
parts. Moreover, in symbolic notation, a part can
span over different staves or, conversely, different
parts can appear on a single staff. Hence, the part
can be seen as a logic unit of the logical view.

Within parts must be encoded the basic music in-
formation, i.e., pitch, duration, and intensity. But
the richer the language, the better we can model a
music work.

The example in Figure 3 is encoded in the proto-
type language we are developing for the MAX Proj-
et (see www.lim.dico.unimi.it/IEEE/XML.html)
and is well detailed in Haus and Longari (2002) and

Layout

Music information encoded in the way described
above allows flexibility of visual rendering. In fact, it
is easy to define elements and properties that, based
on the information contained in the spine and LOS,
organize music symbols in various media, from differ-
ent paper layouts to different screen sizes. This is
because they can relate indirectly to Logical Music
Symbols through the spine structure. Particular ren-
ditions of symbols and non-conventional signs can be
easily modeled with Support Vector Graphics (SVG).

Layer Dependencies

As we can see from Figure 5, the central element of
our view of SMI is the Music Logic layer. All other
layers relate to this core element in a star-like struc-
ture. Only the General layer does not link explicitly
to the Logic layer. Relationships among layers are
implemented by means of XML references. Audio
source material is directly linked to the spine and
thus indirectly to the LOS. Notational and perfor-
ance source material can also be directly linked to
elements in the LOS, because they represent analog-
ous information. The Structural layer represents
music objects and relations among them, and because they are concepts within our SMI, their descriptions rely on the LOS. This structure allows the navigation from one layer to another. For example, suppose that we are listening to a piece of music on a CD and we have the MX description of the piece and a file containing the score of the piece. We can imagine software that reads the current player position and renders the exact page of the score that we are currently hearing, highlighting the performed notes. Conversely, if we are looking at a score and we have a particular audio recording, we could start listening to the piece from any given note.

Open Problems

Every approach to the representation of music information that tries to integrate symbolic, performance, and audio levels must provide two entities: timbral models and interpretative models, both direct and inverse (i.e., from the higher to the lower level of representation or vice versa). Currently known approaches are not able to satisfy these needs completely.

The timbre problem concerns the automatic extraction of timbre features from complex audio signal (polyphonic and multi-timbral audio). Timbre modeling could be improved by means of automatic synthesis of timbre models and their parameters. It is possible to obtain MPEG SASL codes (a score-like representation) extracting features from audio signals, but there is no available coder that can produce MPEG4 SAOL audio codes (i.e., the description of synthesis algorithms; Scheirer and Vercoe 1999).

On the other hand, there is the interpretative problem: how can we translate NIFF codes (i.e., scores) into MIDI codes (i.e., performances) and vice versa? There is a need for interpretative models. This functional unit could render timed, instantiated music information, coded in MIDI format, from, say, a NIFF score. Conversely, we might eventually be able to transcribe music scores from audio signals with a high degree of accuracy.

If we can get beyond the lack of these two models somehow, it becomes possible to process music information among all music layers—analysis, transformation, and synthesis of music information—within an integrated framework.

Versioning

Customization of musical works is a common task in music. Modified versions of musical scores have important value in themselves. Keeping track of these changes can be very challenging, however. The most common are changes in music symbols, such as articulations or embellishments, and translations of text performance indications. In our framework, this can be implemented simultaneously from the point of view of contents and graphical representation. In particular, custom musical signs can be modeled with SVG syntax.
Relationship to Other XML Standards

The past few years have seen XML applications arise for standardization in many problem domains. However, reusing already existing formalizations in a new XML application is much more useful. This ability of XML helps us merge our work with already approved standards that tackle a specific problem.

One of these is the rights and ownership protection problem. In the context of our work, we distinguish between security elements encoded in the source material and security elements expressed in the symbolic domain. Security in the source material is beyond the scope of this project. However, we do address the problem of security in the symbolic domain. Here, we must further discriminate among security protocols in XML, including authentication “Security Assertion Markup Language” or SAML (see www.oasis-open.org/committees/security); access control “cXtensible Access Control Markup Language” or XACML (see www.oasis-open.org/committees/xacml); digital rights management “cXtensible rights Markup Language” or XRML (see www.xrml.org/reference/XrMLTechnicalOverviewV1.pdf); and security for XML, including XML Signature and XML Encryption (see www.w3.org/Signature and www.w3.org/Encryption/2001). Security in XML is useful for the establishment of computing environments and of music services based on XML. Security for XML is needed for the protection and ownership validation of music works.

One other approved standard that is very useful is the vector graphics XML standard, namely Support Vector Graphics (SVG). It can be employed for the representation of musical fonts, custom symbols, and graphical score correction. Newly defined symbols can be easily linked to LOS symbols and to the spine structure.

We have not directly related MX to the well-known XML standard for synchronization Synchronized Multimedia Integration Language (SMIL; see www.w3.org/TR/smil20), because we think that music is a very complex case of multimedia information. Music has many “corner cases” that require a specialized representation and formalization. We think that adapting a standard designed for a different topic cannot be an effective solution.

Conclusion

We have presented our work on the development of a multi-layered, time-based environment for the representation of musical information, which we call MX. We have depicted the main concepts of the MX framework upon which many other systems and applications can be implemented. Our view is strongly oriented to the symbolic domain of music representation but with an innovative interpretation of the role of this domain. The Symbolic layer becomes the central layer in which all other layers have a reference point for their own content, both from the structural and space-time points of view.

Our plan for future work mainly addresses the integration of all available, meaningful experiences of other research groups working on the topic of XML and music. We are also interested in developing innovative topics of interest, starting from the open problems outlined in this article, and in an open approach that could easily integrate both timbre and interpretative models as they become available. Any cooperation is welcome.

Acknowledgments

The authors wish to acknowledge the partial support of this project by Italian MIUR [FIRB “Web-Minds” project N. RBNE01WEJT_005] and the Italian National Research Council, in the framework of the research program “Methodologies, Techniques, and Computer Tools for the Preservation, the Structural Organization, and the Intelligent Query of Musical Audio Archives Stored on Heterogeneous Magnetic Media,” under the Finalized Project “Cultural Heritage,” Subproject 3, Topic 3.2, Subtopic 3.2.2, Target 3.2.1. We also want to acknowledge the members of the IEEE MX Working Group [PAR1599] for their cooperation and interest in our work. We also wish to acknowledge the members of the MAX working group for their cooperation and interest in our work.

We want to give special thanks to Perry Roland for his help in improving the structure and English grammar of the article and for his precious suggestions.

This work has been made possible by the efforts...
of researchers and graduate students of LIM. We wish to especially thank L. Ludovico for his work on refining MX.

References


