Audio Engineering Society

Convention Paper

Presented at the 114th Convention
2003 March 22–25 Amsterdam, The Netherlands

Using and enhancing the current MPEG-7 standard for a music content processing tool

Emilia Gómez, Fabien Gouyon, Perfecto Herrera and Xavier Amatriain

Music Technology Group, Universitat Pompeu Fabra
Passeig de Circumval·lació, 8, 08003 Barcelona, Spain
http://www.iua.upf.es/mtg
{emilia.gomez,fabien.gouyon,perfecto.herrera,xavier,amatriain}@iua.upf.es

ABSTRACT

The aim of this document is to discuss possible ways of describing some music constructs in a dual context. First, that of the current standard for multimedia content description: MPEG-7. Second, that of a specific software application, the *Sound Palette* (a tool for content-based management, content edition and transformation of simple audio phrases). We discuss some MPEG-7 limitations regarding different musical layers: melodic (present but underdeveloped), rhythmic (practically absent) and instrumental (present though using an exclusive procedure). Some proposals for overcoming them are presented in the context of our application.

1. INTRODUCTION

Describing the musical content of audio files has been a pervasive goal in the computer music and music processing research communities. Though it has been frequently equated to the problem of transcription, describing musical content usually implies an applied context that has a non-scholar or home user in the final end of the chain. Therefore, it is usually the case that conventional music representations are not the most relevant for storing content descriptions that are going to be managed by people with different backgrounds and interests (probably quite different from the purely musicological ones). This approach to music content description has been that of the standardization initiative carried out since 1998 by the ISO workforce known as MPEG-7. The Moving Picture Experts Group (MPEG)[1] is a working group of the International Standard Organization/International Electrotechnical Committee (ISO/IEC) in charge of developing standards for coded representation of digital audio and video. Established in 1988, the

[1] http://mpeg.telecomitalialab.com/

group has produced MPEG-1, MPEG-2, MPEG-4, MPEG-7, and now is working on the new standard MPEG-21 "Multimedia Framework" since June 2000. More details on the standards can be found in [1].

MPEG-7 is formally named “Multimedia Content Description Interface”, and it provides multimedia content description utilities for browsing and retrieval of audio and visual content. It was officially approved in 2001 and is currently being further expanded into its second version.

This standard provides normative elements, as Descriptors (henceforth Ds), Description Schemes (henceforth DSs) and a Description Definition Language (henceforth DDL) [18]. The Descriptors define the syntax and the semantic of the represented features. The Description Schemes are used to group several Ds and DSs into structured, semantic units. The Description Schemes are specified using the DDL.

The *Audio* part of the standard [25] relies on two basic structures: the *segment*, inherited from the *Multimedia Description Scheme*, that allows to define a temporal structure of the audio signal, and the *scalable serie*s, a type that is inherited by all the *low-level descriptors* (henceforth LLDs). It then distinguishes two classes of structures, the generic audio description framework and the application-related tools. The first one is defined as the basic compatibility layer upon which generic descriptions and unique applications may be built for any signal, and it includes *low-level* descriptors, the *scalable series* scheme, and the *silence segmen*t. The second one includes sound recognition, instrumental timbre description, spoken content description and melody description tools, as well as tools for audio matching.

The *Sound Palette* application, developed in the framework of the European project CUIDADO, is also committed with applied music description. It is intended to be a an authoring tool for retrieving, editing, transforming and mixing isolated sound samples and phrases of reduced complexity (monophonic musical phrases, rhythm loops). The development of this application calls for a structured set of description schemes covering from signal-related descriptors to user-centered descriptors.

Given our previous experience and involvement in the MPEG-7 definition process [21, 26], we have developed a set of music descriptors and description schemes according to the MPEG-7 DDL. Our goals are manifold. First, coping with the description needs of a pecific application. Second, keep coherence and compatibility with the current MPEG-7 standard. Third, evaluating the feasibility of new description schemes for being considered as possible enhancements to the current standard. In the following, we first briefly review different ways of describing musical content that have been proposed in the literature. We address melodic, rhythmic and instrument characterizations. We report on available Ds and DSs as defined in the MPEG-7 standard: those related to the melody (present but underdeveloped), those related to rhythm (practically absent) and those regarding instruments (present though using an exclusive procedure). Complex music description layers, as it is the case of harmony description, are purposely left out from the discussion.

2. REVIEW

2.1. Melodic Description

Literature is rich in descriptors related to melodic aspects of sound. There has been a big contribution in the context of *Query by Humming* systems, where it is necessary to represent data in a efficient manner so that it can be matched against a database of melodies. However, this is only one of the possible application contexts of melodic description.

The central low-level melodic attribute is the fundamental frequency, which is usually computed using the information of an analysis audio **Frame**.

In the literature, melody has been mainly described using pitch information (a perceptual descriptor, in opposition to the fundamental frequency signal descriptor) of **Note** segments. As a forward step from pitch, pitch contour (or interval information between consecutive notes) has also been used in several applications as query by humming, similarity matching or melodic classification, because it has been found to be more significant to listeners in determining melodic similarity. The earliest approaches disregarded timing information completely, but more recent studies showed that durational values facilitate melody recognition. Other application contexts (as, for example, performance analysis or analysis/synthesis) need a more accurate description, in order to characterize some aspects related to expressivity. Features related to vibrato (rate and amplitude), articulation and dynamics become then significant. Some of these features are already included in the MIDI representation.

Other attributes than a succession of note features have been also taken into account for melodic description of large audio segments. These features are not associated to a Note segment, but to an audio phrase or a larger audio segment. These global descriptors have been used in different application contexts, as melodic classification, comparative analysis and melodic retrieval. Some of them are: key, tessitura, type of intervals, pitch histogram features [31], melodic density and melodic profile. We refer to [13] for a review on melodic descriptors.

Information related to the MPEG-7 Melody Description Scheme (henceforth Melody DS) can be found in [2] and [1], and is explained in [25]. The MPEG-7 standard proposes a fundamental frequency Low-Level Descriptor and a *Melody* DS that includes either melody as a sequence of intervals plus note relative duration, or as a succession of contour and beat values, in addition to some information about scale, meter and key (see Figure 1), which are related to the whole *AudioSegment* to which the Melody DS is attached.

In the MPEG-7 scheme, the melodic contour uses a 5-step contour (from -2 to +2) in which intervals are quantized. Also, it represents basic rhythm information by quantizing the duration of every note at the beat level (i.e. attributing to each note the temporal value of the beat to which it pertains). This can drastically increase the accuracy of matches to a query; however, it is a coarse quantization that would not permit to represent score durations for instance. The lack of other rhythmic descriptors is discussed in a following section. This contour has been found to be inadequate for some applications, as melodies of very different nature can be represented by similar contours. One example is the case of having a descendant chromatic melody and a descendant diatonic one. Both of them have the same contour although their melodic features are very dissimilar.

For applications requiring greater descriptive precision or reconstruction of a given sequence of notes, the Melody DS supports an expanded descriptor set and higher precision of interval encoding. Rather than quantizing to one of five levels, the precise pitch interval (with cent or greater precision) between notes is kept. More accurate timing information is kept by encoding the relative duration of notes defined as the logarithm of the ratio between the differential onsets, following the formula below:

$$Noterelduration[n] = \begin{cases} \log_2\left(\dfrac{Onset[n+1]-Onset[n]}{Onset[n]-Onset[n-1]}\right) & for\ n \geq 2 \\ \log_2\left(\dfrac{Onset[2]-Onset[1]}{0.5}\right) & for\ n = 1 \end{cases}$$

As seen before, arranged around these core descriptors there are a series of optional support descriptors such as lyrics, key, meter, and starting note, to be used as desired for an application.

This expanded description does not take into account silence parts that sometimes play an essential role for melodic perception. Except for phoneme information (*mpeg7:PhoneNGram* element, that has been deleted from the first description scheme), this melodic description (contained into the Melody DS) could be completely extracted from the score, and no information related to the signal level would be necessary. Some examples of melodic description can be found in the MPEG schema specification documents [1,2] and in [12].

As explained, the standard distinguishes three types of attributes:

- Low-level descriptors associated to a time point: *mpeg7:AudioFundamentalFrequencyType*, *mpeg7:AudioPowerType*.
- Features attached to a Note segment (pith note, note relative duration, interval and contour)
- Melodic descriptors related to the *AudioSegment*: key, scale, meter and melody sequence (or sequence of notes).

Regarding the *mpeg7:Note* representation, some important features like e.g. intensity, intra-note segments, articulation or vibrato are lacking; they nevertheless would be necessary for, for example, performance analysis, expressivity characterization and expressive synthesis (it should be noted that some of these features are already coded by the MIDI representation). This *Note* type, in the *Melody* DS, includes only note relative duration information, silences are not taken into account. Nevertheless, it would sometimes be necessary to know the exact note boundaries and other timing descriptors, as discussed in the following paragraphs. Also, the Note is always defined as a part of a descriptor scheme (the *noteArray*) in a context of a *Melody*. One could object that it could be defined as a segment, which, in turn, would have its own descriptors.

Regarding the *mpeg7:key* and *scale* descriptor, the standard does not consider possible changes of tonality inside the *AudioSegment*. This is a general consideration for all the unary descriptors. These two melodic descriptors are also very oriented towards tonal music. Other scalar melodic descriptors could be added to the description scheme according to the type of sounds and application needs.

2.2. Rhythm Description

Considering the word 'rhythm' in a broad sense, one can wish to characterize the rhythm of a single note, of a pattern or of an entire musical movement, either monophonic or polyphonic. It intuitively makes sense to aim at characterizing events corresponding to different temporal scopes with respect to rhythmic attributes. Moreover, different rhythmic descriptors may be relevant in different types of applications, ranging e.g. from performance investigations to song comparisons.

The literature concerning automatic extraction of rhythmic features from audio or symbolic data is indeed wide (see e.g. [16] for a review), let us give a few pointers here. In the context of audio signal classifiers, [29] and [32] propose low-level rhythmic features characterizing in some way the rhythmic strength of the signal. The rate at which one taps his feet to the music (i.e. the Beat) is a more common rhythmic concept that has been addressed widely from both computational and perceptual points of view (see e.g. [28]). Other periodicities than the perceptually most important one (i.e. the Beat) have also been proposed. For instance, among others, [3], [30] and [15] focus on the concept of "tatum" (smaller rhythmic pulse). A wide literature discusses time quantization (or "rhythm-parsing") [9]. Some propose to consider jointly the beat-tracking and rhythm-parsing issues, i.e. the variation of a musical piece's pace and the quantization of note timings (see [6] and [27]). Going to a higher level of abstraction, some (although much less numerous than those focusing on the Beat) seek Meter and rhythmic pattern recognition (see e.g. [4], [10]). Finally, many address the notion of expressive timing (e.g. [7], p. 489), and focus on features like tempo changes, deviations, or event-shifts (see e.g. [3]). An insightful discussion regarding the intertwining of expressive timing, metrical structure and pace of a musical signal can be found in [22].

Some features are standardized since long ago. The MIDI "standard" provides a means of storing (but most importantly transmitting in real-time) the *Tempo* (the number of microseconds per quarter-note), the *Meter* (time signature), and the MIDI *Timing Clock*, all of which are dimensions relative to time and rhythm. The latter permits to represent time by a discrete temporal unit that depends on the notion of tempo (unlike the MIDI *Time Code* which is a representation of *absolute* time in that it follows hours, minutes and seconds and cannot be speeded up or slowed down.). This is a message –the status byte F8– sent from 24 to 480 times per quarter note. If the tempo changes, the MIDI Timing Clocks will pass at a faster rate, but the number of messages per quarter note will stay the same.

Current elements of the MPEG-7 standard that convey a rhythmic meaning are the following:

- The *Beat (mpeg7:BeatType)*
- The *Meter (mpeg7:MeterType)*
- The *note relative duration*

The *Beat* and *note relative duration* are embedded in the melody description (respectively in the melody contour and in the note, as we can see in Figure 1). The *Meter*, also illustrated in [1] in the description of a melody, might be used as a descriptor for any audio segment, however it would not be a relevant descriptor at *any* temporal scope of description.

Here, the *Beat* refers to the pulse indicated in the *Meter* denominator (which does not necessarily corresponds to the notion of perceptually most prominent pulse). The *BeatType* is a series of numbers representing to which beat pertains each note. That is, it represents the quantized positions of the notes, with respect to the first note of the excerpt, the positions are expressed as integers, multiples of a timing reference, the Beat (which actual value in seconds is not given). The *note relative duration* is the "logarithmic ratio of the differential onsets for the notes in the series" [1]. The *MeterType* carries in its denominator a reference value for the expression of the beat series. The numerator serves, in conjunction to the denominator, to refer somehow to pre-determined templates of weighting of the events. It is assumed that to a given meter corresponds a defined "strong-weak" structure for the events.

This representation of rhythm has been proved useful for improving query-by humming applications. But let us wonder what, beyond the scope of this application, could be its limitations.

First comes to mind the fact that the context of rhythm description is, in the current standard, that of a monophonic melody, which seems to be a restriction.

The time signature can be represented, but not the bar lines (*where* is the downbeat, the 'one' in 'one'-'two'-'three'-'one'-'two'-'three'-). Nor can be represented the symbolic values (e.g. 'quarter-notes', etc.) of the notes, indeed, as events are characterized by which beat they belong to, this is not accurate enough to represent already-quantized music where sub-multiples are commonly found (e.g. 'eighth-notes'). Therefore a score could not be represented by means of the current MPEG-7 standard.

The speed of execution of a musical piece (as in a performance, and sometimes suggested in a score) is absent from the standard. Also, the exact timing occurrence of each note cannot be represented. Therefore, there would be no possibility to encode a MIDI data stream for instance.

One may notice that a proposal of tempo descriptor has recently been made to the MPEG consortium: the *AudioTempo* [19]. It is not presently part of the standard, but is candidate for its second version. It is a scalar value assigned to an audio segment, the number of beats per minutes (BPMs). The *AudioTempo* can be useful as a global descriptor of a piece of music, to account for its global pace. However, the evolution of the tempo is also a very important rhythmic feature, representing it by means of the *AudioTempo* would entail a segmentation of the music piece at hand in many audio segments whose only reason of being would be their tempo differences. This is questionable. Moreover, the variation of the tempo is a continuous phenomenon, one may wish to envision it with different levels of accuracy, depending on the application. Considering the assumption of constant tempo being relevant to a given piece, the "phase" of the beat, i.e. the actual point in time where occurs the first beat (in a temporal window length proportional to the inverse of the tempo) is another important feature that is lacking in this proposal. It is indeed completely different to tap one's foot on the beat than against the beat, or slightly ahead or before of it. More important, improving the current standard by adding a single metrical level forgets the fundamental notion of *hierarchy* in the rhythmic structure of music: several metrical pulses coexist and are tightly related [24]. A representation scheme for rhythm should account for this property.

When dealing with expressive performance data (audio or MIDI), quantizing a note time occurrence through the use of the *BeatType*, a rounding towards $-\infty$ occurs; thus, in the case where an event is slightly before the beat (as it can happen in expressive performance) it is attributed to the preceding beat, which in some cases could be dramatic. Finally, provided the underlying metrical structure could be derived from a musical performance, the current representation cannot serve for exploring fine deviations from the structure, which would be useful in applications of e.g. performance comparisons.

A last practical point could be made: current elements of rhythm description in the MPEG-7 standard are extremely sensitive to the determination of the meter, which is still a difficult task for the state-of-the-art in rhythm computational models; and no algorithms are suggested for the determination of these features (even though this is not mandatory in the official MPEG references, informative extraction procedures are provided for many other descriptive features).

2.3. Instrument Description

One can address instrument descriptions at different levels of abstraction. At a high level, one may be interested in verbal descriptions of instruments and organized taxonomies (i.e. telling if a piece of music contains percussion, piano, or a singer, or even if there are tabla or congas, and male or female singers in the piece). At a lower level, the interest can be in descriptions of what in the signal is characteristic of a given instrument, i.e. its timbre (i.e. if the timbre of a given segment is similar to that of another segment from other file). In this section, we present ways to link these descriptions.

When dealing with multi-timbral polyphonies, or when working with sound samples, a need for instrument labeling of segments arises. In the first case, we assume the labeling to be done manually. Indeed, the current state of the art does not provide satisfactory tools for doing it automatically, though acceptable automatic approximations can be achieved when the complexity of the content is low (i.e. when dealing with "rhythm loops"). On the other hand, when working with sound samples (a situation that arises in the context of the abovementioned *Sound Palette*), automatic labeling of instrument classes can be done with high success rates. A convenient way for this consists in deriving a model that characterizes, in a compact way, the spectro-temporal distinctive features for a given instrument class. Models can be computed using quite different techniques (see [20]), and they can be stored for subsequent usage in a recently proposed –and accepted by the major data mining companies- standard that has been termed PMML[2] (Predictive Model Markup Language).

Therefore, in cases where it may be possible to define an explicit mapping between low-level descriptors of the sound (i.e. physical properties) and high-level ones (e.g. actual names of instruments), several types of Ds and DSs are needed:

- Representation schemes for verbal descriptions
- Representation schemes for physical descriptions
- Representation schemes for the description of classification models.

In the most general case (multi-timbral polyphonies), one might still need containers for manually entered verbal descriptions of the instruments, even if not related to physical properties of the sound. Indeed, there is no problem for a content provider to offer exhaustive taxonomies of sounds; it could also be possible for a user to define her/his own devised taxonomies.

The current MPEG-7 standard provides Ds and DSs for timbre as a perceptual phenomenon, useful in the context of search by similarity in sound samples databases. These Ds, grouped together with other so-called "Low Level Descriptors" are: *Harmonic Spectral Centroid, Harmonic Spectral Deviation, Harmonic Spectral Spread, Harmonic Spectral Variation, Spectral Centroid, Temporal Centroid*, and *Log Attack Time*. They assume that, given some

[2] see http://www.dmg.org for the definition documents of PMML.

special conditions, segments containing sounds can be allocated to generic “percussive” or “harmonic” classes, and after that identification, some specific weighting grants a retrieval by perceptual similarity (i.e. given a violin sound, the retrieved list can for sure contain other violin sounds, but also some viola and cello sounds, and maybe some clarinet sound, as the perceptual similarity criteria do not take into account taxonomic information).

Complementary to timbre descriptions, some Ds and DSs permit to address verbal descriptions, allowing to perform categorical queries in databases or to build taxonomies of instruments. The MPEG-7 *ClassificationScheme* defines a scheme for classifying a subject area with a set of terms organized into a hierarchy. A term in a classification scheme is referenced in a description with the *TermUse* data type. A term represents one well-defined concept in the domain covered by the classification scheme. A term has an *identifier* that uniquely identifies a term, a *name* that may be displayed to a user or used as a search term in a target database, and a *definition* that describes the meaning of the term. Terms can be put in relationship with a *TermRelation* descriptor. It represents a relation between two terms in a classification scheme, such as synonymy, preferred term, broader-narrower term, and related term. When terms are organized this way, they form a classification hierarchy. This way, not only content providers but also individual users can develop their own classification hierarchies. A generic classification scheme for instruments along the popular Hornbostel-Sachs-Galpin taxonomy (cited by [23]), would have the schematic expression depicted below. (More examples using the ClassificationSchemed DS can be found in [5].)

```
<ClassificationScheme term=“0”
scheme=“Horbonstel-Sachs Instrument
Taxonomy”>
<Label>”HSIT”</Label>
<ClassificationSchemeRef
scheme=“Cordophones” />
<ClassificationSchemeRef scheme=“Idiophones”
/>
<ClassificationSchemeRef
scheme=“Membranonphones”  />
<ClassificationSchemeRef scheme=“Aerophones”
/>
<ClassificationSchemeRef
scheme=“Electrophones” />
</ClassificationScheme>
<ClassificationScheme term=“1”
scheme=“Cordophones”>
<Label>”Cordophones”</Label>
<ClassificationSchemeRef scheme= “Bowed” />
<ClassificationSchemeRef scheme= “Plucked” />
<ClassificationSchemeRef scheme= “Struck” />
</ClassificationScheme>
<ClassificationScheme term=“2”
scheme=“Idiophones”>
<Label>”Idiophones”</Label>
<ClassificationSchemeRef scheme=“Struck” />
<ClassificationSchemeRef scheme=“Plucked” />
<ClassificationSchemeRef scheme=“Frictioned”
/>
<ClassificationSchemeRef scheme=“Shakened” />
</ClassificationScheme>
<ClassificationScheme term=“3” scheme=
“Membranophones”>
</ClassificationScheme>
…
```

Giving the user the option for defining instrument or sound taxonomies is an important feature, but populating a database with terms from them can be a tedious operation (as it has to be manually performed) unless it is provided a linking mechanism that uses some mathematical model for computing the name of the class, provided some low-level descriptors. This mechanism has been restricted in the current version of MPEG-7 to the timbral description of the audio through a continuous Hidden Markov Model using a low-dimensional representation of the spectrum, the so-called *mpeg7:SpectrumBasis*. This descriptor contains basis functions that are used to project spectrum descriptions into a low-dimensional and decorrelated representation. The basis functions are estimated through singular value decomposition [11] although other methods can be considered. As we have previously mentioned, this approach completely disregards other possibly valid options for sound modeling. We will return on that in a forthcoming section.

3. USING AND ENHANCING MPEG-7 FOR MUSIC DESCRIPTION

3.1. Audio Segment Derivation

In its current version, the MPEG-7 standard can be used for some applications (e.g. query-by-humming, timbral similarity search), however it still needs enhancements to cover a wider range of applications. In our application context, we encountered a problem in a too general definition of the *mpeg7:AudioSegment* for description. A way to deal with limitations identified in previous sections could be to define two types of segments derived from the *mpeg7:AudioSegment:* a *NoteSegment* (in the authors’ opinion, conceptually a note has to be considered as a segment, not a descriptor) and a *MusicSegment* (representing a monophonic or polyphonic excerpt, being possibly decomposed in *Note* or *Music* segments). It is our belief that defining specific temporal scopes of description (with different melodic, rhythmic and instrumental

descriptors) would open the way to more accurate descriptions of musical excerpts (in audio, MIDI or score formats).
Class diagram of MPEG-7 *AudioSegment* and *AudioSegmentTemporalDecomposition* derivations:

3.1.1. The NoteSegment

Segment representing a note. The note has an associated DS, the Note DS, accounting for melodic, rhythmic and instrument descriptors, as well as the low-level descriptors (LLDs) inherited from mpeg7:AudioSegmentType.

```
<complexType name="NoteSegmentType">
<complexContent>
<extension base="mpeg7:AudioSegmentType">
<sequence>
<element name="NoteDS" type="NoteDSType"
minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</extension>
</complexContent>
</complexType>
```

<u>**NoteSegmentTemporalDecomposition**</u>

```
<complexType
name="NoteSegmentTemporalDecompositionType">
<complexContent>
<extension
base="mpeg7:AudioSegmentTemporalDecomposition
Type">
 <element name="NoteSegment" t
type="NoteSegmentType" minOccurs="1"
maxOccurs="unbounded"/>
extension>
</complexContent>
</complexType>
```

3.1.2. The MusicSegment

Segment representing an audio excerpt, either monophonic or polyphonic. This segment has its associated DS, the Music DS, and can be decomposed in other *MusicSegments* (for example, a polyphonic segment could be decomposed in a collection of monophonic segments, as illustrated in Figure 2) and in *NoteSegments*, by means of two fields whose types derive from *mpeg7:AudioSegmentTemporalDecompositionType* (see Figure 3). The Music DS has associated rhythmic, melodic and instrument Ds that differ from those of the *Note DS*.

```
<complexType name="MusicSegmentType">
<complexContent>
<extension base="mpeg7:AudioSegmentType">
<sequence>
 <choice minOccurs="0" maxOccurs=unbounded">
 <element name="MusicSegmentDecomposition"
type="MusicSegmentTemporalDecompositionType"
minOccurs="0" maxOccurs="unbounded"/>
<element name="NoteSegmentDecomposition"
type="NoteSegmentTemporalDecompositionType"
minOccurs="0" maxOccurs="unbounded"/>
</choice>
<element name="MusicDS" type="MusicDSType"
minOccurs="0" maxOccurs="unbounded" />
</sequence>
</extension>
</complexContent>
</complexType>
```

<u>**MusicSegmentTemporalDecomposition**</u>:

```
<complexType
name="MusicSegmentTemporalDecompositionType">
<complexContent>
<extension
base="mpeg7:AudioSegmentTemporalDecomposition
Type">
<element name="MusicSegment"
type="MusicSegmentType" minOccurs="1"
maxOccurs="unbounded"/>
       </extension>
</complexContent>
</complexType>
```

3.2. Segment Classification Scheme

Another way to deal with different types of segments is by using a classification scheme (*mpeg7:ClassificationScheme)*. MPEG-7 provides ways to define classification schemes and hierarchies of segments. This option seems to be more flexible to different segmentation strategies and hierarchy schemes. This possibility is being considered in the current system, because it is more simple and it would not suppose any extension to the current standard.

<u>**Classification scheme definition:**</u>

```
<ClassificationScheme id="SegmentTaxonomy"
uri="urn:upf:cs:SegmentTaxonomy">
<Term termID="S1">
  <Name xml:lang="en"
preferred="false">MusicSegment</Name>
</Term>
<Term termID="S2">
  <Name xml:lang="en"
preferred="false">NoteSegment</Name>
</Term>
<Term termID="S3">
  <Name xml:lang="en"
preferred="false">Attack</Name>
</Term>
</ClassificationScheme>
```

```
<MultimediaContent xsi:type="AudioType">
 <Audio id="mySegment">
  <MediaTime>…</MediaTime>
    <StructuralUnit
href="urn:upf:cs:SegmentTaxonomy:S1">
     <Name>MusicSegment</Name>
    </StructuralUnit>
    <TemporalDecomposition criteria="temporal"
overlap="false" gap="true">
       <AudioSegment id="myNote1">
         <MediaTime>…</MediaTime>
         <StructuralUnit
href="urn:upf:cs:SegmentTaxonomy:S2">
         <Name>NoteSegment</Name>
         </StructuralUnit>
       </AudioSegment>
      <AudioSegment id="myNote2">
         <MediaTime>…</MediaTime>
      <StructuralUnit
href="urn:upf:cs:SegmentTaxonomy:S2">
```

```
        <Name>NoteSegment</Name>
      </StructuralUnit>
…
    </AudioSegment>
  </TemporalDecomposition>
</Audio>
</MultimediaContent>
```

3.3. DS Definition and Extensions

Just like *the mpeg7:AudioSegment* has (among other elements) an element of type *mpeg7:AudioDSType*, the *NoteSegment* has a *Note DS* and the *MusicSegment* has a *Music DS*.

3.3.1. NoteDS

A Note type is defined in the *mpeg7:Melody DS*. The Note DS (attached to a *NoteSegment*) enhances it, including features related to melody (note fundamental frequency and symbolic pitch) and rhythm (temporal location and symbolic duration), as well as features related to expressivity (vibrato, articulation –as attack, decay and release features, not included in this document– and timing deviations from perfect metrical structures) and dynamics (describing intensity).

The exact temporal location of a note is described by the *mpeg7:MediaTime* attribute inherited from the *mpeg7:AudioSegment*.

```
<complexType name="NoteDSType">
<complexContent>
<extension base=“mpeg7:AudioDSType”>
<sequence>
<element name="Frequency" type="float"
minOccurs="0"/>
<element name="Pitch" minOccurs="0"
type=”PitchType”/>
<element name="Intensity" type="float"
minOccurs="0"/>
<element name="Vibrato" minOccurs="0"
type=”VibratoType”/>
<element name=“QuantizedInstant”
type=“QuantizedInstantType”/>
</sequence>
</extension>
</complexContent>
</complexType>
```

Low-level descriptors are inherited from the *AudioSegment* type.

PitchType:

```
<complexType name=”PitchType”>
<complexContent>
<extension base=“mpeg7:AudioDSType”>
<sequence>
 <element name="PitchNote">
  <complexType>
   <simpleContent>
    <extension base="mpeg7:degreeNoteType">
      <attribute name="display" type="string"
        use="optional"/>
    </extension>
   </simpleContent>
  </complexType>
 </element>
</sequence>
<attribute name="accidental"
type="mpeg7:degreeAccidentalType"
default="natural"/>
<attribute name="height" type="integer"
use="optional"/>
</extension>
</complexContent>
</complexType>
```

Intensity: this is a floating value indicating the intensity of the note. It is necessary when analyzing phrasing and expressivity (crescendo, diminuendo, etc) in a melodic phrase, although it could be represented by using the *mpeg7:AudioPower* low-level descriptor.

```
  <element name="Intensity" type="float"
minOccurs="0"/>
```

Vibrato: it is also important when trying to characterize how the musical phrase has been performed; it is defined by the vibrato frequency and amplitude.

```
<complexType name =”VibratoType”>
<complexContent>
 <extension base=“mpeg7:AudioDType”>
  <sequence>
   <element name="Amplitude" type="float"/>
   <element name="Frequency" type="float"/>
  </sequence>
 </extension>
</complexContent>
</complexType>
```

Quantized instant: if one wishes to reach a high level of precision in a timing description, then the decomposition of the music segment into note segments is of interest. In addition to the handling of precise onsets and offsets of musical events, it permits to describe them in terms of position with respect to the metrical grids. In our quantized instant proposal, given a pulse reference that might be the Beat, the Tatum, etc., a note is attributed a rational number representing the number of pulses separating it from the previous one. This type can be seen as a generalization of the *mpeg7:BeatType*, improvements being the following:

- One can choose the level of quantization (the reference pulse doesn't have to be the time signature denominator as in the *BeatType*).
- Even when a reference pulse is set, one can account for (i.e. represent without rounding) durations that don't rely on this pulse (as in the case of e.g. triplets in a quarter-note-based pattern). This feature is provided by the fact that the quantized instants are rational numbers and not integers.
- The rounding (quantization) is done towards the closest beat (not towards -∞).

```
<complexType name="QuantizedInstantType">
<complexContent>
<sequence>
<element name="RefPulse" type= "pulseType"/>
<element name="Numerator" type="integer"/>
<element name="Denominator" type= "integer">
<simpleType>
<restriction base="integer">
<minInclusive value="1"/>
<maxInclusive value="9"/>
</restriction>
</simpleType>
</element>
<element name="NoteDev" type="DeviationType"
minOccurs="0"/>
</sequence>
</complexContent>
</complexType>
```

Deviation represents the deviation of a note from its closest pulse. The deviation is expressed in percentage of a reference pulse, from –50 to +50. (Here, the reference pulse can be different than that used for quantizing, one might want to quantize at the Beat level and express deviations with respect to the Tatum.) This may be useful for analyzing phrasing and expressivity.

```
<complexType name="DeviationType">
<complexContent>
<sequence>
<element name="RefPulse" type= "PulseType"/>
<element name="Dev" minOccurs="0" maxOccurs=
"unbounded"/>
<simpleType>
<restriction base="integer">
<minInclusive value="-50"/>
<maxInclusive value="50"/>
</restriction>
</simpleType>
</element>
</sequence>
</complexContent>
</complexType>
```

3.3.2. MusicDS

Mid and high-level descriptors characterize Music Segments. In addition to the *mpeg7:melody* DS, some new melodic descriptors have been incorporated, related to pitch (as the pitch range, pitch variety or pitch histogram), contour (as the melodic profile), intervals (as the interval distribution) or melodic density. For rhythmic description, we have incorporated a tempo descriptor embedded in a structure for the intertwined metrical levels that coexist in music. Additionally, the DS accounts for possible variations the musical pace. Also, the Music DS accounts for simple series of letters, permitting to describe a signal in terms of recurrences of events; with respect to the melodic, rhythmic or instrumental structures that organize musical signals.

The exact temporal location of the music segment is also described by the *mpeg7:MediaTime* attribute derived from the *mpeg7:AudioSegment*.

```
<complexType name="MusicDSType">
<complexContent>
<extension base= "mpeg7:AudioDSType">
<sequence>
 …
</sequence>
</extension>
</complexContent>
</complexType>
```

Melodic descriptors: we include in this description scheme the mpeg7 descriptors defined in the *mpeg7:Melody* DS, whose types are *mpeg7:MelodySequenceType*, *mpeg7:MelodyContourType*, *mpeg7:ScaleType* and *mpeg7:KeyType*.

As for rhythmic descriptors, there is currently no other way to deal with a change of Key and Scale than defining a different audio segments.

In addition to MPEG-7 Melody DS, other features could be attached to the *MusicDS* representing melodic aspects. We will not propose in this document a list of descriptors (see [13] for a review of different melodic descriptors).

PulseDecomposition: Several pulses (or metrical levels) coexist in a musical piece; this fact is ubiquitous in the literature. In this respect, our description of a music segment accounts for a decomposition in pulses, each pulse has a name, a beginning time index, a gap value and a rate (which is logically proportional to the inverse of the gap; some might prefer to apprehend a pulse in terms of occurrences per minute, some others in terms of milliseconds per occurrence –as in MIDI–).

Among the hierarchy of pulses, no pulse is by any mean as important as the tempo. In addition, the reference pulse for writing down the rhythm often coincides with the perceptual pulse. Therefore, it seemed important to provide a special handling of the tempo: the *PulseDecomposition* type holds a mandatory pulse named Tempo, in addition to it, several other pulses can optionally be defined. Additional pulses can be e.g., the Tatum, the Downbeat, etc.

```
<complexType name="PulseDecompositionType">
<complexContent>
<sequence>
<element name="Tempo" type="PulseType"/>
<element name="OtherPulse" type="PulseType"
minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexContent>
</complexType>
```

Name	*Definition*
Tempo	Container for the Tactus, or Beat. Permits to store its rate (the actual tempo, in beats-per-minute), its phase (through the beginning time index), and

	gap (time between two beats).
OtherPulse	Containers for other pulses. It entails the same features as the Tempo.

Pulse:

```
<complexType name="PulseType">
<complexContent>
<extension base="mpeg7:mediaTimeType">
<sequence>
<element name="PulseName" type="string"/>
<element name="PulseBeg"
type="mpeg7:mediaTimePointType"
minOccurs="0"/>
<element name="PulseGap"
type="mpeg7:mediaIncrDurationType"
minOccurs="0"/>
<element name="PulseVar"
type="mpeg7:seriesOfScalarType"
minOccurs="0"/>
<element name="PulseRate" minOccurs="0"/>
<simpleType>
<restriction base="float">
<minInclusive value="0"/>
</restriction>
</simpleType>
</element>
</sequence>
</extension>
</complexContent>
</complexType>
```

Name	*Definition*
PulseName	Name of the pulse, e.g. 'tempo', 'downbeat', 'quarter-note', 'tatum', etc.
PulseBeg	Time index of the pulse first occurrence. It makes use of the *mpeg7: mediaTimePointType* and it permits to specify the phase of the pulse.
PulseGap	Time difference between two occurrences of the pulse. It makes use of the *mpeg7:mediaIncrDurationType*.
PulseVar	Musical pace variation (see below).
PulseRate	Number of pulses per minutes. Used especially for the tempo, to express the musical speed in beats-per-minute. $pulseRate = \left(60 \cdot Fs \big/ pulseGap[samples]\right)$, where Fs is the sampling frequency.

PulseVar: This is stored as a *mpeg7:SeriesOfScalar*. It is clear that in many music pieces, pulses are not exactly regular, here resides some of the beauty of musical performance; therefore, the regular grid defined by the previous 'beginning' and 'gap' can be warped according to a time function representing tempo variations, the *PulseVar*.
Tempo curves are a common way to represent variations in the pace of a music piece, however, one might remark here that some authors argue convincingly that they should be considered with caution, see [8].

Meter: same as *mpeg7:MeterType*. The Meter is the number of beats between two downbeats. The 'beats' are by default defined in the *PulseDecomposition*, in the Tempo container (see below); although this could be left to the user: one might want to define a quarter note that would differ from the 'perceptual beat', or Tactus. This can be done in using the *OtherPulse* container. The 'downbeats' can also be defined in the *PulseDecomposition*, in the *OtherPulse* container.
The *mpeg7:Meter* does not specify the instant of occurrences of the downbeats. This information can be contained in the *OtherPulse* container, by defining a pulse as the downbeat.
An issue in using the *mpeg7:Meter* is that there is no direct link between this container and the pulses of the *PulseDecomposition*. The coherence between the meter and the different pulses is left to the user.
There is currently no other way to deal with a change of meter than defining a different audio segment.

Sequence: A simple series of letters can be added to the description of a music segment. This permits to describe a signal in terms of recurrences of events, with respect to the rhythmic structure that organizes musical signals. For instance, one may wish to categorize the succession of tatums in terms of timbres –this would look e.g. like the string '*abccacccabcd*'–, and then seek patterns (see [15] for an example).
Categorize segments of the audio "chopped up" with respect to the Beat grid might also reveal interesting properties of the signal. One might want to describe a signal in the context of several pulses. Therefore, several sequences can be instantiated.
Note that this descriptor might also be used to describe any type of patterns, would they be rhythmic, melodic or harmonic (e.g. characterizing an A-B-A form).
Rather than restricting one's time precision to that of a pulse grid, one might wish to categorize musical signals in terms of accurate time indexes of occurrences of particular instruments (e.g. bass drums and snares as in [14], [17]). This, in order to post-process these series of occurrences so as to yield rhythmic descriptors. Here, the decomposition of a music segment in its constituent instrument streams is needed (see Figure 2). For instance, a music segment can be attributed to the occurrences of the snare, another one to those of the bass-drum; timing indexes lie in the *mpeg7:TemporalMask*, inherited from the *mpeg7:AudioSegment*, that permits to describe a single music segment as a collection of sub-regions disconnected and non-overlapping in time.

```
<complexType name="SequenceType">
<complexContent>
<sequence>
```

```
<element name="RefPulse" type="pulseType"/>
<element name="SymbolicElements"
type="string"/>
</sequence>
</complexContent>
</complexType>
```

Name	Definition
RefPulse	Reference to the pulse used for description.
SymbolicElements	Symbolic description of the signal by means of a string of characters.

3.3.3. Extension of the SoundModelType

An extension of *mpeg7:SoundModelType*, that we have named *ExtendedSoundModelType*, could be used for expanding the modelling options and allowing other common ways of representing in a compact way the criteria for assigning a class to a given audio segment. This *ExtendedModelType* would be able to describe classes as Gaussian Mixture Models. The *ExtendedSoundModelType* introduces the possibility of using, instead of the restricted "default" model (i.e. a continuous HMM based on the spectrum bases), any of the models that have been defined in the MPEG-7 Model Description Scheme. Models accommodated by this DS are depicted in Figure 6. Comparing them with those that can be described in PMML, it is clear that the current modelling status of MPEG-7 is lacking (for example, it is not clear how to accommodate tree or neural models in MPEG-7).
Definition of *ExtendedSoundClassificationModel* and *ExtendedSoundModel* types:

4. CONCLUSIONS

We address the issue of musical description in a specific framework, that of the development of an application, a tool for content-based management, edition and transformation of sound samples, phrases and loops: the *Sound Palette*. We intended to cope with the description needs of this application, and therefore we have still left out issues of harmony or emotional load descriptions, as they do not seem to be priorities for such a system. We believe that adding higher-level descriptors (e.g. presence of rubato, swing, groove, mood, etc) needs solid grounding and testing on the proposed descriptors, defining interdependency rules that currently cannot be easily devised. New descriptors and description schemes have been proposed keeping also in mind the need for compatibility with the current MPEG-7 standard; they should be considered as the beginning of an open discussion regarding what we consider as the current shortcomings of the standard.

5. ACKNOWLEDGMENTS

The authors would like to thank Max Jacob for his comments, suggestions and bug fixing regarding the schema specification, and to Enric Mieza and Oscar Celma for the description examples on Classification Scheme.
This work has been funded by the European IST project CUIDADO and the TIC project TABASCO, funded by the Spanish ministry of Science and Technology.

6. REFERENCES

[1] *MPEG Working Documents.* Electronic Citation,. MPEG, http://www.cselt.it/mpeg/working_documents.htm. 2001

[2] *MPEG-7 Schema and description examples.* Electronic Citation. Final Draft International Standard (FDIS), http://pmedia.i2.ibm.com:8000/mpeg7/schema/. 2002

[3] Bilmes J. *Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm.* MSc Thesis/Dissertation MIT, Cambridge, 1993

[4] Brown J., *Determination of the meter of musical scores by autocorrelation*. Journal of the Acoustical Society of America *94,* 1993.

[5] M. A. Casey, *General sound classification and similarity in MPEG-7*. Organized Sound *6,* 153-164, 2001.

[6] Cemgil A., Desain P. and Kappen B., *Rhythm quantization for transcription*. Computer Music Journal *24,* 2000.

[7] Clarke E., *Rhythm and Timing in Music. The Psychology of Music, 2nd edition*. (Ed. Deutsch D.) Academic Press, 1999.

[8] Desain P. and Honing H..*Tempo curves considered harmful. A critical review of the representation of timing in computer music.* In Proceedings International Computer Music Conference, 1991,

[9] Desain P. and Honing H., *The quantization problem: traditional and connectionist approaches.* In *Understanding Music with AI:*

Perspectives on Music Cognition. (Ed. Balaban M., Ebcioglu K. and Laske O.) MIT Press, Cambridge 1992.

[10] Gasser M., Eck D. and Port R., *Meter as mechanism: a neural network that learns metrical patterns*. Connection Science *1,* 1999.

[11] Golub G. H. and van Loan C. F., *Matrix Computations*, The John Hopkins Press, Baltimore, MY 1989.

[12] Gómez, E. *Emilia Gómez's MPEG-7 page: Melody Description Scheme.* Electronic Citation. http://www.iua.upf.es/~egomez/mpeg7/. 2002

[13] Gómez E., Klapuri A. and Meudic B., *Melody description and extraction in the context of music content processing*. Journal of New Music Research *accepted,* 2003.

[14] Goto M., *An Audio-based Real-Time Beat Tracking System for Music with or without Drums*. Journal of New Music Research *30,* 2001.

[15] Gouyon F., Herrera P. and Cano P..*Pulse-dependent analyses of percussive music.* In Proceedings AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio, 2002.

[16] Gouyon F. and Meudic B., *Towards rhythmic content processing of musical signals - Fostering Complementary approaches*. Journal of New Music Research *accepted,* 2003.

[17] Gouyon F., Pachet F. and Delerue O..*On the use of zero-crossing rate for an application of classification of percussive sounds.* In Proceedings Digital Audio Effects Conference, 2000, Verona.

[18] Haas W. and Mayer H., *MPEG and its Relevance for Content-based Multimedia Retrieval*. Journal of Universal Computer Science *7,* 530-547, 2001.

[19] Herre J., Cremer M., Uhle C. and Rohden J. *Proposal for a core experiment on AudioTempo.* MPEG2001/8415 2002

[20] Herrera P., Peeters G. and Dubnov S., *Automatic classification of musical instrument sounds*. Journal of New Music Research *accepted,* 2003.

[21] Herrera, P., Serra, X. and Peeters, G..*Audio descriptors and descriptor schemes in the context of MPEG-7.* In Proceedings International Computer Music Conference, 1999.

[22] Honing H., *From time to time: The representation of timing and tempo*. Computer Music Journal *35,* 2001.

[23] Kartomi M., *On Concepts and Classification of Musical Instruments*, The University of Chicago Press, Chicago 1990.

[24] Lerdahl F. and Jackendoff R., *A generative theory of tonal music*, MIT Press, Cambridge MA USA 1983.

[25] Lindsay A. T. and Herre J., *MPEG-7 and MPEG-7 Audio - An Overview*. Journal of the Audio Engineering Society *49,* 589-594, 2001.

[26] Peeters, G., McAdams, S., and Herrera, P..*Instrument sound description in the context of MPEG-7.* In Proceedings of the 2000 International Computer Music Conference, 2000, Berlin.

[27] Raphael C..*Automated rhythm transcription.* In Proceedings International Symposium on Music Information Retrieval, 2001.

[28] Scheirer E., *Tempo and beat analysis of acoustic musical signals*. Journal of the Acoustical Society of America *103,* 1998.

[29] Scheirer E. and Slaney M..*Construction and evaluation of a robust multifeature Speech/Music discriminator.* In Proceedings IEEE-ICASSP, 1997.

[30] Seppänen J.. *Tatum grid analysis of musical signals.* In Proceedings IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2001.

[31] Tzanetakis, G. *Manipulation, analysis and retrieval systems for audio signals.* PhD Thesis Thesis/Dissertation Computer Science Department, Princeton University, 2002.

[32] Tzanetakis G., Essl G. and Cook P..*Human Perception and Computer Extraction of Beat Strength.* In Proceedings Conference on Digital Audio Effects, 2002.

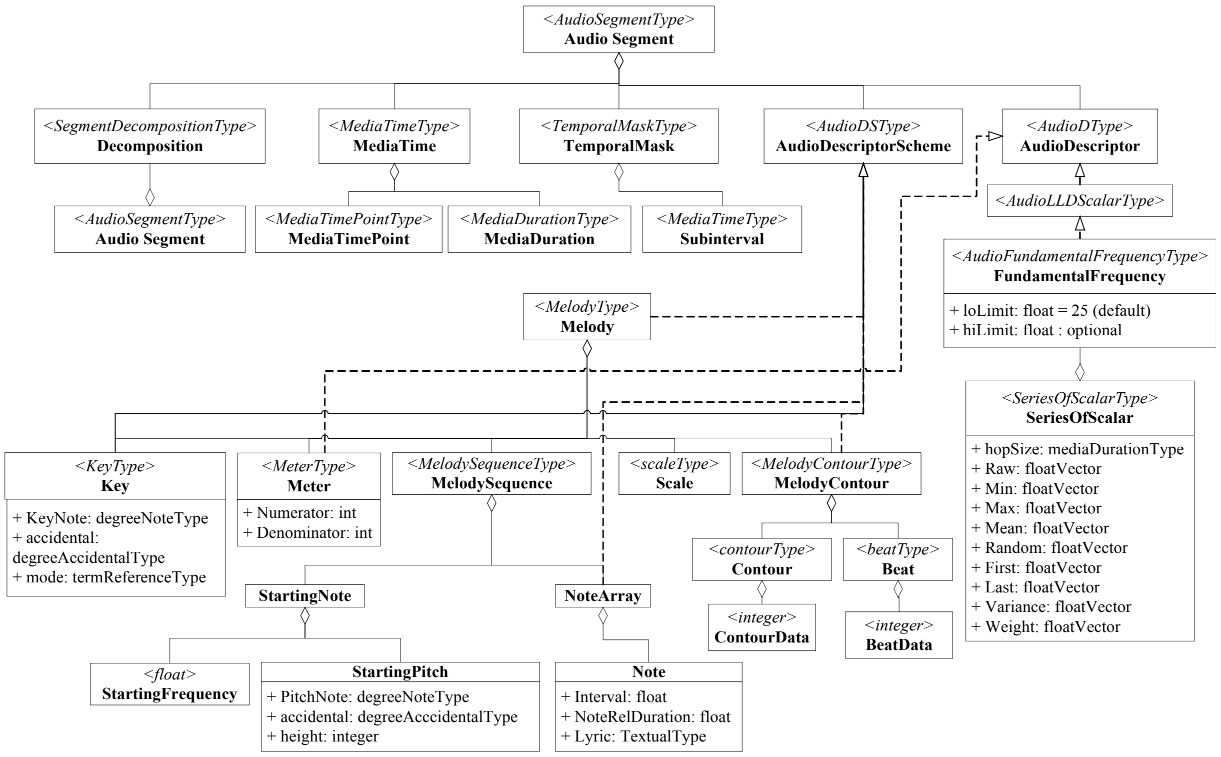
<AudioSegmentType> Audio Segment
<SegmentDecompositionType> Decomposition
<MediaTimeType> MediaTime
<TemporalMaskType> TemporalMask
<AudioDSType> AudioDescriptorScheme
<AudioDType> AudioDescriptor
<AudioSegmentType> Audio Segment
<MediaTimePointType> MediaTimePoint
<MediaDurationType> MediaDuration
<MediaTimeType> Subinterval
<AudioLLDScalarType>
<AudioFundamentalFrequencyType> FundamentalFrequency
+ loLimit: float = 25 (default)
+ hiLimit: float : optional
<MelodyType> Melody
<SeriesOfScalarType> SeriesOfScalar
+ hopSize: mediaDurationType
+ Raw: floatVector
+ Min: floatVector
+ Max: floatVector
+ Mean: floatVector
+ Random: floatVector
+ First: floatVector
+ Last: floatVector
+ Variance: floatVector
+ Weight: floatVector
<KeyType> Key
+ KeyNote: degreeNoteType
+ accidental: degreeAccidentalType
+ mode: termReferenceType
<MeterType> Meter
+ Numerator: int
+ Denominator: int
<MelodySequenceType> MelodySequence
<scaleType> Scale
<MelodyContourType> MelodyContour
<contourType> Contour
<beatType> Beat
<integer> ContourData
<integer> BeatData
StartingNote
NoteArray
<float> StartingFrequency
StartingPitch
+ PitchNote: degreeNoteType
+ accidental: degreeAcccidentalType
+ height: integer
Note
+ Interval: float
+ NoteRelDuration: float
+ Lyric: TextualType

Figure 1: MPEG-7 Melody Description Scheme

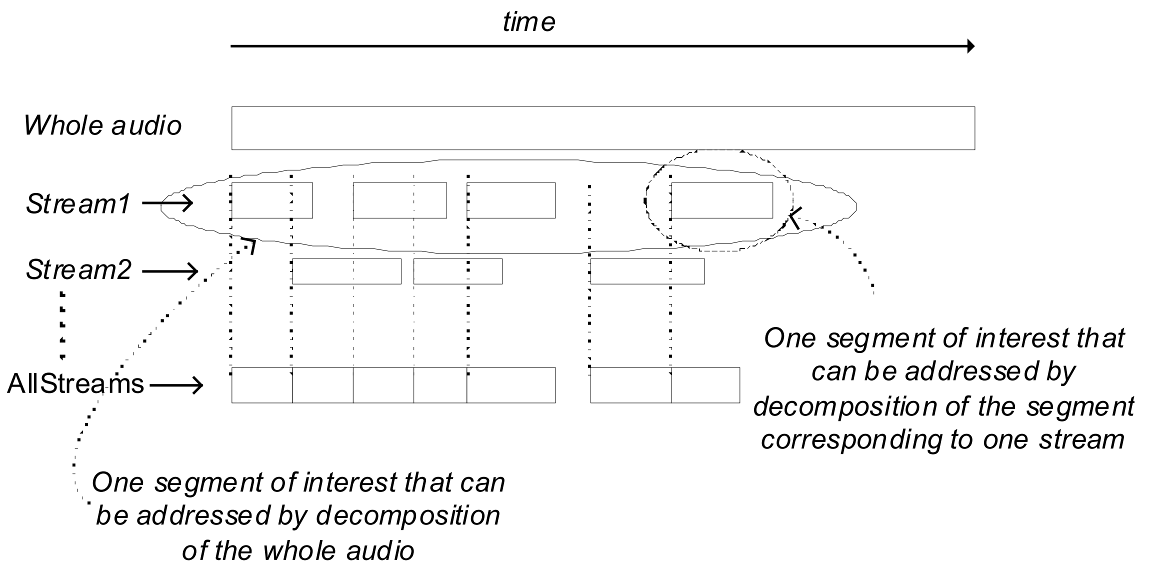
time
Whole audio
Stream1
Stream2
AllStreams
One segment of interest that can be addressed by decomposition of the segment corresponding to one stream
One segment of interest that can be addressed by decomposition of the whole audio

Figure 2: Segment Decomposition

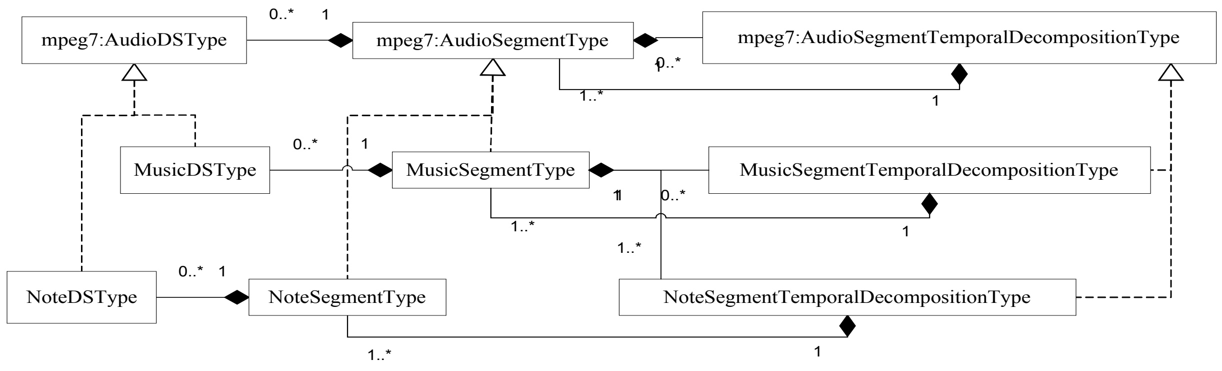
mpeg7:AudioDSType
mpeg7:AudioSegmentType
mpeg7:AudioSegmentTemporalDecompositionType
MusicDSType
MusicSegmentType
MusicSegmentTemporalDecompositionType
NoteDSType
NoteSegmentType
NoteSegmentTemporalDecompositionType
0..*
1
1..*

Figure 3: AudioSegment Derivation

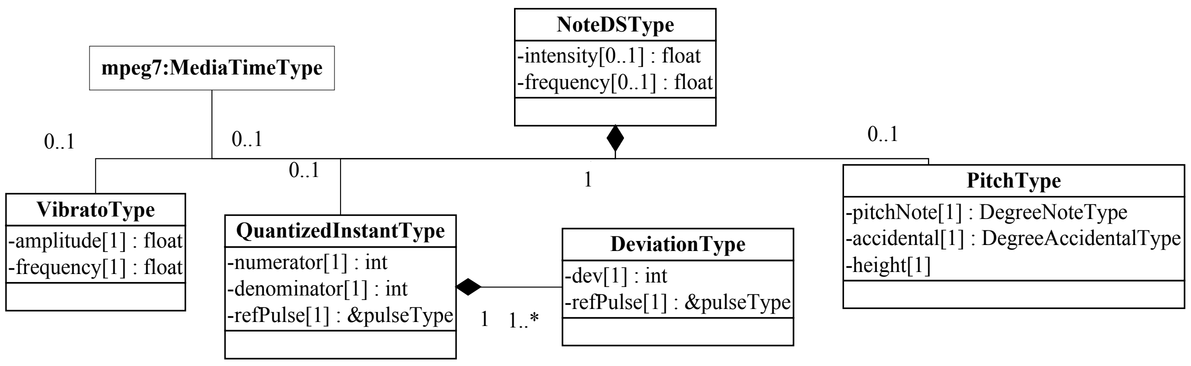

Figure 4: NoteDSType definition

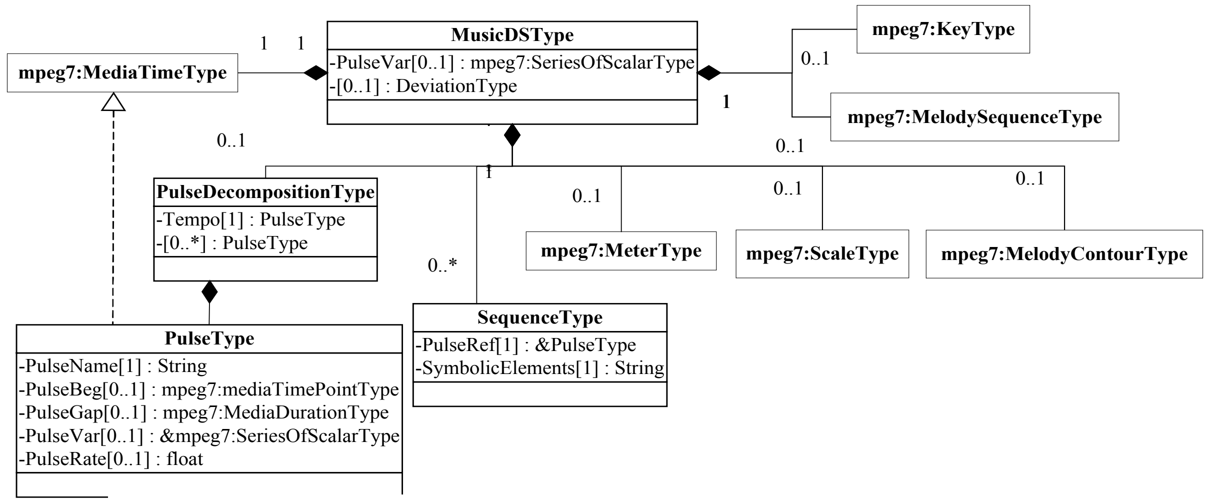

Figure 5:MusicDSType definition

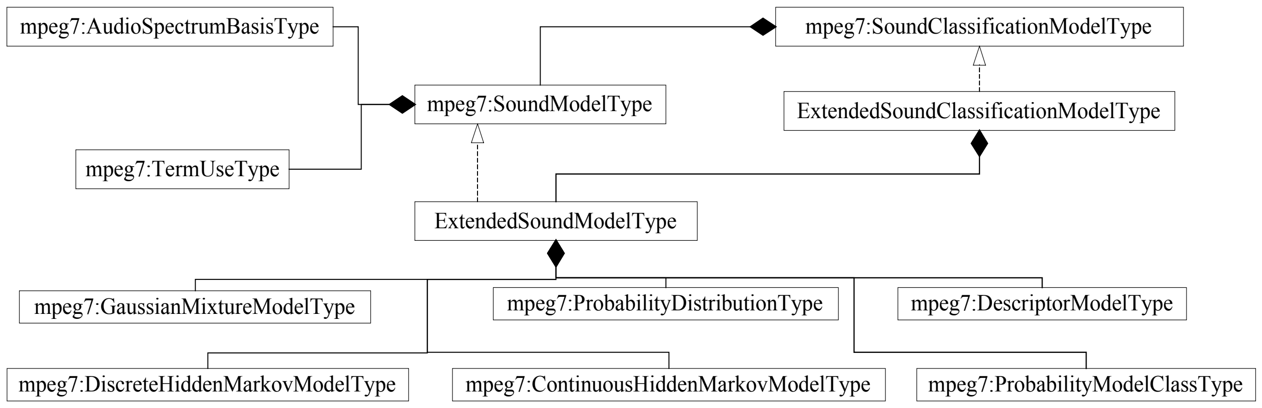
mpeg7:AudioSpectrumBasisType
mpeg7:SoundClassificationModelType
mpeg7:SoundModelType
ExtendedSoundClassificationModelType
mpeg7:TermUseType
ExtendedSoundModelType
mpeg7:GaussianMixtureModelType
mpeg7:ProbabilityDistributionType
mpeg7:DescriptorModelType
mpeg7:DiscreteHiddenMarkovModelType
mpeg7:ContinuousHiddenMarkovModelType
mpeg7:ProbabilityModelClassType

Figure 6: Extension of the SoundModelType