Formal Music Representation; a Case Study: the Model of Ravel's Bolero by Petri Nets †

Goffredo Haus & Antonio Rodriguez

Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Moretto da Brescia, 9
I-20133 Milano (Italy)

INDEX

Keywords: modelling, Petri Nets, music texts, music objects, music algorithms, formal languages, timbral textures

† This research is supported by the Italian National Research Council in the frame of the MUSIC Topic (LRC C4): "INTELLIGENT MUSIC WORKSTATION", Goal C: SISTEMI AVANZATI DI PRODUTTIVITA' INDIVIDUALE, Subproject 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Finalized Project SISTEMI INFORMATICI E CALCOLO PARALLELO.

1. The Main Steps of this Work

Our research began in 1985 when Antonio Rodriguez came to the L.I.M. - Laboratorio di Informatica Musicale from Valencia to define a formal model of the Ravel's Bolero together with Goffredo Haus. This paper tries to outline the main results of the formalization process we have had.

As the methodological path we have followed a sequence of research steps:
a) analysis of the original orchestral score by M. Ravel;
b) recognition of the music objects available within the score;
c) definition of the music objects' map;
d) further analysis of the original score;
e) recognition of the growing rules hidden within the score;
f) analysis formalization;
g) hypotheses of modelling architectures;
h) definition of the model of Ravel's Bolero based on a "music objects & growing rules" approach;
i) score synthesis experimentation by means of the formal model defined.

In § 2. we summarize the musicological analysis we have done (research steps *a-f*), while in § 3. the formal model definition is described (research steps *g-h*); in § 4. we discuss about model variants, alternative modelling solutions and other further scientific-musicological experimentations (research step *i*).

Two relevant aims of our research are the understanding and the modelling of the compositive processes, so that we can define abstract tools suitable for music description and processing which allow to analyze and synthesize scores by means of a Ravel-influenced set of composition tools.

Particularly, we will try to show how Ravel has used certain kinds of instrumentation techniques which are very similar to some of the typical computer music techniques. Furthermore, as a result of our research we are now convinced that Ravel has used the traditional orchestra as the rough means to implement his abstract idea of the Bolero and that the synthetic music tools now available would have been more suitable for the achievement of his music goals.

2. The Analysis of Ravel's Bolero

In this paragraph, we discuss a representative overview of our analysis of *Bolero* by M. Ravel [1]. The choice of the Ravel's Bolero as the analyzing subject was made to give a wider understanding to the reader, due to the popularity of the piece.

Trying to understand the compositive processes, we have naturally left common music notation (i.e. the traditional score) as the representing tool for music information because we are interested in a more abstract level than the performer's one; in other words we have moved from the symbolic (score) level to the structural level of music representation.

Our work is not concerned with single notes but with music notions as melody, rhythm, dynamics, harmony and with score chunks which we call music objects. The following paragraphs will discuss these concepts. On the other hand, our work is concerned with the "life" of structural concepts within the whole composition with particular emphasis on their dynamic behaviour.

2.1. Music "objects"

In this work we show how music structures can be pointed out and processed by means of a more abstract kind of representation than the staff one; this new language allows to explicitly describe and process what we call *music objects* . A music object may be anything could have a music meaning and that we think as an entity, either simple or complex, either abstract or detailed, an entity with a name and some relationship with other music objects.

Therefore, we can describe musical objects at various abstraction levels within a hierarchical context of description; i.e. the structural level, the score level, the acoustical level and any other level we can think as a suitable one for music representation purposes.

Ravel's Bolero may be considered as a structure of music objects which appear and disappear within the music text following exact and systematic rules, which we call *growing rules* . The various kinds of music objects have different roles; there are four kinds of music objects with respect to the aim:

a) *rhythmic objects:*	**A**, **A1** and **A2** objects;
	C object;
	E object;
b) *melodic objects:*	**B1** and **B2** objects;
c) *harmonic objects:*	**D1** object;
	G object;
d) *resolution objects:*	**B3**, **B4**, **Bres, A3**, **Ares, D2** objects.

Music objects often have a manifold role within a composition. Some of the melodic, rhythmic and harmonic objects of Bolero have a specific role within the timbral development too: **B1**, **B2**, **A1**, **C**, **E**, **D1** and **G** objects. The harmonic objects development produces a kind of musical accompaniment. The **E** object performs the role of the timbal like a metaphorical conductor. **A**, **A1**, **A2**, **C** and **E** objects have a rhythmic role most of all. Object **D2**, which is running only in the resolution phase, has a complex role and acts like a synthesis of many objects with rhythmic, dynamics and harmonic roles.

Furthermore, we can identify also other music objects depending on either performance features or other global score structures. The score is fit for many different kind of readings. We give three meaningful examples. The **F** object (see Fig. 1) serves to allow performers' rests during the execution of the score. The **Clock** object (see Fig. 1) integrates the roles of the other objects; its role is something like a smoothing function which softens the steps of the growing rules. In the sopranino sax part (see reference N° 7 in the score, i.e. the second appearance of the minor theme) music object **B2** is assigned excepting a brief subobject which is assigned to the soprano sax; this special solution is due to the exceeding pitch range of the part with respect to the sopranino range. Anyway, we neglect these kind of objects and phenomena within the score because they are strictly alternative to the ones we have used in the formalization process of the score; we have privileged objects which allow to represent the most ideal structure from the compositive point of view rather than from the performance one.

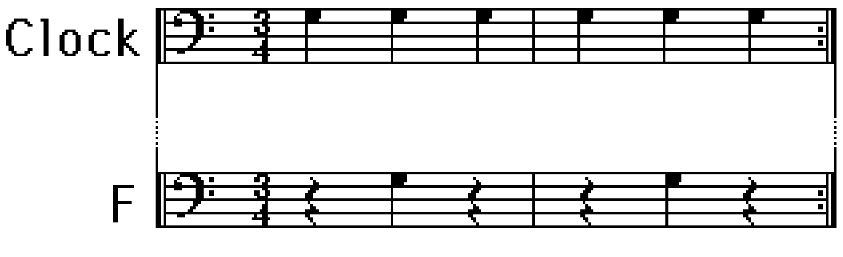

Figure 1 Examples of alternative music objects.

In the brief description of the music objects we distinguish between the specific resolution objects and the other music objects owing to the structural characteristics of the piece, and consequently of our model, which will be discussed in § 2.2. and § 3.4.

2.1.1. Rhythmic Objects

The first music object which comes to light is the two-bars rhythmic pattern of the drum which is obsessively repeated during all the piece; this object seems to be the driving one and it appears in the following variants within the score (see Fig. 2):

A: when the object is assigned to a percussion instrument;
A1: when the object is assigned to a tuned instrument; in this case, the object is completely described by two parameters: the octave and the note with respect to the pitch;
A2: when the object is assigned to a tuned instrument and has an arpeggio form; it is completely described by two parameters: the octave and the key-note of the arpeggio.

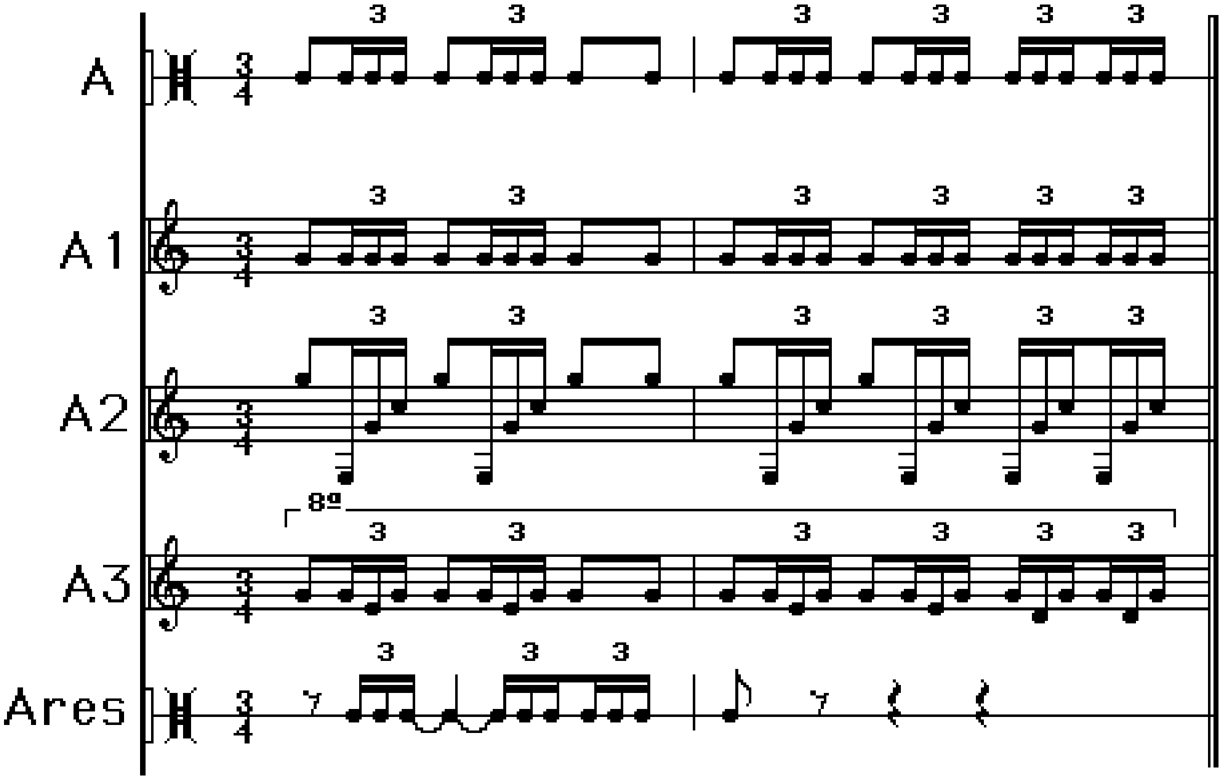

Figure 2 The complete family of the **A** objects (including **A3** and **Ares**).

The other rhythmic objects (see Fig. 3) are:

C: it is a two-bars object which acts as a contrapedal structure;
E: it is a two-bars object which has a weighing effect particularly enhanced by the timbals, but at the beginning of the piece it is assigned to the cellos.

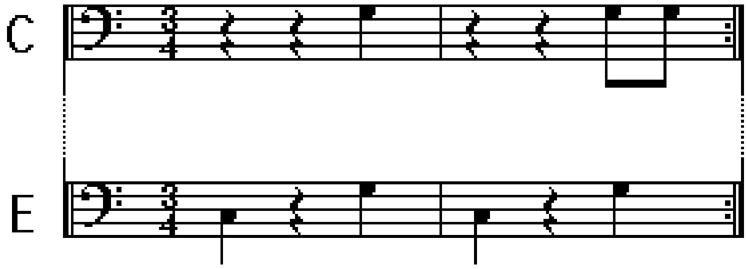

Figure 3 The other rhythmic objects (**C** and **E**).

2.1.2. Melodic Objects

Melody is made up of a major theme (object **B1**) and a minor theme (object **B2**), each one 18-bars long (see Fig. 4 and 5); they are alternatively presented since the beginning to the Finale of the piece (see Fig. 9).

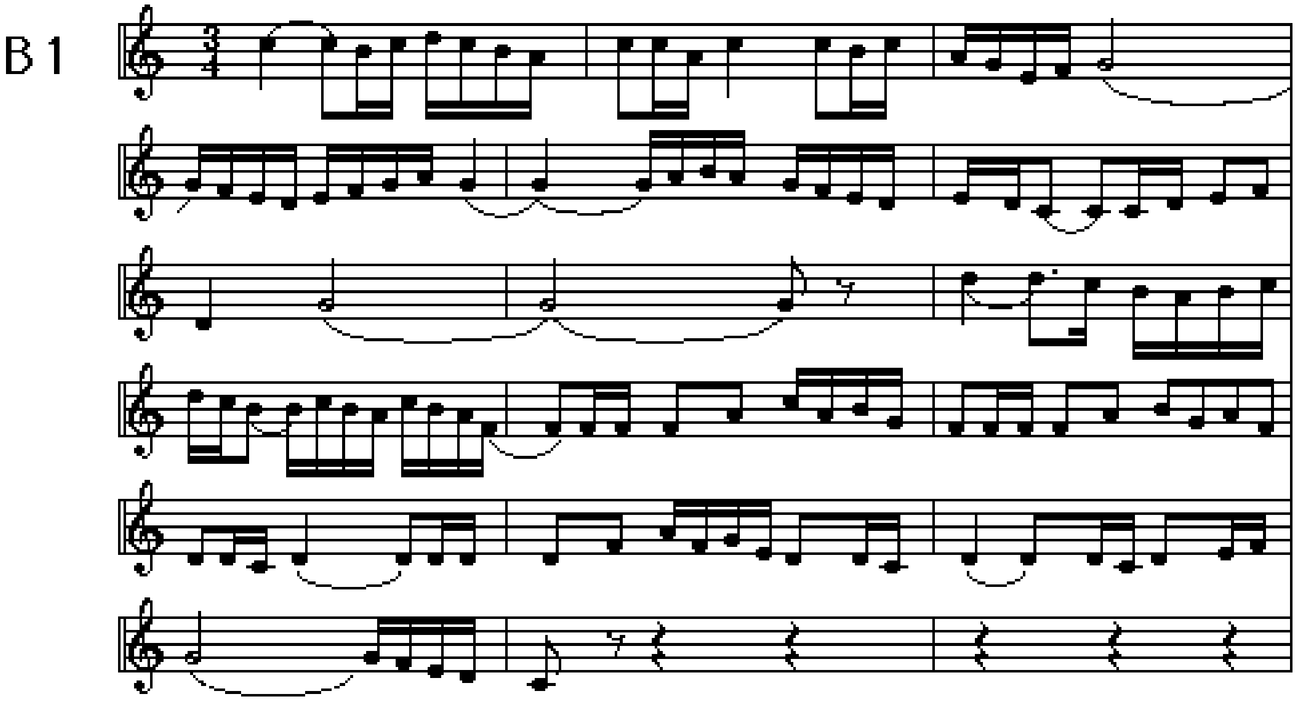

Figure 4 The major melodic theme (object **B1**).

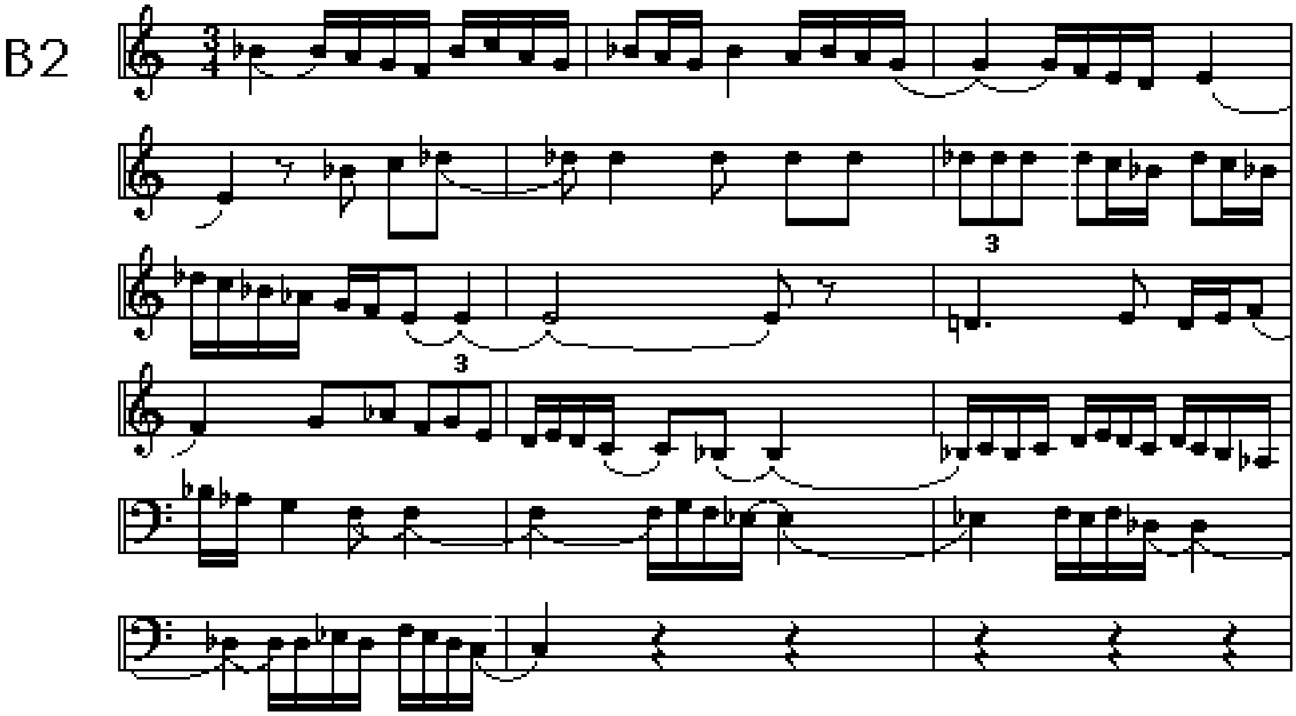

Figure 5 The minor melodic theme (object **B2**).

2.1.3. Harmonic Objects

The **D1** and **G** objects (see Fig. 6) are one-bar objects; they have either a pedal or a harmonic role all along the score; the **D1** object remarks the weak beats while the **G** object remarks the strong beat of the 3/4 bar.

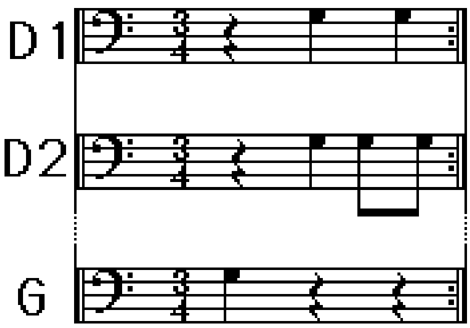

Figure 6 The harmonic objects (including **D2** which substitutes **D1** in the resolution phase).

2.1.4. Resolution Objects

The Finale of the Bolero is characterized by the appearance of some new music objects; they are the following:

B3: it is a modulating variant of the minor melodic theme **B2** (see Fig. 7);
B4: it is a new melodic theme which appears after **B3** and leads to the well known resolution of the trumpets (see Fig. 8);
Bres: it is the melodic resolution object (see Fig. 8);

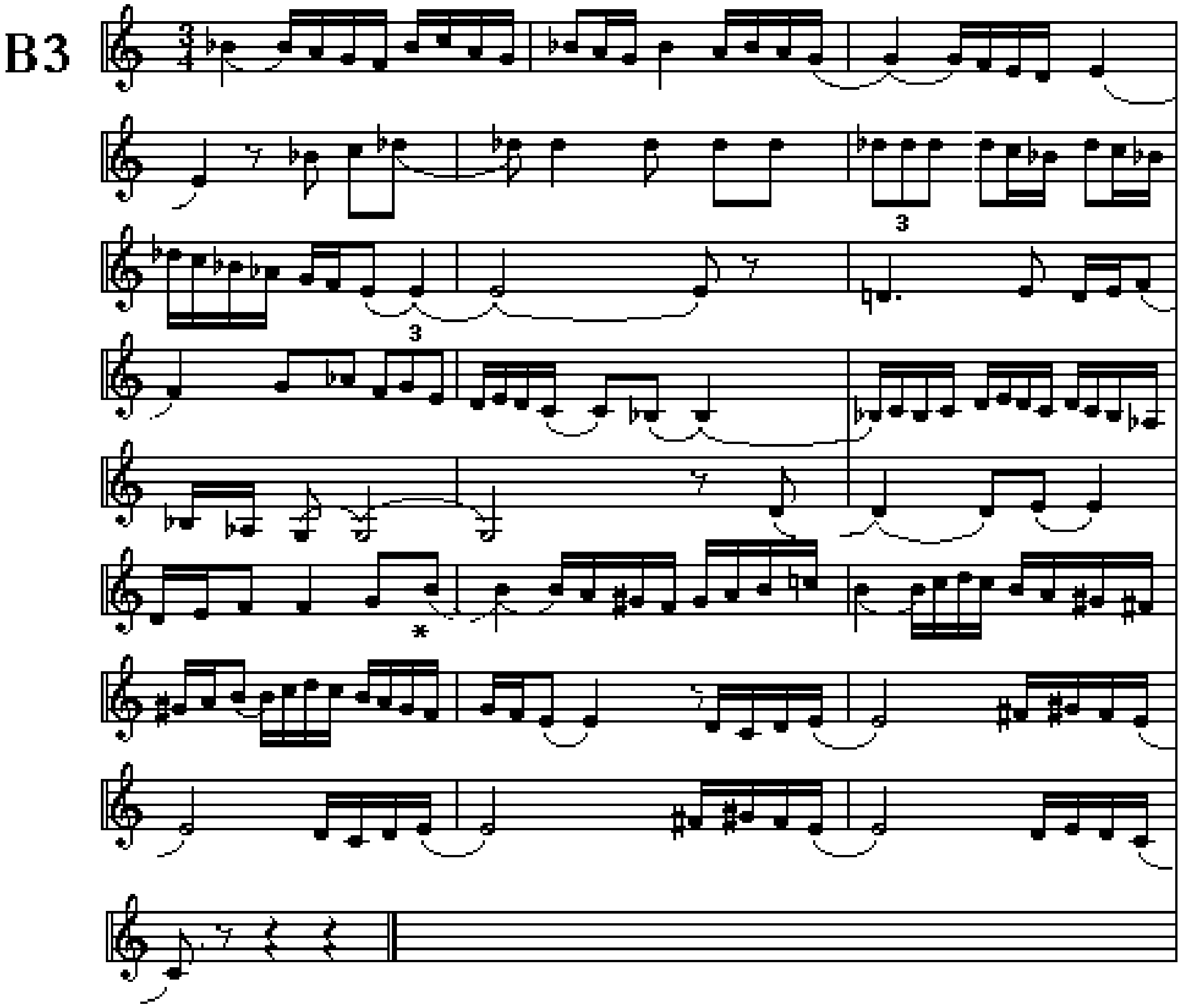

Figure 7 The modulating minor theme in the Finale.

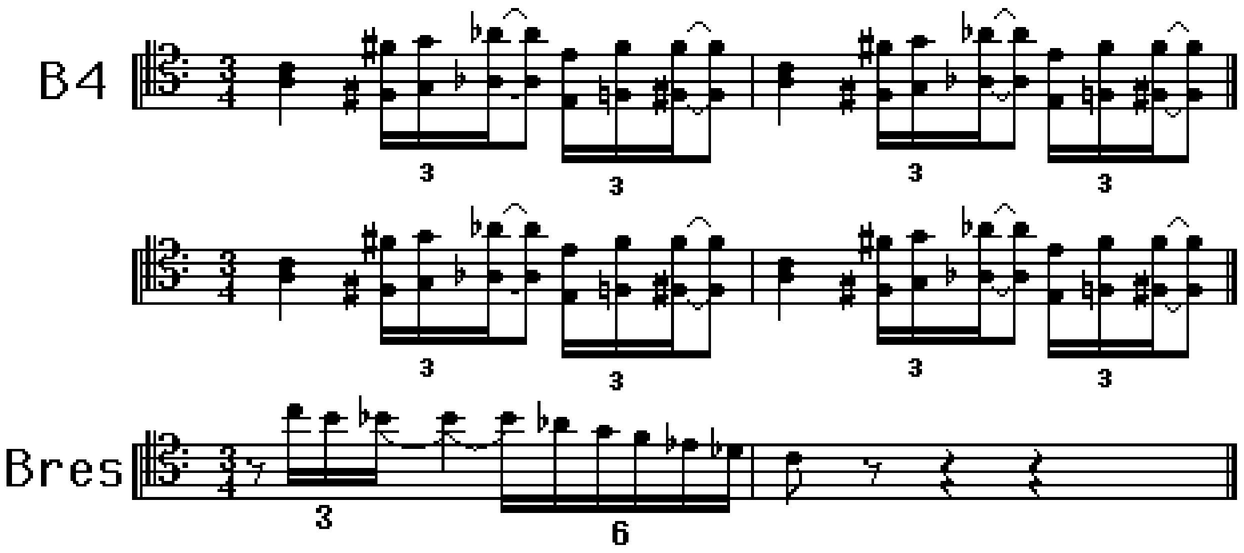

Figure 8 The resolution melodic objects **B4** and **Bres**.

A3: it is a variant of the **A1** object; it has a tremolo function which is carried out by seconds' or thirds' intervals (see Fig. 2);
Ares: it is derived from the **A** object (see Fig. 2);
D2: it is the object which substitutes **D1** in the Finale; it has a hybrid structure which plays a relevant role in the Finale development (see Fig. 6).

2.2. Meta-Structures

We think that Ravel (as most of other composers) would have in mind a model of the composition structure which can not be represented by common music notation; by example, Bolero is clearly made up of cyclic structures within which there are specific growing rules which control rhythm, dynamics, melody, harmony and timbral textures.

At the most abstract level we have a four cycles structure outlined by the melodic major and minor themes; every cycle has a **B1 B1 B2 B2** form and after four cycles there is the resolution phase with **B3**, **B4** and **Bres** themes. So, the melodic objects sequence (see Fig. 9), which determines the global structure of the Bolero score, may be represented as:

S1: **Intro** + 4 * (2 * **B1** + 2 * **B2**) + **B1** + **B3** + 2 * **B4** + **Bres**

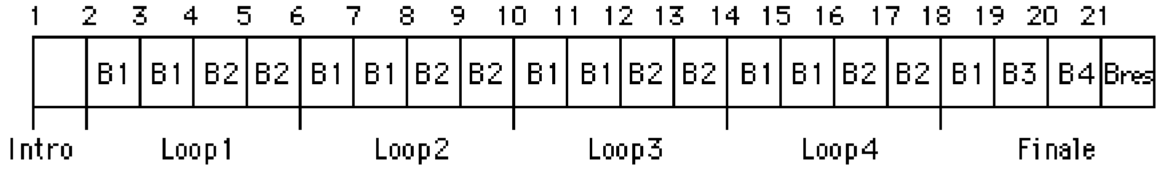

Figure 9 The melodic objects sequence.

We call **Loops** the four cyclic structures and our analysis is based on this first step of modelling; every loop lasts 72 bars (i.e. 18 * (2+2)) and is characterized by specific developments of music objects' role for the rhythmic, dynamic, harmonic and timbral texture growth, while melody is static until the resolution phase. Every loop is made up of four quarters; each quarter of a loop is 18 bars long; the first two bars of each quarter act as a bridge between two consecutive quarters.

Furthermore, together with objects of the **A**, **B** and **C** families, the Bolero score has a meta-family of music objects all of which are characterized by a one-or-two bars duration and someones of the combinations of the three quarter notes of a bar. These abstract family has a fundamental role in the global development of rhythm, harmony and timbral textures; it includes:

a) strong beat structure, i.e. object **G**;
b) weak beats structure, i.e. objects **D1** and **D2**;
c) structure with the full third beat, i.e. object **C**;
d) structure with the empty second beat, i.e. object **E**;
e) structure with the full second beat, i.e. object **F**;
f) structure of all full beats, i.e. object **Clock**;

there is only one combination which is not used by Ravel: the structure with the empty third beat; it seems to be not relevant for the rhythmic plan of the piece; as well as we have previously said that objects **F** and **Clock** are alternative to some of the others and we neglect them.

2.3. The Music Objects' Map

While common music notation is suitable for performers, we think that communication of music information among composers, musicologists, scientists and listeners may be carried out with much more satisfaction by means of structural descriptions which enhance the music objects' life within pieces of music. Here we give five music objects' maps of Bolero which describe appearances and disappearances of music objects; these maps (together with the music objects' scores; see Fig. 2-8) completely cover the information contents of the original score with the exception of duplication, transposition and harmonization of objects which will be discussed in § 2.4.

The x-axis represents time and the time unit on the map is the bar; on the y-axis there are the music objects of Bolero; their appearances in the score are described by rectangles whose length represents their duration in bars.

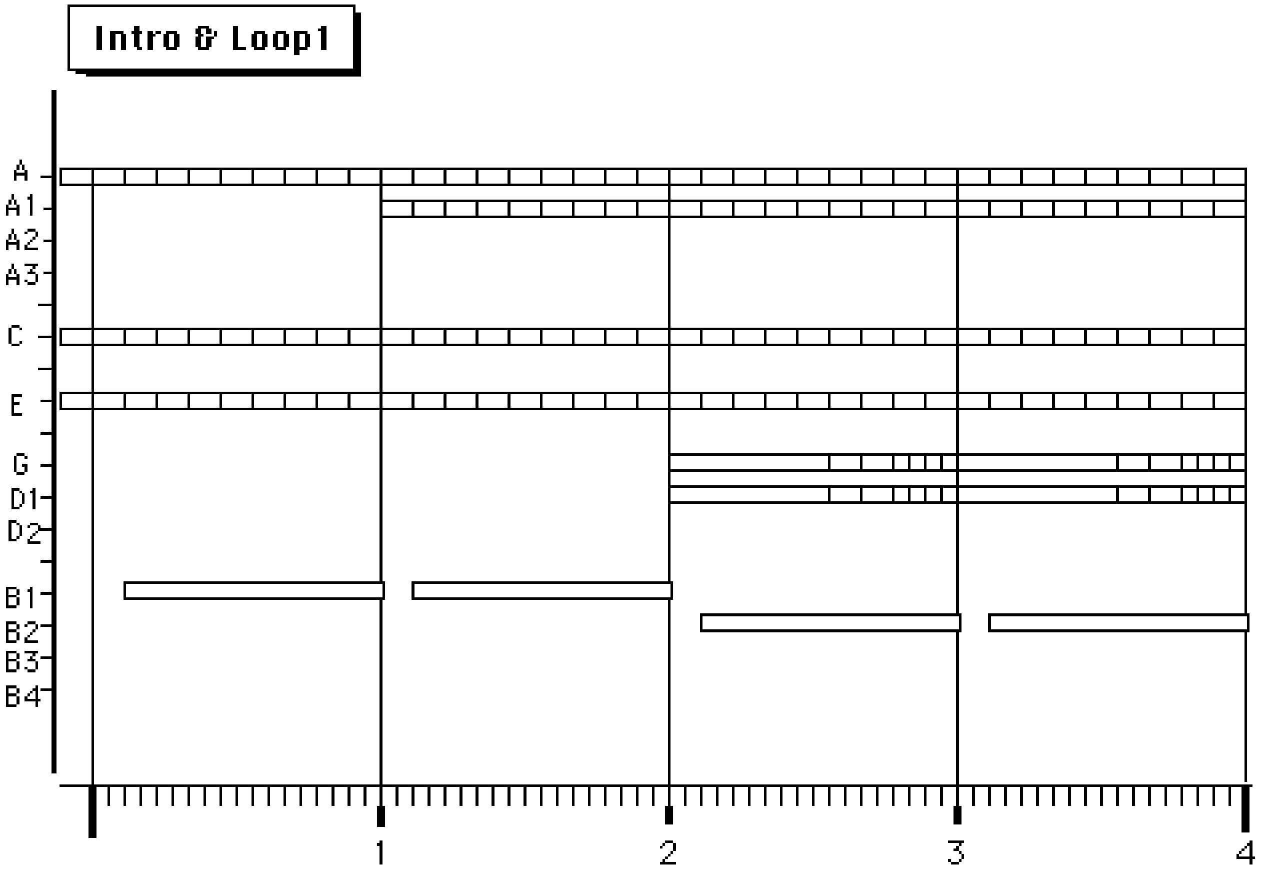

Figure 10 The music objects' map of Intro and Loop1.

The first two bars structure of the piece is a prologue which establishes rhythm; it is made up of objects **A**, **C** and **E**. Then, the properly named Loop1 begins; in the second quarter, object **A1** appears; in the third quarter, objects **G** and **D1** appear (see Fig. 10).

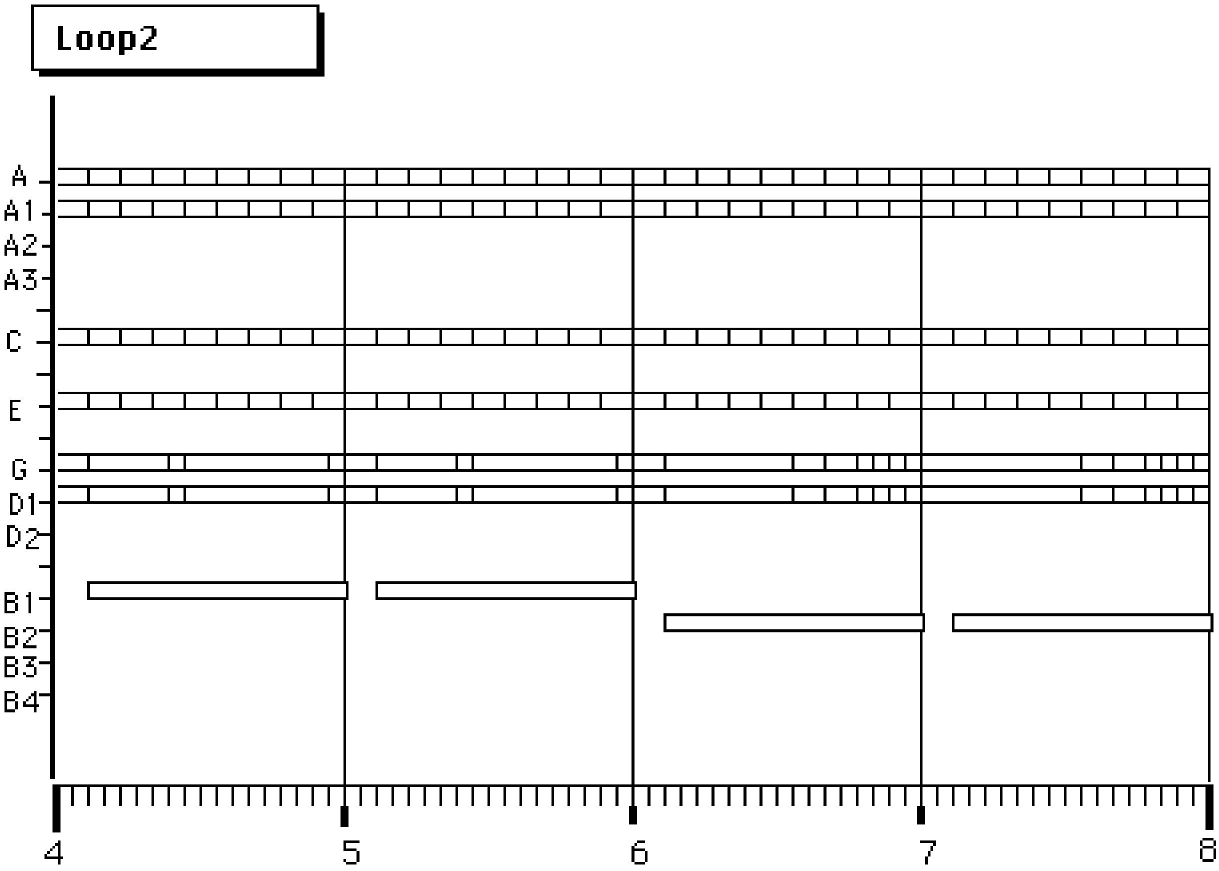

Figure 11 The music objects' map of Loop2.

Loop2 has the same set of running objects of the second half of Loop1 while their quantities and qualities keep on developing with respect to duplication, transposition and harmonization of objects (see Fig. 11).

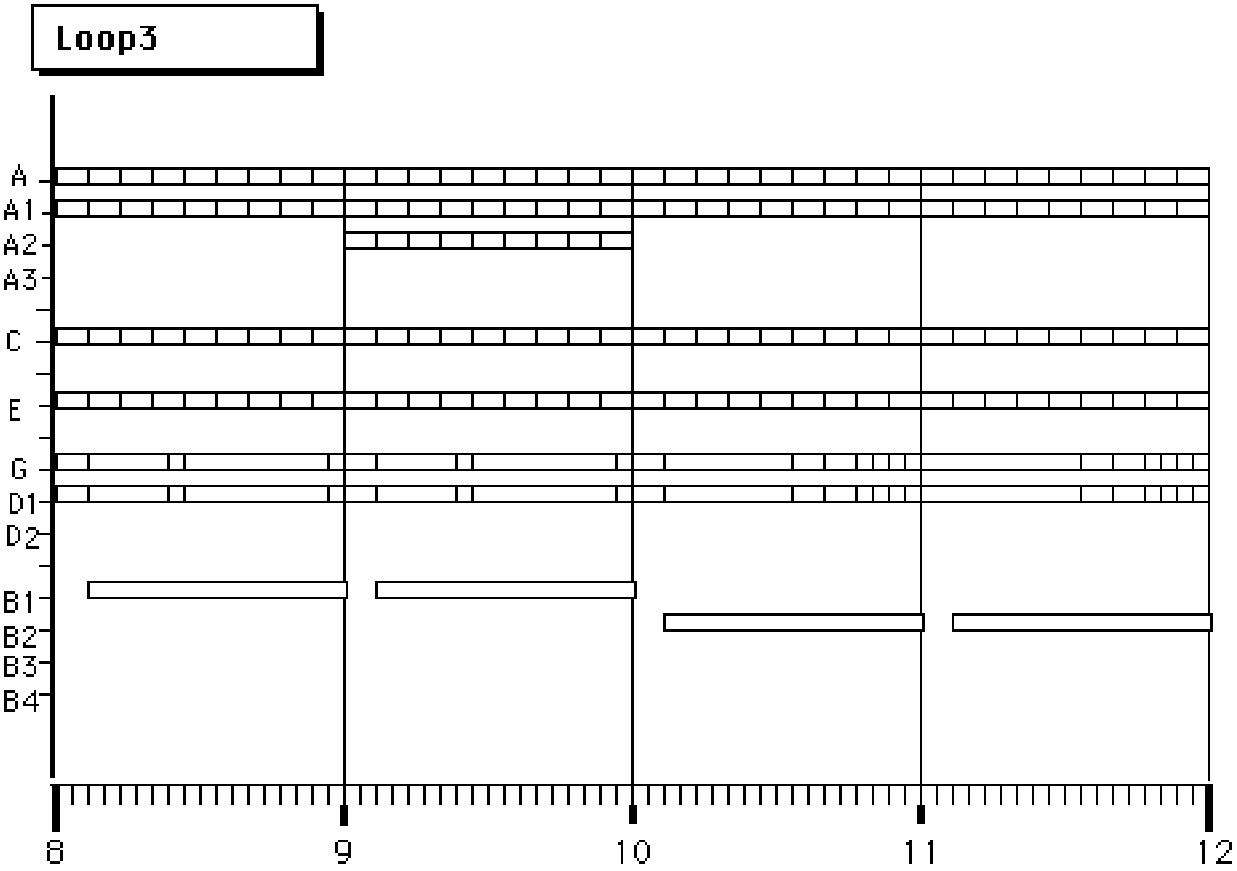

Figure 12 The music objects' map of Loop3.

In the third loop appears the arpeggiato object **A2** which is alive only during the second quarter of the loop (see Fig. 12). The quantities and qualities of music objects develop accordingly with the growing rules.

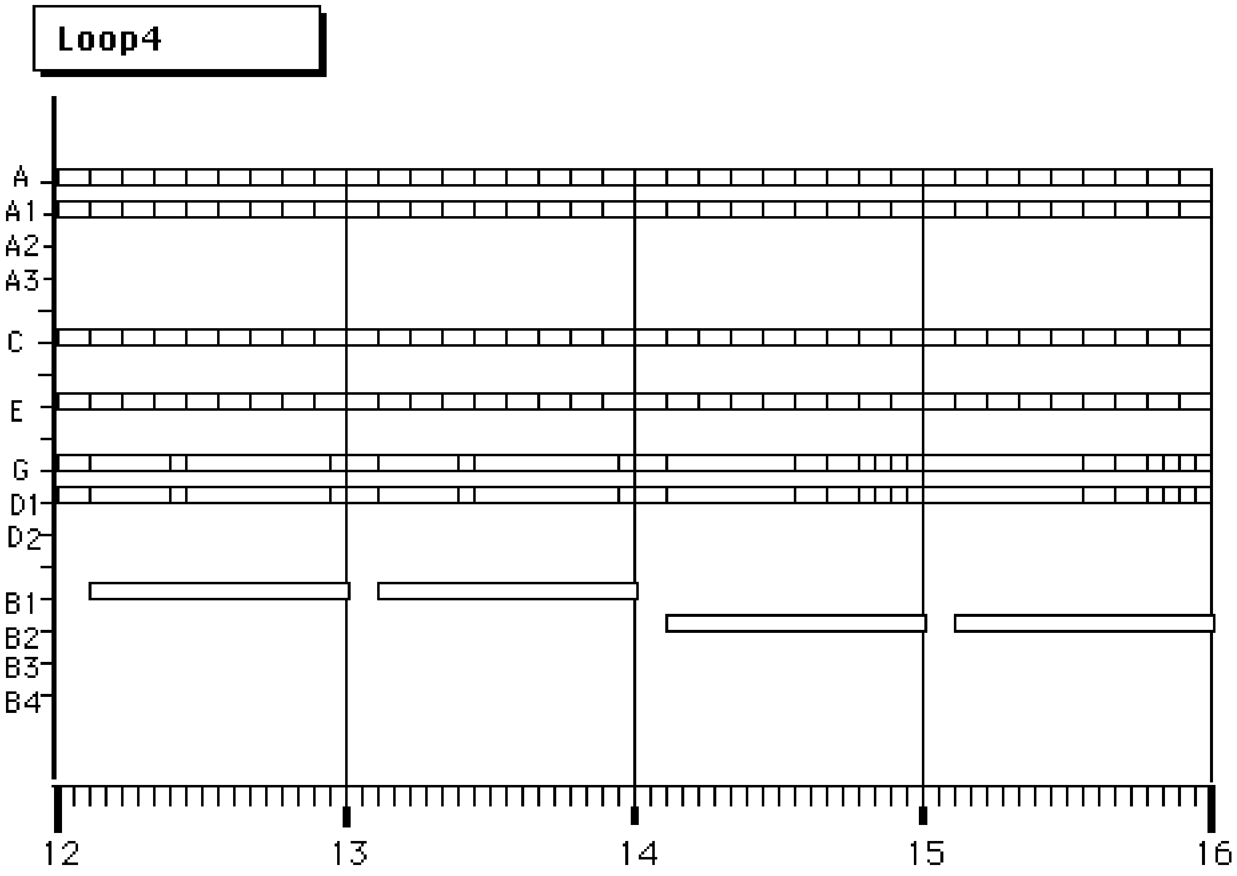

Figure 13 The music objects' map of Loop4.

Loop4 has the same music objects' map of Loop2 while their quantities and qualities keep on further developing with respect to duplication, transposition and harmonization of objects (see Fig. 13).

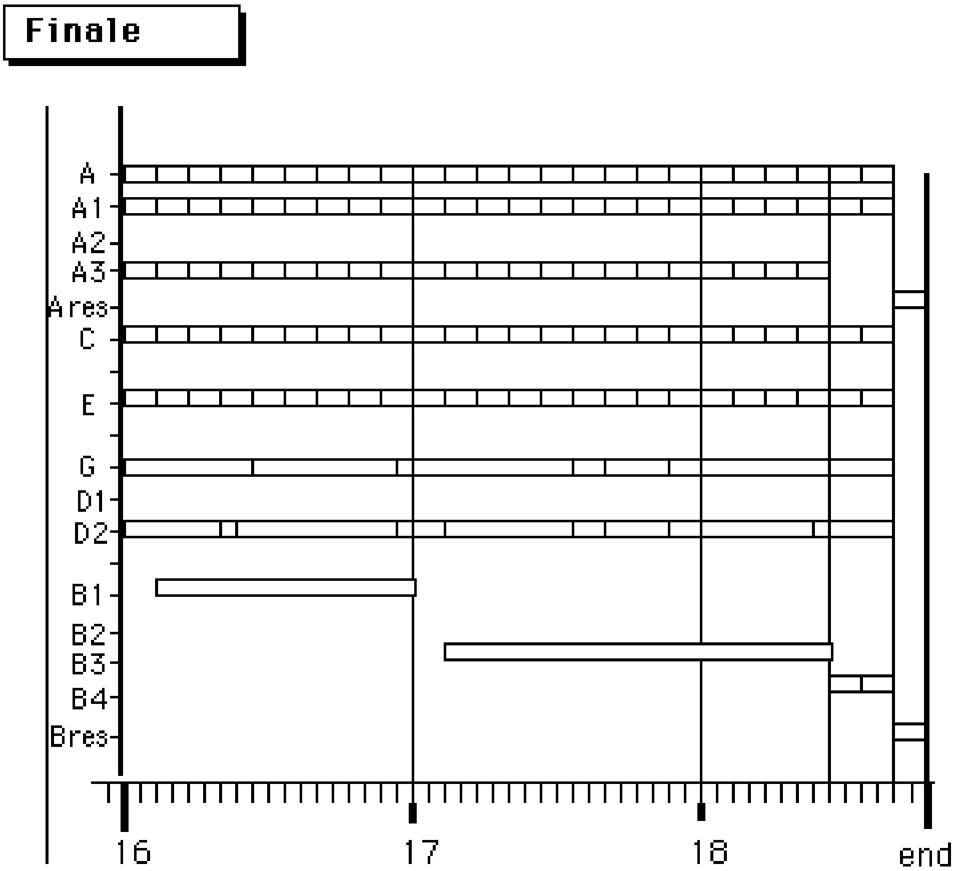

Figure 14 The music objects' map of the resolution phase.

The resolution phase of Finale is characterized by the appearance of the special rhythmic and harmonic objects **A3** and **D2**, the melodic objects **B3**, **B4** and **Bres** and particular developments of music objects.

2.4. Growing Rules

The deep structure of Bolero lies in the synergy among various development rules. The resulting effect is a general growth of dynamics, harmonization and complexity of timbral characteristics. Particularly, the odd expositions of both the major and minor themes (**B1** and **B2**) have an underlying process of timbral complexity growth, while the even expositions have an underlying process of harmonization growth.

Ravel plays by speeding up the growth of dynamics and by restraining the growth of harmonization; or viceversa. The overall effect, hearing Bolero, is the sensation that there is a general wealth within sound contents while all musical parameters are changing and the speed of the changes is always fluctuating.

Now, it may be useful to briefly discuss some concepts which are relevant for the understanding of the growing rules we have recognized within the score.

duplication of music objects : the sound obtained coupling two sounds generated by two equal instruments at both the same pitch and the same loudness gives a new sound with a loudness which is the logarithm of the sum of the two original sounds and the timbre is similar to the one of a single instrument but with the random modulation (pulsation) which is due to the interaction between the two single sound processes;

octave of music objects' instances : from the perceptive point of view, the listener cannot hear at the same loudness two phisically equal music objects which are transposed to different octaves, i.e. human beings has not a linear response to acoustical processes with respect to pitch; this kind of phenomenum is described by the well known Fletcher curves; we can roughly say that central octaves give louder sounds (or better, so they are perceived) than the lower and upper ones at the same physical level of sound;

interaction between dynamic intention and timbre : the timbre of an instrument depends on specific characteristics of the instrument itself; but is necessary to remind that the same instrument played at the same pitch gives different timbral characteristics which depends on the dynamic intention of the player; so, the role of the same instrument may be very different in the compositive model of a music piece with respect to the dynamic intention with which it has to be played;

timbral hierarchy of orchestral instruments : the traditional orchestra is usually organized by families of instruments with respect to both the graduation and the site within the whole orchestra from the physical point of view.

2.4.1. Duplication and Dynamics Rules

Even with a superficial listening the Ravel's Bolero gives the feeling of a crescendo which flows in a "ff" all-together Finale. This feeling is mainly due to the duplications of music objects rather than to the dynamics of single objects (which, anyway, are also growing). We distinguish among three different kind of duplications:

unison duplications : increase the loudness of sound and create a pulsating effect of which we have spoken in the previous paragraph;

octave duplications : increase the loudness of sound with respect to the Fletcher curves besides with the logarithm of the sum of the single sound amplitudes, enhance the harmonic relationships among timbral partials and create a pulsating effect very close to the kind of a unison duplication;

other duplications : have the same effects of the octave duplications with the addition of a new one; the new effect is due to the modulations between sound harmonic partials and sound frequencies determined by the 12 tone tempered intervals; for example: the third harmonic partial interacts with the fifth of a major/minor tonal scale, the fifth harmonic partial interacts with the major/minor third of a major/minor tonal scale, and so on.

The different kinds of duplications are used by Ravel with different aesthetic and music aims which we recognize as the followings:
a) duplications with dynamic aim;
b) duplications with timbral aim;
c) duplications with harmonic aim.

While the dynamic aim is somewhat obvious, the timbral aim is a revolutionary one at the time of the Bolero composition (1928): really, Ravel synthesizes new timbres by linear combinations of the traditional instruments' ones; the harmonic aim is close to the composition techniques of classic traditional music.

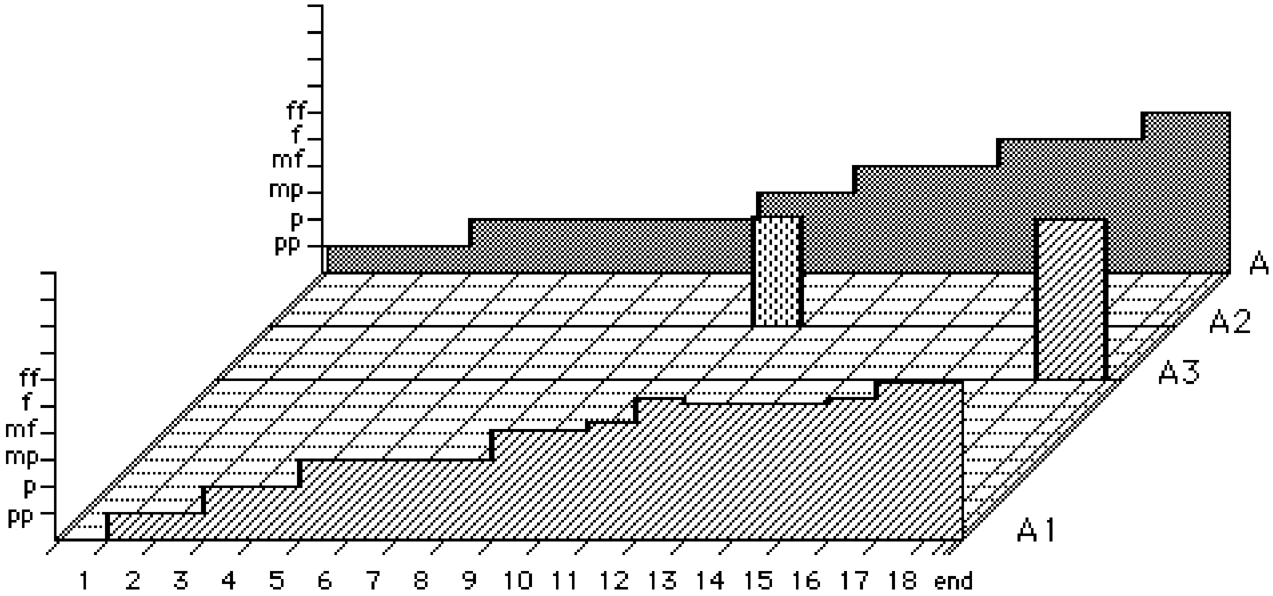

Figure 15 The dynamics growth of the **A**-family of music objects.

Figure 15 shows, for example, the dynamics growth of the A-family of music objects; all the other music objects have a similar development with respect to dynamics; in fact, no changes of dynamics parameters are made apart from the numeric score references steps. It is plain that all objects grow continuously from the beginning to the end of the piece with a very few exceptions. To soften the dynamics growth steps, Ravel increases the dynamic parameters of the various instruments out of phase one each other. So, the comprehensive effect is an almost continuous growth of dynamics.

2.4.2. Harmonic Rules

The harmonic contents of Bolero are mainly due to both **D1** and **G** music objects. Different harmonic structures are used to accompany the major and minor themes; these structures grow continuously during the four loops; in the Finale object **D1** is substituted by object **D2**.

We can define a set of tables which describe the behaviour of harmonic structures underlying melodic objects; we define now the concept of harmonic variant that is a music object with the same metrics but with some different pitch with respect to an original music object; we use the notation:

<original music object identifier>**.**<melodic object identifier> (variant number)

Object **D1** has four variants underlying **B1** which appear in the following sequence:

D1seq1:	D1.B1(1)	7 times
	D1.B1(2)	once
	D1.B1(3)	9 times
	D1.B1(4)	once

while has six variants underlying **B2** which appear in the following sequence:

D1seq2:	D1.B2(1)	10 times
	D1.B2(2)	twice
	D1.B2(1)	twice
	D1.B2(3)	once
	D1.B2(4)	once
	D1.B2(5)	once
	D1.B2(6)	once

Both sequences are 18 bars long; it is obvious because both **B1** and **B2** are 18 bars long. The harmonic process begins within Loop1 when the minor theme appear for the first time. The sequence underlying **B2** is slightly different when object **B1** disappears and object **B2** appears (in the middle of the loop structures), beginning within Loop2; the difference involves the first step of the sequence

	D1.B2(1)	10 times

which is substituted by a bridge object for the first 2 bars; so, the first step becomes:

D1seq2b:	D1.B2(0)	twice
(beginning)	D1.B2(1)	8 times

The complete harmonic process of object **D1** is then described by the following expression:

S2: 2 * **D1seq2** + 3 * (2 * **D1seq1 + D1seq2b + D1seq2**)

Object **G** has the same behaviour of **D1** during the four loops, that is: 4 variants underlying **B1** and six variants underlying **B2**, with the same sequences and with a bridge object at the beginning of the transition major/minor theme.

In the Finale phase, instead, object **G** has the following sequences; we have used the identifier **Res** as:

Res = **B1 + B3** + 2 * **B4 + Bres**

underlying **B1**:

Gseq3:	G.Res(1)	8 times
	G.Res(2)	9 times
	G.Res(1)	once

and after:

Gseq4:	G.Res(1)	10 times
	G.Res(2)	twice
	G.Res(1)	4 times
	G.Res(2)	twice
	G.Res(1)	8 times
	G.Res(1)	4 times

Therefore the complete harmonic process of object **G** is described by the following expression:

S3: 2 * **Gseq2** + 3 * (2 * **Gseq1 + Gseq2b + Gseq2**) + **Gseq3 + Gseq4**

Object **D2** has three variants, in the Finale phase, underlying **B1** which appear in the following sequence:

D2seq1:	D2.Res(1)	6 times
	D2.Res(2)	once
	D2.Res(3)	10 times
	D2.Res(2)	once

and other six variants after **B1**:

D2seq2:	D2.Res(1)	twice
	D2.Res(4)	8 times
	D2.Res(3)	twice

D2.Res(4)	4 times
D2.Res(3)	twice
D2.Res(5)	7 times
D2.Res(6)	once
D2.Res(1)	4 times

2.4.3. Timbral Textures

Ravel calls his Bolero an "instrumentation essay *without music* "; really, his composition is very rich from the point of view of orchestral textures. Ravel bases his technique on three basic concepts:

a) a complex timbre may be obtained by the duplication of many simple timbres;
b) a complex timbre may be obtained by the superimposition of many simple timbres with harmonic relationships among them (third, fifth, octave, ecc.);
c) a complex timbre (like the previous described ones) may be transformed varying the relative dynamics developments of the duplicated and/or superimposed simple timbres.

In our research we have been very interested in Ravel's instrumentation technique but we neglect these aspects here because we have chosen to develop a formal model in which the timbre aspect is described at the symbolic level; in other words, a timbre is described by means of a symbol, both simple and complex timbres; see § 3.4.1. where we define the model conventions.

3. The Formal Model of Ravel's Bolero

We have already spoken about the musicological analysis we have done and the results we have had which are the basis of the formalizing process. Now we describe the Bolero score by means of a formal model based on Petri Nets and a music algebra as the describing tools. In the following paragraphs we show how it is possible to represent music information starting from the metaphor of music objects and their transformations.

3.1. Working Hypotheses on the "Ideal" Bolero

In the beginning of the formalization process we have defined some working hypotheses which have to act as guidelines of our modelling work; it is obvious that many kinds of formal descriptions of the Bolero score may be done, so our approach is only a particular one; anyway, the hypotheses that now we will discuss is very close to a "synthetic" approach like those of computer musicians, but not as "a priori" approach; it is the natural consequence of our analytic results.

We state the following hypotheses:

H1: all the analytic process has given a lot of music informations; most of them are in the form of data tables; a modelling aim is to infer algorithms from data tables, so that the model could be parametrically controlled;

H2: the datatables-to-algorithm inference requires the recognition of exceptions and their handling (by example, normalization i.e. approximation);

H3: harmonic developments may be described and processed by rules;

H4: timbral developments may be described and processed by means of additive synthesis technique;

H5: dynamic developments may be described both directly and as the result of the concurrent effects of **H3** and **H4**;

H6: the formal model we develop allow to synthesize scores by parametrical controlling of the model i.e. varying music objects and/or their transformations; the new scores are very close to the original Bolero score at the structural level of representation; this hypothesis will be discussed in § 4.

3.2. Hierarchically Organized Levels of Representation

As a further guideline for our formalization process, we choice to develop a model which satisfies the following general criteria:

C1: the model is multilayered so that every abstraction level has its own representation level;

C2: the model has high graphical features so that its understanding may be made easier and evocative;

C3: the model has no complex information structures so that their description and processing are made easier;

C4: the model has a non-optimized formal representation i.e. the minimum number of information structures which completely describe the score; it is preferable to have an information structure for every concept which is interesting to represent rather than an optimized model without a biunivocal correspondence between information structures and concepts.

3.3. Petri Nets as the Music Description Formal Tool

It is time to introduce the formal tools we have used; let us begin with a brief summary of syntactic and semantic notions.

Since 1980 we are experimenting Petri Nets (PNs) as the basic tool for music description and processing [2] [3]; other experiments have been carried on by S. Pope [4] based on PrT Nets [5]. Although PNs are not a specific music tool but a general purpose tool [6] [7] [8] [9] [10] [11], PNs are well accepted by the musicians who have experimented our PNs-based methodologies and tools [12].

A PN is usually defined by a quadruple **<P**, **T**, **A**, **M>**, where **P** is the set of places, **T** is the set of transitions, **A** is the set of arcs connecting both places to transitions and transitions to places, **M** is the marking of the net. Our implementation of Petri Nets extends the "classic" definition, so that the musician can build up hierarchical music models defining music objects and music algorithms made up of functional expressions. In this way, a Petri Net is defined by means of the triple **<P**, **T**, **A>** and the following three statements:
N1: **P ∋ p** ≡ **<identifier**, **tokens**, **capacity**, **MIDI channel**, **object**, **play**, **file >**
N2: **T ∋ t** ≡ **<identifier**, **algorithm >**
N3: **A ∋ a** ≡ **<node-from**, **node-to**, **multiplicity >**
where:
P is the set of places; **T** is the set of transitions; **A** is the set of arcs; **identifier** is the label of the node (both place and transition); **tokens** is the number of tokens within the place, that is the value of the marking function **M** (see below); **capacity** is the maximum number of tokens allowed for that place; **MIDI channel** is the number of the MIDI channel where the music object, if there exists, has to be played; **object** defines whether a music object may be associated to the place or not; **play** defines whether the object has to be played or not; **file** defines whether the music object associated to the place, if there exists, is stored on file or not; **algorithm** defines whether the algorithm associated to the transition, if there exists, has to be executed or not; **node-from** is the identifier of the node from which starts the arc; **node-to** is the identifier of the node to which ends the arc; **multiplicity** is the numeric label of the arc. If a place has an associated music object which is stored on file, its **identifier** represents both the label of the node and the file identifier.

P, **T** and **A** are finite sets; **P** cannot be empty. Both objects and algorithms have their own syntax definitions.

The marking of a a net is the distribution of token into the place nodes; their number and distribution may change during net execution. A marking function is defined as the following array:

N4: $\mathbf{M} \equiv < \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_i, \ldots, \mathbf{m}_n >$

with

N5: $\mathbf{N} \ni \mathbf{m}_i = \mathbf{M}(\mathbf{p}_i)$ with $\mathbf{P} \ni \mathbf{p}_i$ for every $\mathbf{m}_i$

where **n** is the number of places within the net and **N** is the set of positive integers.

The execution of a net is made up of transition firings. A transition may fire if each one of the input places, i.e. places which are connected with oriented arcs to the transition, has at least one token. The transition firing has two effects: to decrement the marking of each input place by one token and to increment the marking of each output place, i.e. a place which is connected with an oriented arc from the transition, by one token.

After the starting of a net, firings follow one another until there are no more transitions which may fire. At the end of transition firings, the execution of the net stops.

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

N6: $\mathbf{S} \ni \mathbf{s} \equiv$ **<identifier**, **node-begin**, **node-end >**

which is the set of hierarchy links,where:

identifier is the label of the net, **node-begin** is the identifier of the beginning node for the morphism application, **node-end** is the identifier of the ending node for the morphism application.

The **S** set represents all the information about the hierarchy of the model, while the **P**, **T**, **A** sets completely describe a particular net of the model; in other words, a model is defined by an **S** set and as many triple **<P**, **T**, **A>** as the number of the nets within the model.

In order that music PNs be as simple and terse as it is possible, we use a particular kind of PNs which includes some well-known extension of Place-Transition (PT) nets: *structured* nets [13], arcs with *multiplicity* which define a subset of *self-modifying* nets [14], places with *capacity* [15] together with a special music algebra to define algorithms associated to transitions; all PNs shown in this work are made by the *SCORESYNTH* program [16] [17] [18], which is an editor/executor of extended PNs.

We have chosen a particular kind of morphisms which seems to be suitable for our purposes: the refinement morphisms; the hierarchical approach introduced by Kotov is rather close to our implementation. Refinements allow to define very complex PNs models by means of simple PNs and hierarchical structures, i.e. allow to design models by either a top-down or a bottom-up approach. A refinement, that we call a *subnet*, is a PN which gives a more detailed description of a node (either a place or a transition) of the upper level of abstraction. We call that node the "father" node and that subnet the "daughter" net. The restriction we have defined on the structure of subnets (i.e. nets of lower levels) to implement hierarchy is determined by the one-begin-node/one-end-node morphism rule. As we can see below, we have choiced to use only places as the hierarchical nodes of the model of Bolero.

Capacity is a feature of place nodes which defines the maximum number of tokens allowed for each place. Particularly, if we have all the places of a net with capacities equal to one, then we have a condition-event net; on the other side, if we don't have any limitation to the marking of place nodes, then we have place-transition nets; so, our choice to introduce the capacity feature is aimed to the maximum flexibility during models definition. The capacity feature implies that the firing rule has to be modified; conditions of transition firings doesn't change with respect to the input places while transitions cannot fire if one output place, at least, is marked with as many tokens as its capacity value; in fact, if transition fires the marking of that place will exceed its capacity.

The multiplicity extension allows to add numerical labels on the arcs. Also the multiplicity feature requires modifications to the firing rule: a transition may fire if each one of the input places has at least as many tokens as the numerical label on the arc connecting the place to the transition; the transition firing has now the following effects: it decrements the marking of each input place by as many tokens as the numerical label on the arc connecting the place to the transition and increments the marking of each output place by as many tokens as the numerical label on the arc connecting the transition to the place.

Within a PN, we connect the concept of place either with observable musical objects (which are represented by Standard MIDI Files 1.0 music objects associated to places) or control objects, i.e. objects which don't have a direct musical meaning; on the other hand, we connect the concept of transition with processes (transformation rules); both places and transitions may be used to represent the model hierarchy (representation of lower-ranking PNs). The complete syntax definition of both music objects and the related music algebra is described in [18]. Here we give the BNF of both music objects and algorithms.

Objects:

```
< MusicObject >      ::=    < MusicObject > . < MetaEvent > |
                            < MusicObject > < MIDIEvent > |
                            . < MetaEvent > |
                            < MIDIEvent >
< MetaEvent >        ::=    Tempo < integer > |
                            Base < integer > |
                            Signature < integer > < integer > |
                            Rest < integer > |
                            MidiEx < MIDIExBytes >
< MIDIExBytes >      ::=    " ...any sequence of bytes, in hex, according to MIDI..."
< MIDIEvent >        ::=    < Header > Pgm  < integer > |
                            < Header > After  < integer > |
                            < Header > Polyp  < integer > |
                            < Header > Ctrl  < integer > < integer > |
                            < Header > Bend < integer > |
                            < Header > < NoteEvent >
< Header >           ::=    < Channel > < Quarter > < Offset >
< NoteEvent >        ::=    < Duration > < Pitch > < KeyVel >
< Pitch >            ::=    < Note+Mod > < Octave > | < KeyNumber >
< Note+Mod >         ::=    < Note > | < Note > #
< Note >             ::=    A | B | C | D | E | F | G
< Channel >          ::=    1 | ... | 16
< Quarter >          ::=    < positive integer > {...depends on song length}
< Offset >           ::=    < positive integer > {...depends on .Base}
< Duration >         ::=    < positive integer >
< KeyVel >           ::=    0 | ... | 127
< Octave >           ::=    -2 | ... | 8
< KeyNumber >        ::=    0 | ... | 127
```

Algorithms:

```
< Algorithm >        ::=    < Algorithm > < Operator > | < Operator >
< Operator >         ::=    < QualOp1 > :  < RangeExpression > ,  < RangeExpression > |
                            < QualOp2 > :  < RangeExpression > ,  < RangeExpression > ,
                            < OpExpression > |
                            < QualOp3 > < TonalExpression > :  < RangeExpression >,
                            < RangeExpression > , < OpExpression > |
                            < QualOp3 > < TonalExpression > :  < RangeExpression >,
                            < RangeExpression > , { < MusicObjectId > , < RangeExpression >} ,
                            < OpExpression > |
                            < QualOp4 > :  < RangeExpression > , < RangeExpression > ,
                            < OpExpression > |
                            < QualOp4 > :  < RangeExpression > ,  < RangeExpression > ,
                            { < MusicObjectId > ,  < RangeExpression >} , < OpExpression >
< QualOp1 >          ::=    I | K | S
< QualOp2 >          ::=    M | R
< QualOp3 >          ::=    P
< QualOp4 >          ::=    C | D | L
<MusicObjectId >     ::=    "... alphanumeric string ..."
< TonalExpression >  ::=    [ < Note > < Tonality > ]
< Tonality >         ::=    m | - | t | p
< RangeExpression >  ::=    < RangeExpression > + < RangeTerm > |
                            < RangeExpression > - < RangeTerm > |
                            < RangeTerm >
< OpExpression >     ::=    < OpExpression > + < OpTerm > |
                            < OpExpression > - < OpTerm > |
                            < OpTerm >
< RangeTerm >        ::=    < RangeTerm > * < RangeFactor > |
                            < RangeTerm > / < RangeFactor > |
                            < RangeFactor >
< OpTerm >           ::=    < OpTerm > * < OpFactor > |
                            < OpTerm > / < OpFactor > |
                            < OpFactor >
< RangeFactor >      ::=    < integer > |
                            < MetaChar >|
                            ( < RangeExpression > )
< OpFactor >         ::=    < Pitch > |
                            < integer > |
                            < MetaChar >|
                            ( < OpExpression > )
< MetaChar > ::=     $ | ! | % | ?
< Pitch >            ::=    < Note+Mod > < Octave > | < KeyNumber >
< Note+Mod >         ::=    < Note > | < Note > #
< Note >             ::=    A | B | C | D | E | F | G
< Octave >           ::=    -2 | ... | 8
< KeyNumber >        ::=    0 | ... | 127
```

We mention the meaning of each terminal character that identifies the kind of operator (QualOp):

C (Channel) = operator on channels;
D (Duration) = operator on durations;
L (Loudness) = operator on intensities;
M (Multiply) = operator for multiplying notes;
R (Rotate) = operator for rotating notes;
P (Pitch) = operator on pitches;
I (Inversion) = operator for inverting the ordering of notes;
K (Kill) = operator for deleting notes;
S (Save) = operator for preserving notes.

Furthermore, when we are designing a net model, we often have to create many similar nets, which only differ for place attributes and algorithms, but which are identical as their structures. So, we have introduced a special extension to PNs that we call *macro* nets. Infact, we can use one only net as the base model and then modify its attributes and/or algorithms at our need. So, it is also possible for the user to create his own libraries, with some recurrent net models he uses.

The way we can do this is by writing, for every subnet place or subnet transition we want, a modifier list for the net which is loaded into memory and actualized before the execution of the whole expanded net. The model we have defined makes and extensive use of the macro net feature.

The BNFs listed below define the syntax of modifier lists:

```
< ModifierList >      ::=   < ModifierList > < Modifier> | <Modifier>
< Modifier>           ::=   < PlaceModifier > | < TransModifier > | < ArcModifier >

< PlaceModifier >     ::=   < PlaceId > : < PlaceOpList >
< PlaceOpList >       ::=   < PlaceOpList > < PlaceOp > | < Place Op >
< PlaceOp >           ::=   < PlaceAttribute > | < PlaceSetting >
< PlaceAttribute >    ::=   Play | Mute | File | NotFile | Obj | NotObj | Simple | Subnet
< PlaceSetting >      ::=   < NewId > |
                            C = < Capacity > |
                            M = < Token > |
                            CH = < Channel > |
                            { < MusicObject > }

< TransModifier >     ::=   < TransId > : < TransOpList >
< TransOpList >       ::=   < TransOpList > < TransOp > | < TransOp >
< TransOp >           ::=   < TransAttribute > | < TransSetting >
< TransAttribute >    ::=   Enabled | Disabled
< TransSetting >      ::=   < NewId > |
                            { < Algorithm > }

< ArcModifier >       ::=   < PTArcOp > | < TPArcOp >
< PTArcOp > ::=       < PlaceId > -> < TransId > = < Multiplicity >
< TPArcOp > ::=       < TransId > -> < PlaceId > = < Multiplicity >
< Multiplicity >      ::=   1 | ... | 240
```

Petri nets are commonly represented by oriented bipartite multigraphs; we too use the graphic form of representation with the following set of icons:

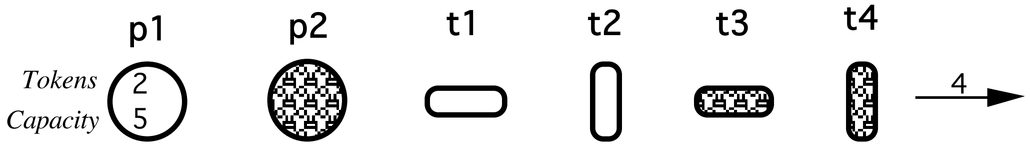

Figure 16 Petri nets icons.

The above icons represent (from left to right): a named simple place with tokens and capacity attributes, a named place which represents a subnet, a horizontal named simple transition, a vertical named simple transition, a horizontal named transition which represents a subnet, a vertical named transition which represents a subnet and, at last, an arc with the multiplicity attribute set to four.

The definition of music objects, place attributes, algorithms and modifier lists is made by suitable editing windows of the SCORESYNTH program.

The modelling of music by our implementation of PNs makes possible the description of the *structural* level of music [19] as a multilevel environment within which musical objects may flow concurrently and interactively; the more detailed levels of representation (symbolic, physic, operating, sample levels) may be described and/or synthesized depending on the individual modelizations choices of musicians using PNs.

This approach extends the notion of musical process [20] by the introduction of concurrent process, without time explicit description as in [21]. PNs are particularly suitable to describe concurrent processes and to control their synchronization or not. When a transition may fire, we don't know when it will fire; and the duration of firing is assumed to be instantaneous. When we use PNs to describe asynchronous models we have no problem with this feature. There are no measurements of time within Petri Nets and every event has an indetermined duration: it is the behaviour of the net that implicitly determines timings. On the other hand we can synchronize events simply by the structuring of suitable nodes' connections.

Another feature of PNs is their asynchronous nature; in fact, there is no clock within a PN and all the events have an undetermined duration, i.e. events' durations cannot be determined. It is the structure of the net which determines the firings' sequence, without respect for their durations.

At last, the firings' sequence changes among different executions of the net; so there is no univocity between a net and its firings' sequence. We can have many transitions qualified for firing at the same time and we cannot know which one will fire as the first: the net does not describe this kind of information. So, every execution of a PNs model may give a different firings' sequence.

This feature may seem to be too abstract for the modelling of timed complex processes as the music ones; different approaches are known about how to describe events' durations within PNs; most of them represent events by means of timed transitions. Our approach represent timed events by places and their transformations by transitions. The firing of a transition has a null duration. In this way, when a token is put into a place with an associated event the token can not be considered for the firing of transitions connected to the place until the associated event is ended.

We can represent the same musical information by PNs at a lower or higher level of abstraction by means of specific modelling approaches. Thus, we can make a top-down description, as suggested in [22]; we might also realize a bottom-up description as attempted in [23]. The top-down approach is closer to the synthesis of musical objects (musical composition, syntax definition, sound synthesis, ecc.) while the bottom-up approach is closer to the analysis of musical objects (musicological analysis, spectral analysis, compacting, ecc.).