PARADIGMS FOR THE HIGH-LEVEL MUSICAL CONTROL OF DIGITAL SIGNAL PROCESSING

Marco Stroppa

Hochschule für Musik und Darstellende Kunst
Stuttgart, Germany
stroppa@mh-stuttgart.de

ABSTRACT

No matter how complex DSP algorithms are and how rich sonic processes they produce, the issue of their control immediately arises when they are used by musicians, independently on their knowledge of the underlying mathematics or their degree of familiarity with the design of digital instruments. This text will analyze the problem of the control of DSP modules from a compositional standpoint. An implementation of some paradigms in a Lisp-based environment (*omChroma*) will also be concisely discussed.

1. LACK OF GENERALIZED ABSTRACTIONS

Although many ways of producing sonic processes by means of computers have already been • devised and abundantly investigated, little work has been done so far to search for musically-relevant control models independent on a composer's personal view. A basic definition of "control" is not difficult to find (see 1.1); however, when closely perused, it turns out to be quite a thorny, possibly endless issue.

The recent development of gestural interfaces and of real-time hardware has not solved the problem either, but has only displaced it from one (textual) to another type of interface. Moreover, for reasons of computational efficiency the expressive power of real-time devices is still quite poor in comparison with a non real-time approach [1].

In addition, whoever used a computer over a period of time long enough probably underwent the excruciating experience of having to express identical concepts in different grammatical flavors as new environments became available and the old ones no longer worked. Not only is the time spent in porting the same system onto another platform wasted from a compositional standpoint, but it also reveals that a serious problem of abstraction still subsists.

1.1. The issue of the musical control of DSP modules

To state it as plainly as possible, controlling DSP modules means devising appropriate abstractions to deal with large amounts of data sent to banks of sound-generating patches.

Such *patches* are collections of DSP modules[1] whose main function is to produce sound. A *bank* is a group of functionally identical patches that differ only by their input data. In this text, we will assume that a way to generate a patch is always available and will concentrate on control data. We will also presuppose that such data have a compositional purpose, that is they are written and composed, as is a score for acoustic instruments. We will not directly tackle real-time gestural controls or improvisation, even though many questions are similar and the two approaches could be combined.

When the goal of generating sound with a computer is not only of simulating a pre-existing acoustic model, but also of providing an "esthetic experience",[2] the only person able to express a final judgment of quality is the musician him- or herself. This might seem a banal tautology, but it is precisely this "incursion" of the musician's "Weltanschauung" into an apparently technical issue that makes it arduous to solve and enticing to investigate.

1.2. A first example

When the musical task is to produce a single, unique sound, there are usually many equivalent solutions. For example, to generate Jean-Claude Risset's first bell sound[3] using a synthesizer of the Music X family,[4] the following implementations produce strictly identical sounds when fed with the same data:[5]

a) wave-table synthesis

The control paradigm consists in one single oscillator, whose audio table is made of 9 very high harmonics (scalers: 56, 92, 119, 170, 200, 274, 300, 376, 407) of a sub-audio fundamental frequency (4 Hz) with different relative amplitudes (0.36, 0.36, 1,

[1] In whatever environment and sound-generating paradigm one might think of. For instance, a subtractive synthesis patch is a collection of filters (and probably of other modules as well), a patch using physical modeling might be seen as a collection of connected vibrating units, and so on.

[2] Which embodies a certain compositional idea, apart from whether it is judged as being "good" or not by the community of listeners. This statement ought to be further argued and is used here only to highlight the importance of taking into account the musician's perspective when delving into the issue of sound control.

[3] Catalogue's n. 430, three successive approximations of a bell sound [2].

[4] Such as Csound, Music V, Common Lisp Music, SuperCollider, etc.

[5] To eliminate possible differences in the maximum amplitude, the synthesis should be floating-point and the final sounds rescaled to the same value.

0.62, 0.55, 0.05, 0.05, 0.038, 0.05).[1] It is the most efficient and most constraining solution: all the harmonics will have the same duration and amplitude profile. If a vibrato needs to be applied[2], they will all vibrate at the same rate and with the same interval. However, sounds with an arbitrary amount of harmonics can be generated in a very straightforward way by simply changing the parameters of the table.

b) synthesis bank in the patch

The strategy is to write a patch containing simple sine-tone generators added together and individually controlled[3]. The computation is less efficient, but the main drawback is that the maximum number of synthesis modules is fixed[4] within the patch. On the other hand, this patch can be easily modified to include, for instance, separate amplitude envelopes or vibrato modules for each harmonic or groups of harmonics, which will however still have the same duration. This implementation is a reasonable compromise between efficiency and flexibility and is often the only possible one when using real-time hardware[5]. Different versions of this control paradigm were used by Stephen Mc Adams and his collaborators when testing the importance of common patterns of vibrato as a means of fusing or separating simultaneous sound sources[6] [3].

c) synthesis bank in the control data

This implementation still deals with banks, but the patch contains only one single sine-tone oscillator. The bank is entirely controlled from the data: each time a new harmonic is needed, a copy of the patch is dynamically allocated. The mechanism for adding up all the instanciated harmonics has to be provided by the synthesizer. In the Music X family it also comes with other control primitives, such as temporal information (starting time and duration of each harmonic) and an automatic time sorting of the instructions[7]. This is both the least efficient and most flexible solution: each harmonic has an independent amplitude envelope, duration and starting time. However, other control paradigms, such as, for instance, grouping harmonics and giving them an identical random amplitude[8] would be quite clumsy to implement.

[1] The chosen synthesizer should of course provide primitives implementing this abstraction. Although not used by Risset in this specific example, this approach is consistent with the composer's control models and was adopted in other examples of the catalogue.
[2] Via a simple modification of the patch.
[3] In this case the control data will look like couples of values, frequency / amplitude (i.e. 224, 0.36; 368, 0.36; etc.).
[4] One can always use fewer modules, by setting, for instance, the amplitude to 0 when a module is not needed, but this solution is rather awkward.
[5] Where the size of the patch will probably correspond to the maximum allowed by the hardware.
[6] I studied these Music-10 patches at IRCAM in 1982.
[7] In this example the temporal information is only used to indicate the total duration (see the control data in chapter 2.1).
[8] A typical paradigm to be performed at the level of a patch and not of control data.

1.3. Sonic potential

It is however very unlikely that a musician generate a "unique" sound. Whether to improve the quality[9] or to generate several sounds sharing certain common features, she or he will have to cope with processes of sonic development, rather than with individual sounds. These are multitudes of sound processes which both implement and develop[10] an original "idea"[11]. In this framework, therefore, a sound-generating patch must be considered in terms of its *sonic potential*, [4] that is of all the classes of the sonic material that it is able to generate, an infinite quantity, although probably only a limited amount will satisfy the musician's requirements.

From this perspective, the implementations above are no longer sonically equivalent. In our carefully chosen example, they produce the same acoustical result, but since they represent it in different ways, they belong to distinct sonic potentials. Any given solution to a sound-synthesis problem must therefore provide much more than just a patch: by embodying an underlying control paradigm and a certain way to represent it, it has to generate a sonic potential whose characteristics will more clearly emerge when dealing with several sounds. Seen from this standpoint, a sonic potential already captures the idea of how sound should be structured, controlled, represented, developed and "enjoyed[12]". This is at the same time a control problem, a compositional task, an esthetical issue and an epistemological question.

2. LACK OF GENERALIZED ABSTRACTIONS

Since the esthetical needs of a musician cannot be guessed, every attempt at searching for a more general solution must generate a system both as open as possible and very easy to personalize. The musician's first task will then be to adapt it to his or her own particular way of thinking about sonic potentials.

2.1. Change of representation

The environment we have been developing for almost 20 years[13] addresses this issue from the perspective mentioned just above[14].

[9] Risset proposes three increasingly more refined versions of his bell sound. The second and third one require a control model of type "c".
[10] The "development" of a musical idea is relatively easy to observe in instrumental music, but much more difficult when dealing with sonic processes. It will however not be further developed here.
[11] This "idea" will correspond to the musician's concept of what sort of sonic process to obtain and how to represent it, even if the quality of correspondence may be poor.
[12] That is what a "good" sound experience is.
[13] Started as a set of Music V's PLF subroutines [5] written in Fortran at the "Centro di Sonologia Computazionale" of the University of Padua (1980-82), it was first extended and translated into Lelisp while I was a student at the Media Laboratory of the Massachusetts Institute of Technology (*Chroma*, 1984-86). It was then ported and largely redesigned as a virtual synthesizer in the CLOS environment (Common Lisp's object-oriented system [6]) at IRCAM with the cooperation of Serge Lemouton (1995-6), and finally generalized and incorporated into *Open Music* [7] still at IRCAM (1999-2000).
[14] The fact that it was used for all my electronic productions in several centers and using different software synthesizers, as well by other

It implements a control paradigm of type "c", where the control data are sets of time-tagged instructions. The data structure looks like a matrix, where the rows and columns set up vectors of values for identified control parameters. In the case of Risset's example, this will yield the structure of figure 1.

Start Time (sec)	Duration (sec)	Amp (0-1)	Freq (Hz)
0	20	0.36	224
0	20	0.36	368
0	20	1.0	476
0	20	0.62	680
0	20	0.55	800
0	20	0.05	1096
0	20	0.05	1200
0	20	0.038	1504
0	20	0.05	1628

Figure 1. *Control data seen as a matrix*

In this matrix, some data vary at every row (e.g. the frequency), while others do not.

The most significant conceptual change introduced by *Chroma* concerns the way this matrix is represented. Instead of having a variable number of rows (one per instance of a patch) with a fixed number of columns (control parameters needed by the patch), *Chroma* uses matrices with a fixed number of rows (control parameters) and a variable number of columns (instances). In other words, *Chroma* "turns" the rows into columns and vice versa (fig. 2).

E D	0 (for all the instances)									
D	20 (for all the instances)									
A	1	.36	.36	1	.62	.55	.05	.05	.04	.05
Fq	"look for the values in a data base"									

Figure 2. *A different representation of the matrix*

When the matrix is read vertically, column by column, each control instruction will be reconstructed[1]. Such structure, called an "**event**[2]", is the basic control model used by *Chroma*. It corresponds to an arbitrarily large bank of a single DSP patch within a given synthesizer. It is however a much more powerful abstraction than a simple change of representation, since the values can be given both as literals and as functions, thus providing both the abstraction[3] and the flexibility needed by a musician to implement his or her own functions.

composers at IRCAM already shows that a certain degree of generality was achieved.

[1] There are also other minor changes, such as the transformation of the absolute temporal information (start time) into "entry delays" (ED) relative to the global start time of the process. However, these changes do not affect the basic model.

[2] The words "matrix" and "event" are quite similar, although "event" implies an underlying compositional model (the conceptual model of the sonic process), while "matrix" has a more technical meaning (the control interface) in the context of *Chroma*.

[3] This was one of the main reasons for using Lisp, where data and functions are easily interchangeable.

2.2. Implementation in *Open Music: omChroma*

Initially designed for symbolic computation, *Open Music* is a computer-assisted composition software providing a complete visual programming interface to Common Lisp/CLOS [6]. The user drags and drops *icons* (any representable object), and *containers* (editable panels giving access to the internal structure of objects). The control matrix was implemented as a container. A set of pre-defined and extensible generic functions allow the user to send messages (data or functions) to instances of the container, called *factories,* boxes with inputs and outputs (figured as small round inlets and outlets) that are connected to the internal slots of the object to be created (fig. 3).

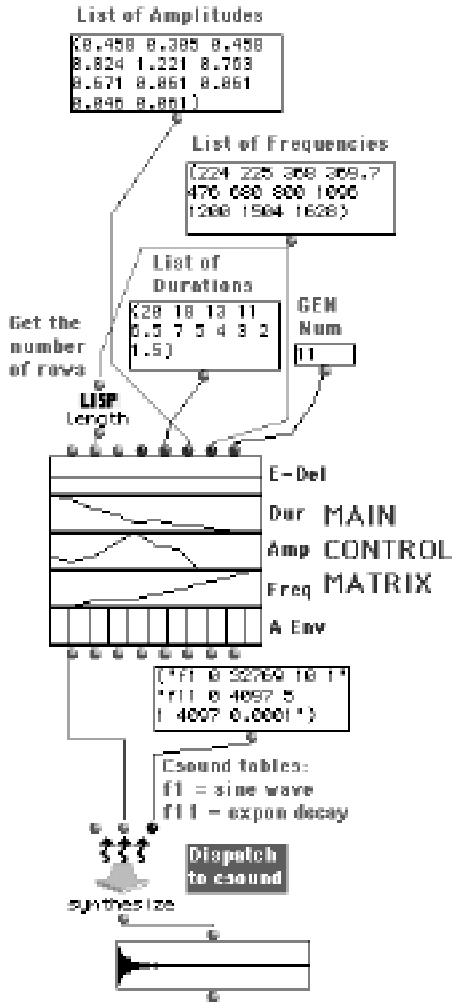

Figure 3. *Reconstruction of Risset's bell sound in omChroma*

Another main advantage was that these functions could be easily connected to the basic compositional processes (harmonic, rhythmic, and the like) available in *Open Music*, independently from the constraints of DSP controls [8].

In this matrix inlets can be either single values, lists of values, break-point tables or Lisp functions. The figure 3 shows the most straightforward translation of Risset's example in *omChroma.* The main control matrix is connected to the generic function "synthesize" (see 2.4) calling "Csound". As with all *Open Music* factories the values can be graphically displayed and manually edited.

This implementation can be very easily abstracted into a more general model for this kind of sonic potential (fig. 4): all the inlets are fed with symbolic or algorithmically-computed values[1].

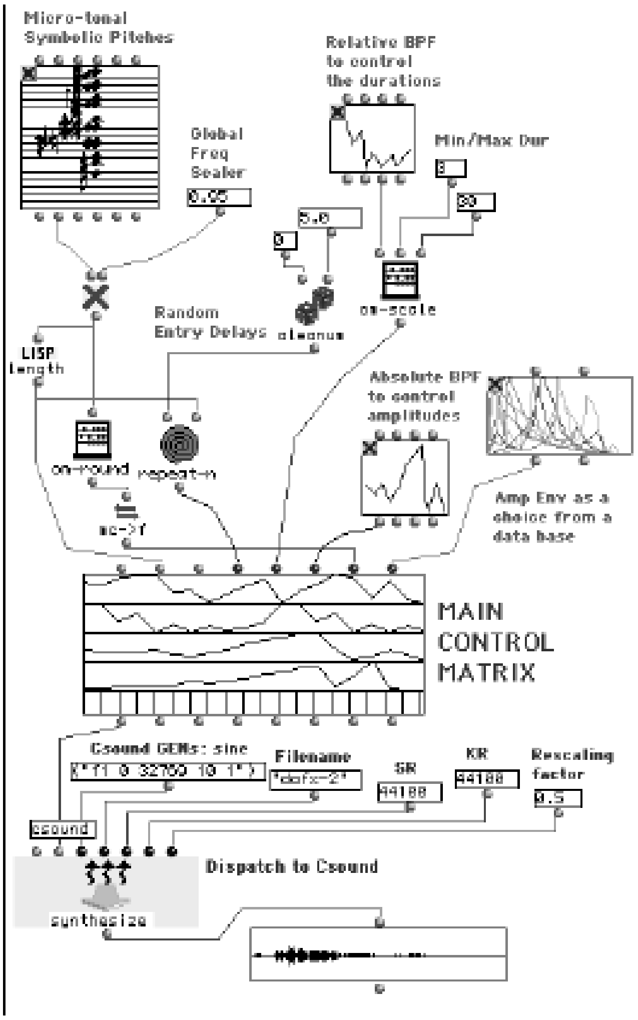

Figure 4. *Generalized model of the control matrix*

2.3. Banks are micro-clusters

Another important change refers to the way banks are conceived. Psychoacoustical research and practical experience have shown the importance of jitter and vibrato to obtain perceptually more natural and musically more satisfactory sounds. Therefore, not only have all the patches used so far for our own personal work a jitter and vibrato module per component, but this concept was further extended: each *component* is actually represented as a "micro-cluster", that is as a set of *sub-components* centered around the frequency contents of the main component and not directly specified within the matrix[2]. The density and frequential width of the cluster, the algorithm used for its computation as well as its temporal profile are parameters set by the user.

When density or width are 0, the model is a bank of type "c". If they are small and aleatorically distributed, interesting, constantly-changing beatings are produced around every component (fig. 5)[3]. This implements a sort of interpretation scheme: each time the score is computed, it never produces an identical signal, but the same "sound idea". Each sound file is hence acoustically unique. Finally, when the parameters are more extreme, the result tends to be perceived as another compositional material, although it may belong to the same sonic potential[4].

```
;    ADD1 0.0 10.0 1.0 6
;    (GLOBAL EVENT START: ADDITIVE SYNTHESIS)

; St. Time, Dur, Amp, Fq, St Pan, Jitt Amp, Trem Amp/Fq, etc…
i11   0.75   2.25 0.09346 234.26   0.5 0.049  0.05 5.84
;    3 sub-components
i11 0.825 2.174 0.09 237.8     0.5 0.049  0.05 5.84
i11 0.901 2.098 0.09 244.52   0.5 0.049  0.05 5.84
i11 0.977 2.022 0.09 241.09   0.5 0.049  0.05 5.84

i11 0.572 2.427 0.88 492.83   0.5 0.036 0.018 4.339
;    2 sub-components
i11 0.638 2.361 0.093 480.86  0.5 0.036 0.018 4.339
i11 0.704 2.295 0.093  491.3   0.5 0.036 0.018 4.339

i11 0.394 2.605 0.83 767.78   0.5 0.033 0.049 2.64
;    no sub-components

i11   0.0    1.0 0.19 262.81   0.0 0.067  0.05 2.693
;    1 sub-component
i11   0.03    0.8 0.19 267.89   0.0 0.067  0.05 2.693

i11   0.8    2.8 0.15 417.19   0.2 0.052 0.021  5.808
```

[1] Even if not familiar with this visual environment (and in spite of the graphical resolution of the figure), it is not hard to see that, for instance, the entry delays are randomly generated by repeating the call to "aleanum" as many times as there are "frequencies" in the object, while the amplitude envelope is chosen from a data base of Break Point Tables (BPF). BPF's also control the duration and amplitudes of the sound (when a control structure of type BPF is connected to an inlet of type "number" it will be automatically sampled over the number of components), whereas the frequencies come from a symbolic chord derived from Risset's example. *Open Music* allows for both a graphic programming style as well as straight Lisp code. Some of the control algorithms are written directly in Lisp for reasons of expressive efficiency.

[2] An elementary reference to this kind of model is already found in Risset's third version of his bell-like sound : the two lower partials are doubled and slightly mistuned and thus generate some beatings that improve the quality of the result.

[3] The figure 5 shows the beginning of a Csound score used for my piece *Traiettoria* [9], where each component is surrounded by few sub-components.

[4] The beginning of the computer part of my work *Traiettoria...deviata* - the first movement of *Traiettoria* - is a process where the amount of "width" and "density" are progressively, albeit not linearly increased. The delicate additive-synthesis and frequency-modulation sounds that start the work develop into larger clusters in about fifty seconds.

```
;      12 sub-components
i11   1.042     2.8 0.15 407.57   0.2 0.052 0.021  5.808
i11   1.045     2.8 0.15 450.27   0.2 0.052 0.021  5.808
i11   1.048     2.8 0.15 455.13   0.2 0.052 0.021  5.808
i11   1.051     2.8 0.15 414.51   0.2 0.052 0.021  5.808
; etc…
```

Figure 5. *Csound score with micro-clusters*

2.4. Virtual synthesizer

Every sound-generating model requires control data whose structure depends on the patch and on the actual synthesis engine being used. Many controls,[1] however, are conceptually the same, independently of their implementation: a vibrato, for instance, is just a "vibrato", no matter how it is controlled at the level of the chosen synthesizer!

The abstraction procedures contained in *omChroma* provide very efficient tools for coping with these issues; since a matrix knows which synthesis engine it is using, it can isolate its peculiarities from the external controls sent to it, thus acting as a syntactic interface. Similar data will no longer necessitate different structures when sent to different engines: since the matrix will automatically take care of it, the user is free to concentrate on higher musical issues[2]. Algorithms for sound control are therefore isolated from a given synthesis engine by an intermediate layer called a *virtual synthesizer*, that is, a "language to represent the specific parameters of sonic processes independently of any given real synthesizer, synthesis engine and computer platform".

The interpretation of the matrix data is concentrated within the method *synthesize*. Depending on the target synthesizer, it will automatically dispatch to either the appropriate score-generation function (as with Csound) or a method talking in real time with such engines as Max/msp or jMax via a communication channel[3]. As new synthesis engines are added to the environment, only the low-level layer providing the interface with the synthesizer will have to be updated.

The figure 6 shows a simple application of the virtual synthesizer. The basic material comes from an analysis of the sound of a cymbal using Diphone's ModRes[4]. ModRes produced an SDIF file loaded into the patch **"fob"**.

The left side of the figure instanciates an event of type Chant and passes it to "synthesize" calling the chant's patch number 0 (FOF bank). The right side instanciates an event of type Csound, which receives the same analytical data. The two sounds are strikingly similar.

This example elucidates the salient features of a virtual synthesizer: the same data are used to run different synthesis engines or various algorithms. In this case, an identical DSP unit is run on two distinct engines. It would have been possible to use the same data to control other synthesis algorithms (additive synthesis, formantic frequency modulation, filters, etc.) within the same synthesis engine.

Some restrictions, however, do apply, since any given engine has some peculiarities that are not found in others, let alone changes in the implementation or the efficiency.[5]

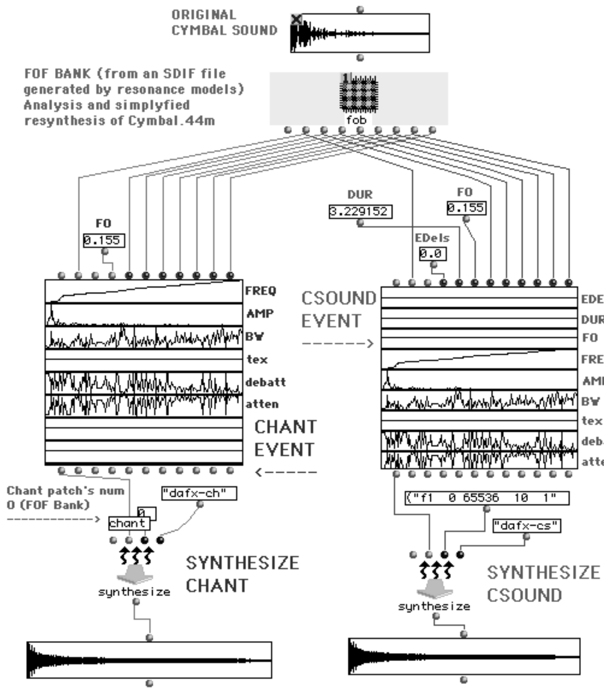

Figure 6. *Same control data dispatched to two different synthesis engines*

2.5. Further developments

Possible developments are only briefly hinted at. Among many alternatives they include: finding classes that represent the synthesis engine itself, and not only its control data; further extending the independence of the control abstraction from a synthesis model itself, that is generalizing the model of a virtual synthesizer; implementing algorithms that link the symbolic computation of material for instrumental music with analysis and control data for computer-generated sounds; providing a data base "presets" of proved importance.

---

[1] Such as amplitude envelopes, frequencies, maximum amplitudes, vibrato, and so on and so forth.

[2] Notice that the choice of this interpretation of the matrix is arbitrary; it is made here only because it is the most practical one when dealing with the Music X style of control. When applied to other engines, different interpretations will eventually be necessary.

[3] Currently, methods are available for Csound, Chant, and Modalys [10] as well as for writing data in the SDIF file format.

[4] Models of Resonance (see the work of the Analysis and Synthesis Team at IRCAM, www.ircam.fr, for further details).

5 For instance, the Csound FOF contains a built-in "octaviation" field that is not directly accessible in Chant and ought to be implemented in a higher control layer. Being an object of type "matrix", it also directly allows for different entry delays of each FOF. This would be relatively cumbersome to implement in Chant. On the other hand, Chant allows for several embedded layers of control, like a global "phrase envelope" applied to a whole sequence of events, that are quite laborious to realize in Csound.

3. EPISTEMOLOGICAL SIGNIFICANCE

We started this text by highlighting the puzzling aspects of any system for the high-level control of sound: its exigency of both efficiency and generality requires a powerful and expressive environment; its final dependence on the composer's judgment demands both flexibility and easiness to personalize. These are hard features to combine together and might explain the meager amount of research in this domain.

We have seen that each "event" necessarily "captures" a certain way of thinking about sound, which is related both to some more or less implicit knowledge about sound potential and to esthetical considerations. As a consequence, even if many tasks still remain very technical, it is not possible to thoroughly delve into this issue outside of a fundamental epistemological framework, the best-suited context where to tackle this sort of questions in their essence.

3.1. Educational responsibility

Such a framework is not easy to handle. Unfortunately, one of the hardest tasks facing a musician when learning sound control is the scarcity of analytical documentation available about past works in this domain[1], let alone insufficient technical reports or program notes. Too often is the musician forced to start once more from scratch! This greatly contrasts with the study of other artistic disciplines, where he or she can learn both from the example of the masters of the past as well as from personal, creative work.

This text has attempted to show that it is not only a matter of building yet another environment, but mainly of addressing the appropriate epistemological framework. The study of such a framework, with its essential abstract quality with respect to a given synthesis engine or model might become a precious source of information for musicologists and musicians, something comparable to examining a composer's sketches for instrumental music.

The detailed analysis of the examples will probably speed up the learning process, hence leading to the expression of more advanced needs, which will in turn call for more powerful solutions, and so on. A much better understanding of the mixture between these embedded, contrasted, sometimes opposed intellectual tasks is therefore needed in order to come up with more powerful solutions. It is an endeavor that requires a highly interdisciplinary approach drawing from such diverse domains as machine-independent visual programming, music cognition, DSP, symbolic music writing and sound design.

The high-level musical control of sonic processes is indeed a very multifaceted and subtle domain. Further research might reveal radically different approaches, which will eventually lead to the discovery of yet unsuspected ways to compose sonic processes and to methods to link them with both instrumental composition and processes coming from other media.

[1] Or, in the rare cases where it is available, it refers most of the time to obsolete systems , that are no longer in usage.

4. REFERENCES

[1] Stroppa, M. 2000. "Live Electronics and Live Music: towards a Critique of Intercation", in "The Esthetics of Live Electronics", M. Battier, ed. Harwood Academic Press.

[2] Risset, J-C. 1969. "An Introductory Catalogue of Computer Synthesized Sounds", reprinted in "The Historical CD of Digital Sound Synthesis", Computer Music Currents nº 13, Wergo, Germany.

[3] Mc Adams, S. 1981. "Spectral Fusion and the creation of auditory images", in M. Clynes, ed. Music, Mind, and Brain: The Neuropsychology of Music. New York: Plenum.

[4] Cohen-Lévinas, D. 1993. "Entretien avec Marco Stroppa". Les Cahiers de l'IRCAM, Nº 3, pp. 99-117.

[5] Mathews, M. 1969. "The Technology of Computer Music". MIT Press.

[6] Steele, G.L. 1990. "Common Lisp The Language", 2nd Edition. Digital Press.

[7] Assayag G., Rueda C., Laurson M., Agon C., Delerue O. 1999. "Computer Assisted Composition at Ircam : PatchWork & OpenMusic". Computer Music Journal 23:3.

[8] Agon C., Stroppa M., Assayag G. 2000. "High Level Musical Control of Sound Synthesis in OpenMusic", Proceedings of the International Computer Music Conference, Berlin.

[9] Stroppa, M. 1984. "Traiettoria", a cycle of three pieces (Traiettoria…deviata, Dialoghi, Constrasti) for piano and computer-generated sounds. Recorded by Wergo, Digital Music Digital, n. WER 2030-2. Pierre-Laurent Aimard, piano, Marco Stroppa, sound projection.

[10] Eckel, G., Iovino, F., Caussé, R. 1995. "Sound Synthesis by Physical Modelling with Modalys". Proceedings of the ISMA.