TRANSMITTING AUDIO CONTENT AS SOUND OBJECTS

XAVIER AMATRIAIN, PERFECTO HERRERA

Music Technology Group, IUA, UPF, Barcelona, Spain
xamat@iua.upf.es
perfe@iua.upf.es

As audio and music applications tend to a higher level of abstraction and to fill in the gap between the signal processing world and the end-user we are more and more interested on processing content and not (only) signal. This change in point of view leads to the redefinition of several "classical" concepts, and a new conceptual framework needs to be set to give support to these new trends. In [2], a model for the transmission of audio content was introduced. The model is now extended to include the idea of *Sound Objects*. With these thoughts in mind, examples of design decisions that have led to the implementation of the CLAM framework are also given.

INTRODUCTION

As applications tend to increase their level of abstraction and to approach the end-user level it seems clear that one of the focuses is to step up from the signal processing realm and directly address the content level of an audio source. The term "content-processing" is therefore becoming commonly accepted.[8][10] [17]
The basic idea when implementing a content processing scheme is to have a previous analysis step in which the content of the signal is identified and described. Then this description can be transmitted, transformed... Content description is usually thought of as an additional stream of information to be attached to the actual content. However, if we are able to find a thorough and reliable description we can think of forgetting about the signal and concentrate on processing only its description. And, as it will later be discussed, the goal of finding an appropriate content description is very much related to the task of identifying and describing the so-called sound objects.
Bearing these previous ideas in mind, a model of content transmission (see Figure 1) is proposed as a general framework for content-based applications. Any content-based application can be seen as a subset or particular module of this generic model.
The model is based on an analysis-synthesis process. Therefore, the only data involved in the transmission step is the content description taking the form of metadata. A multilevel 'content description tree' is used as an efficient representation of the identified sound object hierarchy. Several technologies are available for representing content description, but, taking into account our experience in MPEG-7's standardization process[25], we would encourage an XML-based

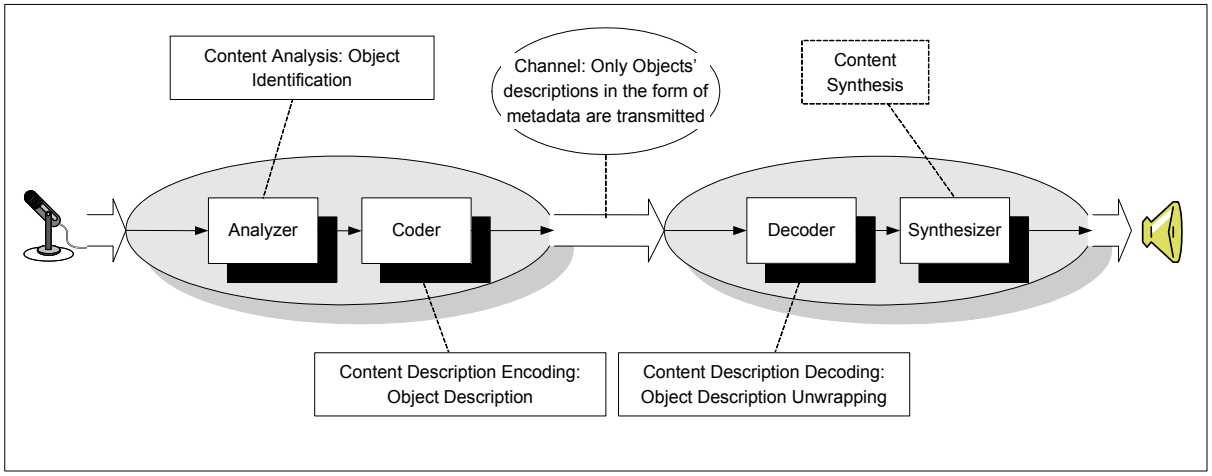

Figure 1: Content object-based transmission model

metadata language such as MPEG-7's DDL[19].
It is interesting enough to note that such a transmission model implies a redefinition of the schemes commonly used to model the communication act itself [12] as it can be seen as a step beyond Shannon and Weaver's traditional communication model [35] (see Figure 2). In our model, the stream to be transmitted is no longer seen as a stream of bits with no abstract meaning, information is an abstraction of the actual content, in other words, a 'stream of meaning'.

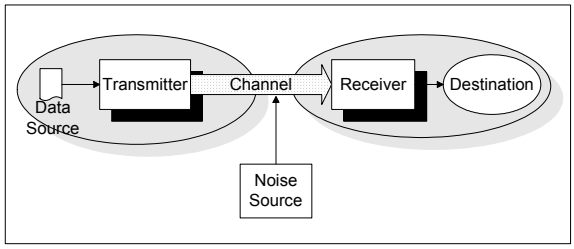

Figure 2: S&W traditional transmission model

1 THE SOUND OBJECT

The main goal of such as model is thus to analyze the signal, identify sound objects and describe them in an appropriate way. But, before getting any deeper into the different modules that make up the model, it is necessary to have a clear idea of what we mean when talking about sound objects.
Maybe the most commonly accepted definition of a Sound Object is that related to Pierre Schaefer's theories[29]. In [11], a Sound Object is defined as "any sound phenomenon or event perceived as a coherent whole (...) regardless its source or meaning". Although this definition might be useful from a psycho acoustical or perceptual point of view, it is not so from an implementation or engineering point of view.
Other explanations of an "object" from a multimedia point of view result in definitions with a narrower scope (see [37], as an example of the use of "objects" from a physical models perspective). In MPEG-7's Multimedia Description Scheme[6] an object is defined as "(...) a perceivable or abstract object in a narrative world. A perceivable object is an entity that exists, i.e. has temporal and spatial extent, in a narrative world (e.g. Tom's piano). An abstract object is the result of applying abstraction to a perceivable object (e.g. any piano)."
In this section we intend to give a clear definition of what is meant when talking about this idea. For doing so, we rely on definitions given to similar concepts in other areas. Especially we refer to the idea of Object Oriented Programming, commonly used in the computer programming knowledge corpus. It is interesting to note, though, the strong relation there has traditionally been between OO technologies and computer music or sound signal processing [26]. As a matter of fact, the definition we will later introduce can be seen as a superset and conceptual enhancement of other previously introduced concepts (see [28], for example).
Alan Kay (one of the fathers of OO when designing the Smalltalk programming language, inventor of the modern laptop and the window-based GUI system) includes in his definition of OO the sentence "everything is an object" [16]. Following this same idea, when dealing with Object Oriented Sound Processing, everything must be thought of as an object: a sound stream is an object, a track is an object, a musical note is an object, an instrument is an object…These objects have different properties and relate between them in different ways.
More precisely, and again following the OOP theory, an object is made up of an identity, a state and a behavior. The identity is the property that may be used to distinguish two instances (or objects) that have identical state and behavior. The state of an object is the summing up of the values of the different attributes or properties that an object may have. On the other hand, the behavior is the way that a particular object responds to a message (let's say a given effect or analysis algorithm, for example).
Let us see a basic example. In a sound stream we have a number of tracks one of which contains a trumpet performance. In this track, there may be different and identical notes (same pitch, same loudness, same attack type…). Thus, at first sight, we might distinguish four different kinds of objects:

- The whole sound stream
- Our set of tracks (out of which we concentrate on the one with the trumpet performance)
- The instrument in that track (trumpet)
- Any number of notes in the track

As a first and basic interpretation, the UML[24] object diagram of the system is depicted in Figure 3.

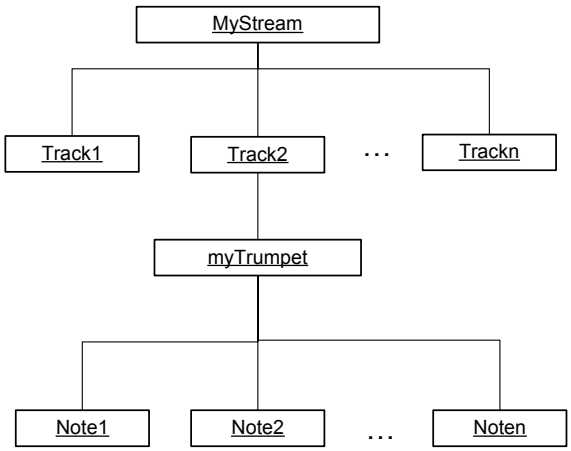

Figure 3: UML Object diagram of a simple audio stream

On the other hand, we define a class as a container of objects that comply with an identical behavior. Following the previous example, we could define what the class Sound_Stream, Audio_Track, Instrument and so on should behave like. The UML class diagram of the previous example would be the one represented in Figure 4.

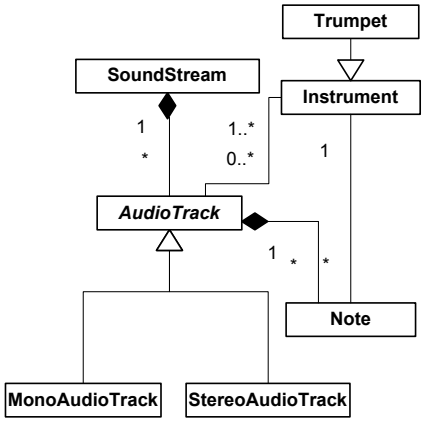

Figure 4: UML simplified class diagram representing an audio stream

which, for those not familiar with UML, should be read: a Sound Stream is made up of any number of tracks (a track can only belong to a single stream); an Audio Track is related to a single instrument and an instrument can be recorded into different tracks; an Audio Track is also made up of any number of notes which have an association relation with the instrument that produced them; trumpet is a particular case of an instrument (behaves like an instrument but may add specific behaviour) and Mono Audio Track and Stereo Audio Track are particular cases of Audio Tracks.

When declaring a class, we must ask ourselves what should be the behavior of a given class declaring methods for that purpose. The class SoundStream, for example, might have methods such as AddAudioTrack(), FindInstrument()… We should therefore identify the attributes that will be used to distinguish the state of two objects belonging to a same class. In that sense, for example, we should identify the attributes that may allow us to distinguish two different instruments (trumpet and piano). We may end up having a diagram similar to the one depicted in Figure 5.

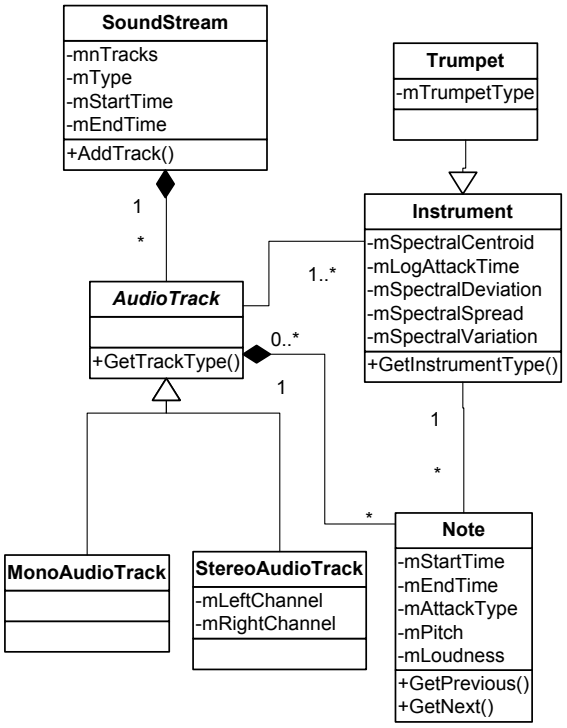

Figure 5: Complete UML class diagram

The previous diagram, though, does not explicitly show our first hypothesis of *everything* being an object. For doing so, the only missing link we should add is the fact that every class in our model should be a subclass of the SoundObject superclass. The diagram would then become (previously introduced methods and attributes are not shown for simplicity) the one depicted in the following figure:

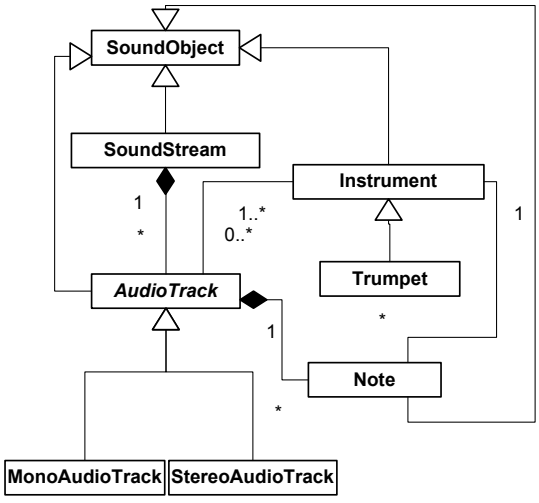

Figure 6: The "everything is a sound object" class diagram

Note that from now on, we will treat every bit of audio content as an object on its own. For that reason, the word 'content' is sometimes used as a synonym for 'object', 'content description' becoming then 'object description', for example.

2 THE ANALYSIS STEP: CONTENT EXTRACTION AND OBJECT IDENTIFICATION

The easiest way to add content description to an audiovisual chunk of information is by means of textual or oral annotation. The extraction process is in that case performed by an 'expert' that can interpret the content, extract some useful information and classify each sound object, provided there is an appropriate taxonomy available.

When thinking in terms of automatic content-extraction[31], two levels of descriptors are usually distinguished: low-level and high-level content descriptors. As a first approach, and in the broad sense, low-level descriptors are those related to the signal itself and have little or no meaning to the end-user. In other words, and thinking in terms of our domain, these descriptors cannot be 'heard'. On the other hand, high-level descriptors are meaningful and might be related to semantic or syntactic features of the sound. These latter will be the ones that will be used to classify sound objects into the class they belong.

It is obvious that the borderline between these categories is thin and not always clear. Some descriptors can be viewed as either low or high-level (or as either syntactic or semantic) depending on the characteristics of the extraction process or the targeted use. Although these categories will be used throughout this paper, we might better think in terms of a multilevel analysis scheme as the one depicted in Figure 7.

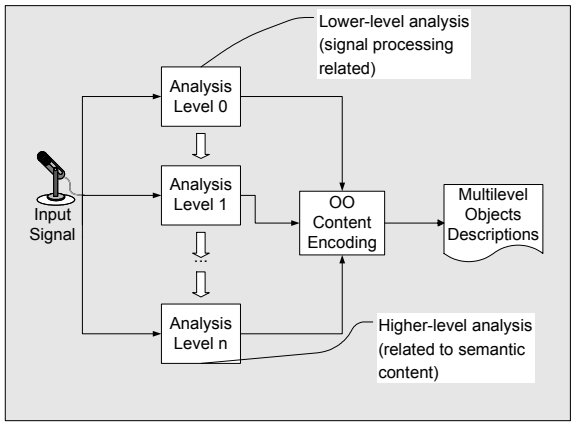

Figure 7: Multilevel analysis step

2.1 Low-level content descriptors

As mentioned before, low level descriptors are closely related to the signal itself or any of its representations. Any audio signal can be represented as a time-domain signal or as its spectral transform, and following this same idea a first (and yet incomplete) categorization, separates low-level descriptors into two categories: temporal and spectral descriptors.

Temporal descriptors can be immediately computed from the actual signal or may require a previous adaptation stage in order to extract the amplitude or energy envelope of the signal, thus only taking into account the overall behaviour of the signal and not its short-time variations. Examples of temporal descriptors are attack time, temporal centroid, zero-crossing rate, etc...

Many other useful descriptors can be extracted from the spectrum of an audio signal. These descriptors can be mapped to higher level attributes. As a matter of fact, of the basic dimensions of a sound, two of them (pitch and brightness) are more easily mapped to frequency domain descriptors and a third one (timbre) is also very closely related to the spectral characteristics of a sound. A previous analysis step needs to be accomplished in order to extract the main spectral features. For inharmonic sounds a Fourier analysis (FFT or STFT) can be enough, but a further step (which may include fundamental extraction, peak tracking and some sort of separation of the sinusoidal and residual component of the signal) is useful for the analysis of harmonic features [33]. Descriptors directly derived from the spectrum are, for example: spectral envelope, power spectrum, spectral amplitude, spectral centroid, spectral tilt, spectral irregularity, spectral shape, spectral spread…; derived from the spectral peaks: number of peaks, peak frequencies, peak magnitudes, peak phases, sinusoidality…; derived from a fundamental detection: fundamental frequency, harmonic deviation; etc...[34]

2.2 High-level content descriptors

While descriptors presented in the previous section are purely morphologic (that is, they do not carry any information on the actual meaning of the source and just refer to its inner structural elements), high-level descriptors can carry either semantic or syntactic meaning.

Syntactic high-level descriptors can be sometimes computed as a combination of low level descriptors. They usually refer to features that can be understood by an end-user without previous signal processing knowledge but do not carry semantic meaning. In other words, syntactic descriptors cannot be used to label a piece of sound according to what actually 'is' but rather to describe how it is distributed or what is made of (i.e. its structure). Thus, syntactic descriptors can be seen as attributes of our sound classes but, by themselves,

cannot be used to identify objects and classify them. For that reason, the computation of syntactic descriptors (either low or high-leveled) is not dependent on any kind of musical knowledge, symbolic or "real-world" knowledge. In [25], for example, we presented a way of describing timbre of isolated monophonic instrument notes (the scheme for computing the descriptors of a harmonic timbre is depicted in Figure 9). In the case of our timbre descriptor, for example, the resulting descriptor is not sufficient to label a note as being 'violin' or 'piano' but rather to compute relative perceptual distances between different instrument samples.

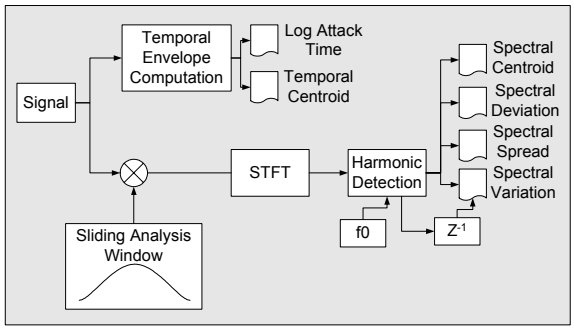

Figure 8: Combining low-level descriptors for creating higher-level syntactic descriptors: MPEG-7's Timbre Descriptor

When trying to label a chunk of audio with a semantic descriptor we are implicitly performing a classification activity and thus identifying sound objects. We therefore need to apply more high-level or real world knowledge. The degree of abstraction of a semantic descriptor though has a wide range, labels such as 'scary' or more concrete such as 'violin sound' can be considered semantic descriptors.

The main purpose of a semantic descriptor is to label the piece of sound to which it refers using a commonly accepted concept or term that corresponds to a given "sound class" (e.g. instrument, string instrument, violin...). It is interesting to note that, in this case, the classification process is performed in a top-down manner. Using low-level or high-level syntactic descriptors we might be more or less immediately be able to identify our piece of sound in as belonging to a quite abstract class (in the worst case we are always able to classify it as a Sound Object). Applying both real-world knowledge and signal processing knowledge we may be able to get our problem to a more concrete ground and start *down-casting* our description to something like "string instrument" or "violin" (see Figure 9).

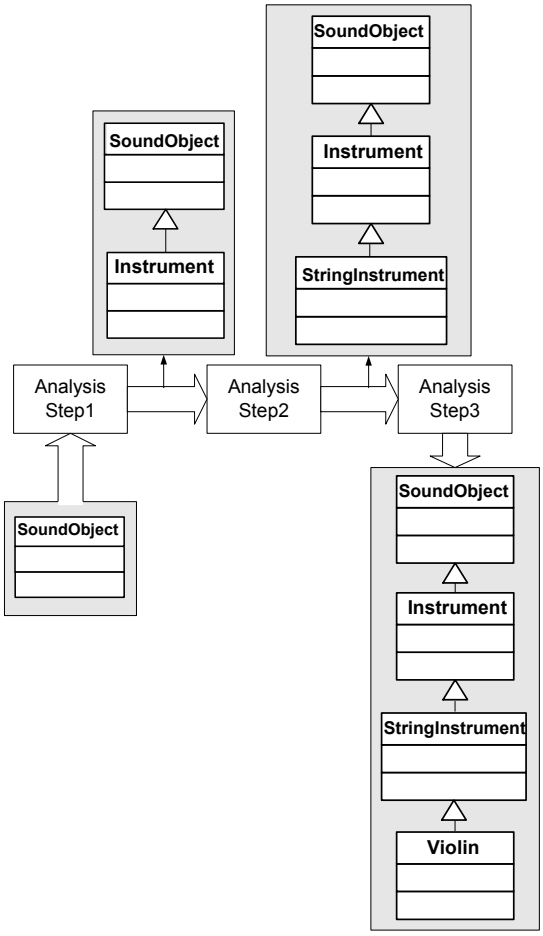

Figure 9: Multilevel semantic analysis/classification and polymorphic objects

Other semantic descriptors, though, do not aim at classifying the sound but rather at describing some important feature or attribute. These descriptors can classify a sound as 'loud', 'bright' 'scary'... In this case, conversely to what happened with the previous classifiers, the more concrete a feature is the easier it will be to derive it from our previously computed low-level or high-level syntactic descriptors. For example, a label like "bright" might be directly derived from the "spectral centroid" low-level descriptor. Much more real-world knowledge must be applied to be able to classify a sound as "sad" or "frightening" (see Figure 10).

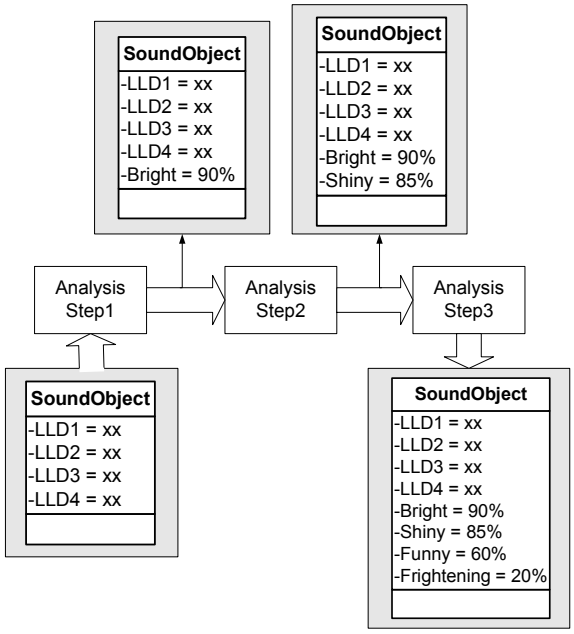

Figure 10: Multilevel semantic analysis for adding higher-level abstract features

Different proposals have been made in order to create a semantic map or multi-level structure for describing an audio scene, ones of them being the ten-level map presented in the MPEG Geneva meeting (May, 2000)[15]. This proposal includes four syntactic levels and six semantic levels: Type/Technique, Global Distribution, Local Structure, Global Composition, Generic Objects, Generic Scene, Specific Objects, Specific Scene, Abstract Objects, and Abstract Scene. (Note: the word 'object' is used here as a synonym of 'source' and should not be understood in the sense of the definition given in this paper).

While that proposal is quite theoretical and simple, and comes from a generalization of a similar structure proposed for video description, other proposals come from years of studies on the specific characteristics of an audio scene and have even had practical applications. One of the most renowned techniques that can fit into this category is CASA (Computer Auditory Scene Analysis)[7]. It is far beyond the scope of this paper to go deep into any of these proposals, but it is interesting to note that CASA has addressed the issue of describing complex sound mixtures that include music, speech and sound effects, also providing techniques for separating these different kinds of streams into sound objects (see [23], for example).

3 THE CODING STEP (CONTENT DESCRIPTION)

In the coding step, all the content information extracted in previous steps needs to be encoded in an appropriate format. Binary and textual based versions of the format should be provided in order to provide both coding and transmission efficiency and readability. It is also important for the coding scheme used to offer support to the way that the output of our analysis block is organized. In that sense, it is necessary to use a highly structured language that enables the description of a tree-like data structure giving also support to object oriented concepts.

Maybe the first idea that comes to mind is using UML as a way of describing our content. UML is indeed a highly structured language and supports all OO concepts. It would be an excellent choice for describing our Sound Classes. But it is not so appropriate if what we want is to describe the state of our objects/instances or, in other words, make our objects persistent. The UML Object Diagram (see Figure 3) does not seem a good choice for doing so.

But there are many examples of coding schemes used for encoding metadata or, more precisely, audiovisual content description, perhaps the most ambitious being MPEG7. Although MPEG7 is focused on search and retrieval issues, the actual encoding of the audiovisual content description is flexible enough as for being used by a system as the one proposed in this article[13][18]. It is based in an extension of W3's XML-Schema called MPEG7's DDL (Descriptor Definition Language). XML-Schema is a definition language for describing the structure of an XML document using the same XML syntax and it is supposedly bound to replace the existing DTD language. It is thus a tagged textual format but it also includes support for most Object Oriented concepts [38]. Note so, that XML-Schema will be the language used for structuring our content or defining Sound Classes, but the actual output of the analysis or content of the identified objects will be a standard XML document. See [9] for a thorough example of how MPEG-7's DDL may be used to serve our purposes, defining in this case a multilevel content hierarchy for sound effects. Furthermore, and as introduced in [14], a mapping can also be accomplished between UML-described classes and XML output.

On the other hand, the encoding step must also be in charge of deciding the degree of abstraction to be applied to the output of the content extraction step. This decision must be taken on the basis of the application and the user's requirements although it will obviously affect the data transmission rate. The encoder must decide what level of the content tree should indeed be encoded depending on the degree of concreteness demanded to the transmission process, degree that will usually be fixed by the particularities of the receiver. If only high-level semantic information is encoded, the receiver will be forced to use more of its 'artificial imagination' (see next section). The more low-leveled the information encoded is, the more 'real world knowledge' the receiver should have.

Another subject, which will not be dealt with in this paper, is how this textual information could be compressed and transformed into a more efficient binary format suitable for transmission.

4 THE DECODING STEP: CONTENT INTERPRETATION

The main task of the decoder is to interpret the information received through the channel in order to be able to feed the Synthesizer with the correct parameters. Encoded sound objects must be interpreted and prepared for the next module's requirements. Two main processes are expected from the decoder depending if the content description received is high or low level. We will now detail their main characteristics.

If the decoder is input low-leveled descriptions, there are two options, depending on the application requirements. The low level descriptors can be directly fed into the Synthesis engine or there can be an intermediate 'abstraction process' (see Figure 11). In the abstraction process, the decoder has to use 'real world' knowledge in order to convert low-level information into mid-level information, more understandable from the synthesizer point of view. If the abstraction process is omitted and the synthesizer receives low-level information but this description is not exhaustive, those parameters not specified should be taken as default. Thus, paradoxically, the synthesizer is granted some degrees of freedom and the result may loose concretion. An example of this situation would be an input like 'sound object, centroid=120Hz". It is obvious that many sound objects comply with this low-level description, the decoder would be in charge of adjusting other necessary parameters.

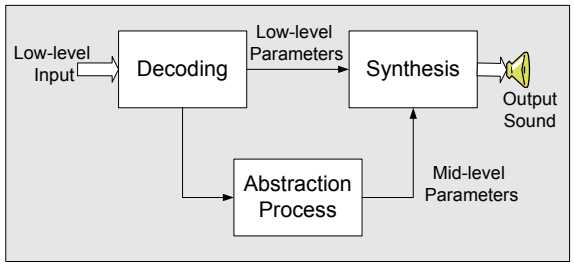

Figure 11: Low-level input to the Decoder: Abstraction Process

If the input to the decoder consists only of high-level semantic information, an intermediate 'inference' process is always needed in order to make the content description understandable by the synthesis engine (see Figure 12). This process, contrary of the 'abstraction process' earlier mentioned, might be better understood by using an example. Imagine the decoder's input is 'violin.note'. The synthesizer will be unable to interpret that content description because of its degree of abstraction. The decoder is thus forced to lower the level of abstraction by suppressing degrees of freedom. The output of the decoder should be something like 'violin note, pitch: C4, loudness: mf…'.

Both abstraction and inference are indeed one-to-many process, that is, the same input should yield a finite set of different outputs. The way the decoding process gets rid of the degrees of freedom should rely on user or application preferences as well as on random processes or context awareness. In the previous example, the decision on the note and loudness to be played could be based on knowledge on the author, the style, the user's likes, previous or future notes, harmony and a final random process to choose one of the best alternatives.

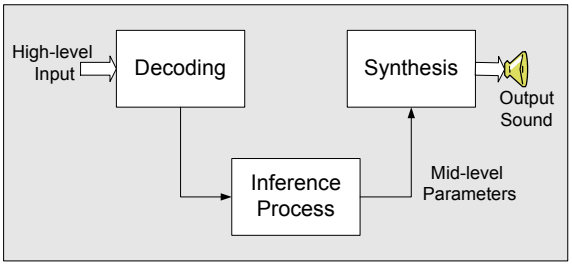

Figure 12: High-level input to the Decoder: Inference Process

5 SYNTHESIS STEP

The key point of the language used for expressing synthesis parameters is that it must not only meet the requirements of the synthesizer's input but also the needs of the decoder's output.

Many languages have been developed for the purpose of controlling a synthesizer [1][20][32]. Among them, the most extended one is MIDI [21][22] although its limitations make it clearly not sufficient for the system proposed in this paper. Another synthesis language that deserves consideration at this point is MPEG4's SAOL (Structured Audio Orchestra Language)[30] [36].

SAOL has been mostly developed at the MIT and has been recently standardized by MPEG and included in MPEG4. SAOL is indeed an evolution of the well known CSound synthesis language and also includes support for some OO concepts. The main advantage of using SAOL at this point of the process is that it should be possibly linked into the parameters coming out from the analysis step, provided that MPEG7 was used at the encoding process.

6 A COMBINED RECEIVER SCHEME: CONTENT-BASED SYNTHESIS

Although sometimes it may be useful to conceptually separate the receiver into a decoder and a synthesizer,

many other times, a combined scheme that treats the receiver as a whole will be more feasible.
In that case, the resulting receiver scheme is what we call a "Content-based Synthesizer", or "Object-based Synthesizer" which, at first sight, does not defer much from that of a "traditional" synthesizer (see Figure 13).

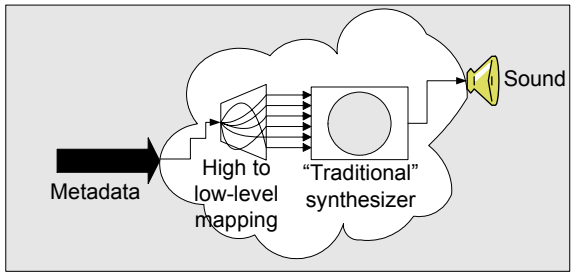

Figure 13: Combined scheme for modeling the receiver of a content transmission system

In a general situation, a simple mapping strategy may be sufficient. But if the level of abstraction of the input metadata is higher, the gap between the information transmitted and the parameters that are to be fed to the synthesis engine might be impossible to fill using conventional techniques. Imagine for example a situation where the transmitted metadata included a content description such as: [*genre: jazz, mood: sad, user_profile: musician*].
The latter example leads to the fact that we are facing a problem of search and retrieval more than one of finding an appropriate mapping strategy. We could have a database made up of sound files with an attached content description in the form of metadata. The goal of the system is then to find what object in the database fulfils the requirements of the input metadata (see Figure 14).

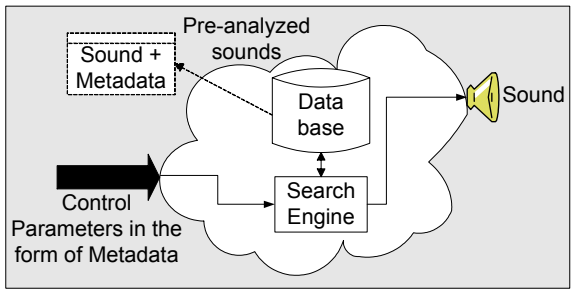

Figure 14: Search and retrieval as a means of synthesizing

A problem we still have to face with such a model is the difficulty to automatically extract parameters with such a level of abstraction from the signal itself. We can find examples of existing applications that implement the system depicted in the previous figure but they always need a previous step of manually annotating the content of the whole database.
A possible solution to this "inconvenience" is the use of machine learning techniques. It is recently becoming usual, in this sort of frameworks, to implement, for example, collaborative filtering engines (classification based on the analysis of users' preferences: "if most of our users classify item X as being Y, we label it that way"). In that case though, the classification and identification is performed without taking into account any inner property of the sounds. On the other hand, if what we intend to have is a system capable of learning from the sound features, we may favor a Case-Based Reasoning (CBR) engine as the one used in [5].

7 THE CLAM FRAMEWORK

Bearing all previous considerations in mind, and using an approach derived from the different models previously introduced, the CLAM framework is being developed in our team. CLAM stands for C++ Library for Audio and Music and means "a continuous and loud sound produced by people as approval or disagreement with a given event" in Catalan.
The CLAM framework has two operating modes: unsupervised and supervised. In the former, it may be used as a regular open-source/cross-platform C++ library. In the second mode (still in development) a box-and-arrows approach is used in order to build fast prototypes for general music and audio processing.
CLAM includes utilities such as cross-platform Audio and MIDI stream I/O, file I/O and a number of processing algorithms both in the time and frequency domain, all of it cross-platform and tested completely under Linux and Windows and partially under MacOS. At this moment, more than 250 C++ classes have been implemented, resulting into about 50,000 lines of code.
Different kind of objects are distinguished in the framework but the most widely spread categories are the Processing and the Processing Data objects. Processing classes embed any process that can be actually accomplished in the framework. This processing is performed as response to a call to a compulsory Do() method. The input/output of data to the processing objects is done manually (as arguments of the Do() method) or using an interface of Ports in the non-supervised mode. In any case, the input/output to a Processing object must always be a Processing Data object. Apart from Ports, Processing objects also own control and configuration objects (see Figure 15).

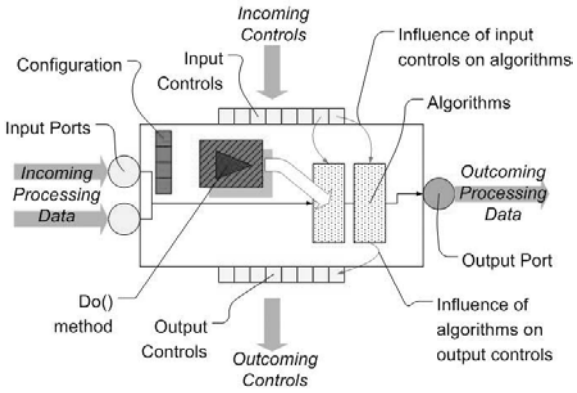

Figure 15: Model of a Processing object in the CLAM framework

Processing Data classes embed all kind of data that can be processed in the system. They include classes for holding audio, spectrum, spectral peaks, segments, frames, descriptors... Processing Data classes are implemented using a macro-derived system generically called Dynamic Types that enable run-time instantiation of attributes and ensure a homogeneous interface for all Processing Data objects.

Both Processing and Processing Data can be made persistent (stored to disk) or loaded on run-time and in any part of the thread. For doing so, XML has been chosen as a general-purpose structured language although the architecture allows easy extension to other formats such as SDIF. The structure of the XML document correspond to the actual structure of the object in memory[14].

Although the framework is far from finished it has already been used in a number of internal research projects with applications as distant as near lossless time-stretching, high quality saxophone synthesis, MPEG7 compliant description analysis toolkit, real-time processing for electro-acoustical performance and SMS analysis/synthesis toolkit. CLAM will be released as open-source as part of the AGNULA IST European project.

8 CONCLUSIONS

As detailed in the different sections, the model proposed is based on a new paradigm: the transmission of content as sound objects. The model must be understood as a working framework rather than as a system that should be due in the short term. Even so, the necessary technologies to implement the different modules are already available or are expected to be in a short term as interest in content-processing grows among different research teams.

One may question the benefits of content-based audio applications. By concentrating on the transmission of content description we are actually favoring the distinction between content and its realization. And, by doing so we favor a higher level approach, encapsulation, concept reuse, the upcoming of new applications (i.e. content-based transformations), data reduction, and robustness enhancement, to name a few.

9 ACKNOWLEDGEMENTS

The work reported in this paper has been partially funded by the IST European project CUIDADO and by the TIC national project TABASCO.

REFERENCES

[1] Amatriain, Xavier; Bonada, Jordi; Serra, Xavier. "METRIX: A Musical Description Language and Class Structure for a Spectral Modeling Based Synthesizer" *in Proceeding of the Digital Audio Effects Workshop (DAFX98)*. Barcelona, Spain, 1998.

[2] Amatriain, Xavier; Herrera, Perfecto. "Audio Content Transmission", *in Proceeding for the 2001 DAFX Conference*. Limerick, Ireland, December 2001.

[3] Amatriain, X; Bonada, J.; Loscos, A.; Serra, X..; "Spectral Processing" in *DAFX. Digital Audio Effects*. John Wiley and Sons, Ltd. (In Press). 2002.

[4] Amatriain, X; Bonada, J.; Loscos, A.; Arcos, J. L., Verfaille, V. "Addressing the Content Level in Audio and Music Transformations", in Journal of New Music Research, 2002. (In Preparation)

[5] Arcos, J. L.; R. López de Mántaras; X. Serra. 1998. "Saxex: a Case-Based Reasoning System for Generating Expressive Musical Performances", *Journal of New Music Research, Vol. 27, N. 3*, Sept. 1998.

[6] van Beek, P.; Benitez, A..; Heuer, J.; Martinez, J.; Salembier, P.; Shibata, Y.; Smith, J.R. ; Walker, T. "Text of 15938-5 FCD Information Technology – Multimedia Content Description Interface – Part 5 Multimedia Description Schemes", Singapore. 2001.

[7] Bregman, A. S. *Auditory Scene Analysis: the Perceptual Organization of Sound*, MIT Press, Cambridge, MA, 1990

[8] Camurri, Antonio; "Music Content Processing And Multimedia: Case Studies and Emerging Applications of Intelligent Interactive Systems", *Journal of New Music Research, Vol.28 No.4*, 1999.

[9] Casey, Michael; "MPEG-7 Sound-Recognition Tools" in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11 (6). June, 2001

[10] Chiariglione, Leonardo; "The Value of Content", *Technology Reviews*, March 2000.

[11] Chion, Michel; *Guide des Objets Sonores. Pierre Schaeffer et la Reserche Musicale*. INA-GRM/BUCHET.CHASTEL. 1983

[12] Darnell, Donald; *Approaches to Human Communication*, Richar Budd and Brent Ruben eds, Spartan Books, New York, 1972

[13] Ebrahimi, Touradj and Christopoulos, Charilaos; "Can MPEG-7 be used beyond database application?", input document for the Atlantic City Meeting of MPEG, October 1998. Doc. num. M3861

[14] Garcia, David; Amatriain, Xavier. "XML as a means of control for audio processing, synthesis and analysis." *in Proceedings of the MOSART Workshop on Current Research Directions in Computer Music*. Barcelona, Spain, 2001.

[15] Jaimes, Alejandro; Benitez, Ana B.; and Chang, Shih-Fu; "Multiple Level Classification of Descriptions for Audio Content", input document for the Geneva Meeting of MPEG, May 2000. Doc. num. M6114

[16] Kay, Alan. "The Early History of Smalltalk" *in Proceedings of the Second ACM SIGPLAN History of Programming Languages Conference. ACM SIGPLAN Notices 28(3): 69-75*. 1993

[17] Karjalainen, M. "Immersion and content- a framework for audio research", *in Proceedings of the IEEE Workshop of Applications of Signal Processing to Audio and Acoustics*, 1999.

[18] Lindsay, Adam and Kriechbaum, Werner; "There's More Than One Way to Hear It: Multiple Representations of Music in MPEG-7", *Journal of New Music Research, Vol.28 No.4*, 1999.

[19] Martínez, Jose M., "Overview of the MPEG-7 Standard", document number: ISO/IEC JTC1/SC29/WG11 N4031 http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm

[20] Mc Millen, Keith. 1994. "ZIPI: Origins and Motivations". *Computer Music Journal 18(4). pp 48-96*

[21] MIDI Manufacturers Association. *MIDI 1.0 Detailed Specification.* Los Angeles: The International MIDI Association. 1998

[22] Miles Huber, David.1991. *The MIDI manual*. USA. Howard W.Sams.

[23] Nakatani, Tohomiro and Okuno, Hiroshi G.; "Sound Ontology for Computational Auditory Scene Analysis", *in proceeding for the 1998 conference of the American Association for Artificial Intelligence*.

[24] OMG (Object Management Group); *OMG Unified Modeling Language Specification. Version 1.4*. September 2001.

[25] Peeters, Geoffroy; Herrera, Perfecto; Amatriain, Xavier. "Audio CE for Instrument Description (Timbre Similarity)", Input document for the Maui Meeting of MPEG, November 1999. Doc. num. m5422

[26] Pope, Stephen Travis (editor). *The well-tempered object, Musical Applications of Object-Oriented Technology*. MIT Press. 1991.

[27] Rolland, P.; Pachet F. 1995. "Modeling and Applying the Knowledge of Synthesizer Patch Programmers" in *G. Widmer (ed.), Proceedings of the IJCAI-95 International Workshop on Artificial Intelligence and Music, 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada.

[28] Scaletti, C. and Hebel, K. "An Object-based Representation for Digital Audio Signals", in *Representations of Musical Signals*. MIT Press. 1991.

[29] Schaeffer, Pierre; *Traité des Objets Musicaux*. Editions Du Seuil. 1966.

[30] Scheirer, Eric D.; "SAOL: The MPEG-4 Structured Audio Orchestra Language", *Proceeding for the 1998 ICM*.

[31] Scheirer, Eric D.; "Music Listening Systems", Phd Thesis for the MIT, June 2000

[32] Selfridge-Field, Eleanor.1997. *Beyond Midi, The Handbook of Musical Codes*. MIT Press.

[33] Serra, X. 1996. "Musical Sound Modeling with Sinusoids plus Noise", in *G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors, Musical Signal Processing*. Swets & Zeitlinger Publishers.

[34] Serra,X. and Bonada,J. "Sound Transformations Based on the SMS High Level Attributes". *Proceedings of the Digital Audio Effects Workshop (DAFX98)*, Barcelona, November 1998.

[35] Shannon, Claude and Weaver, Warren; *The Mathematical Theory of Communication*, Univ. of Illinois, Urbana, 1949

[36] Synthetic/Natural Hybrid Coding (SNHC) section of the MPEG-4. "Final Committee Draft Version 1.8." Document num.FCD ISO/IEC 14496-3 Subpart 5. MIT Media Laboratory. http://sound.media.mit.edu/mpeg4

[37] Tolonen, Tero. "Object-Based Source Modeling for Musical Signals", *in proceedings of the 109th AES Convention*, Los Angeles, 2000.

[38] W3's XML-Schema homepage, [http://www.w3.org/XML/Schema]n