The Control of the CHANT Synthesizer in OpenMusic: Modelling Continuous Aspects in Sound Synthesis

Jean Bresson, Marco Stroppa

THE CONTROL OF THE CHANT SYNTHESIZER IN OPENMUSIC: MODELLING CONTINUOUS ASPECTS IN SOUND SYNTHESIS

Jean Bresson

STMS: IRCAM-CNRS-UPMC
1, place I. Stravisnky 75004 Paris, France

Marco Stroppa

University of Music and Performing Arts
Urbanstr. 25, 70182 Stuttgart, Germany

ABSTRACT

The specific model and control paradigm of the CHANT synthesizer raise interesting issues and possibilities for the control of continuous aspects in sound synthesis. New objects for the control of CHANT in OpenMusic and in the OMChroma library provide high-level tools and structures for integrating these aspects in compositional frameworks.

1. BASICS AND HISTORICAL OVERVIEW

The CHANT program [16] was developed in the early 80s for controlling sung voice synthesis models. In this program sounds are produced by assembling synthesis modules mainly compound of FOFs[1] and filters, while "control" processes determine the evolution of these modules' parameters. The fine tuning and control of different parameters of the FOFs and filters allows to reproduce high-quality vocal sounds by modelling the real behavior of the vocal system. Despite this voice-based model, the modular aspects and control possibilities of CHANT make it capable of generating wide ranges of spectra and sounds more or less related to a vocal simulation.

The control of the CHANT modules was initially performed following a "rule-based" model. Algorithmic processes (rules) could be activated or plugged to the synthesizer and generate the values of a determined set of parameters at a given control rate: An active rule process knows about the synthesizer's current state and about the current time, and can set or modify any parameter accordingly. Predefined rules derived from signal and psychoacoustic analyses have been implemented, such as the automatic determination of the formant relative amplitudes or bandwidths, or their evolutions depending on the variation of other external or internal parameters. More complex or specific rules could also be programmed and applied to personalize the synthesizer's behavior. The Formes environment [14] was created shortly after in order to handle the generation and application of such rules in higher-level and larger-scale temporal structures.

In the 90s CHANT was ported and used in several different contexts, such as the PW-Chant library in PatchWork [11] or in the Diphone software [15], but for the most part, "rules" and other control aspects were left out of the synthesizer.

More recently, two CHANT libraries were developed in the OpenMusic computer-aided composition environment [3]: Chant-lib and OM-Chant. In Chant-lib[2] the synthesis kernel of CHANT was re-implemented in the Csound language, and the main control rules from PW-Chant were ported to OpenMusic. The work presented in this paper concerns OM-Chant and its integration to the OMChroma system [1]. It uses the original CHANT kernel and provides means to generate and format adequate parameters for this synthesizer, which constitutes an original complementary approach regarding control paradigms and applications.[3]

2. CONTROL OF CHANT SYNTHESIS

Sound synthesis with CHANT is performed and controlled at two different layers: a synthesis patch runs continuously and its parameters are periodically modified by external control changes (see Figure 1).

The modules in the synthesis patch are FOF banks (FOF generators activated by a fundamental frequency excitation signal), filter banks, noise generators and input sounds.[4] The latest CHANT implementation is controlled via files in the SDIF format [17] where all parameter values and evolutions are set and stored in precisely time-tagged frames. These parameters changes are neither necessarily equally sampled nor synchronous: each module can be controlled individually at any moment or at any rate. Between each specified value change for the synthesis parameters, CHANT performs linear interpolations producing smooth variations and transitions between the successive states of the synthesizer's modules. This control scheme induces an original "continuous" approach to the control of sound synthesis, which is conceptually different (and complementary) to the "event-based" approach used by many existing synthesis control systems.

[1] FOF synthesis (*Fonctions d'Onde Formantiques*, or Formant Wave Functions [13]) consists in defining functions corresponding to the different formants of a sound spectrum, that is, to generate the signal resulting from impulses going through different resonant filters. The parameters of the FOFs (central frequency, bandwidth, amplitude, etc.) allow to determine and control the position and shape of the formants, which is well adapted to the simulation of the singing voice.

[2] Chant-lib by Romain Michon, http://electro-m.fr/∼romainM/

[3] See [7] for a description / comparison of Chant-lib and OM-Chant.

[4] In theory the patch modules can be combined freely, but the current CHANT implementation only provides a set of predefined patches identified by a patch number (e.g. Patch 0 = FOFs only; Patch 1 = FOFs + Noise → Filter; Patch 2 = Noise + Sound → Filter; etc.)

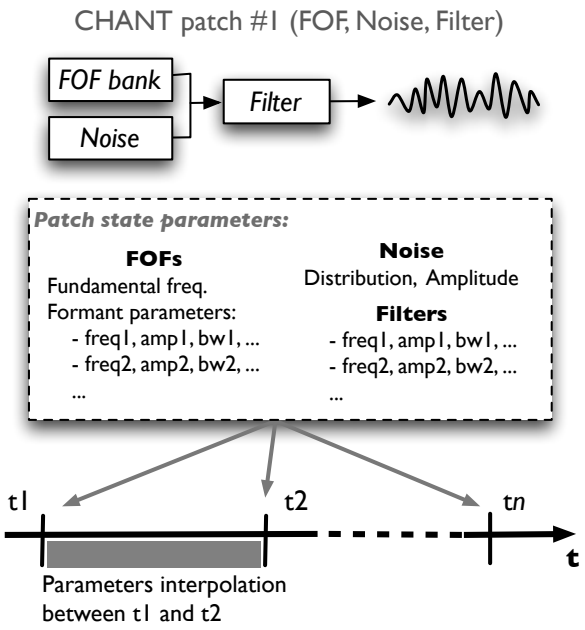

Figure 1. The control of the CHANT synthesizer: timed state modification and continuous interpolations.

3. CONTINUOUS VS. DISCRETE PARADIGM

The distinction between discrete and continuous temporal structures in sounds is a long-standing issue [8]. Indeed, it is not always straightforward to discriminate in sounds and sound generation processes what comes within discrete or continuous time: Sounds can be considered as continuous objects (acoustic waves with continuously varying internal/spectral morphologies), or as discrete structures (sequences of samples, for digital sounds, but also at a higher level, as significant or self-consistent musical objects).

In the tradition of the Music *N* languages [12], the note concept has been preserved as basic specification in sound synthesis processes: There is a clear distinction between the sound microstructure (DSP processes defined in the synthesis "instruments") and the "score" where discrete events are scheduled. Compositional issues therefore come under discrete time, organizing events inside the sound, or organizing sound events themselves. Continuous aspects (for instance modulations, evolutions or transitions between states of the parameters) are mostly handled inside events using static structures (tables or breakpoint functions, for instance) or appropriate modules within the DSP patch.[5] This approach, however, makes it difficult to handle and control the transitions and articulations between events, which occur naturally and are fundamental in the perception of sounds. On the other hand, systems like CHANT/Formes, but also other functional languages such as Arctic [9], tend to consider sound as a unique continuous flow undergoing global compositional processing (from microstructure to high-level musical forms). In this case continuous evolutions are controlled by processes computing periodically the signal or sound synthesis parameters, possibly starting from reference states also integrated in the control level.[6]

At a meeting point of these two complementary approaches, composition with sound synthesis should therefore not only position synthesis events and parameters in time, but also access the continuous structures of sound textures. This dual conception requires dedicated systems for the specification and control of sound characteristics.

4. OM-CHANT

A toolkit in the new OpenMusic OM-Chant library allows to format control structures and create SDIF files adapted to the control of CHANT (see Figure 2). Utilities are also provided in order to facilitate setting the CHANT parameters (data base of formant values, implementation of the different CHANT predefined rules, etc.)

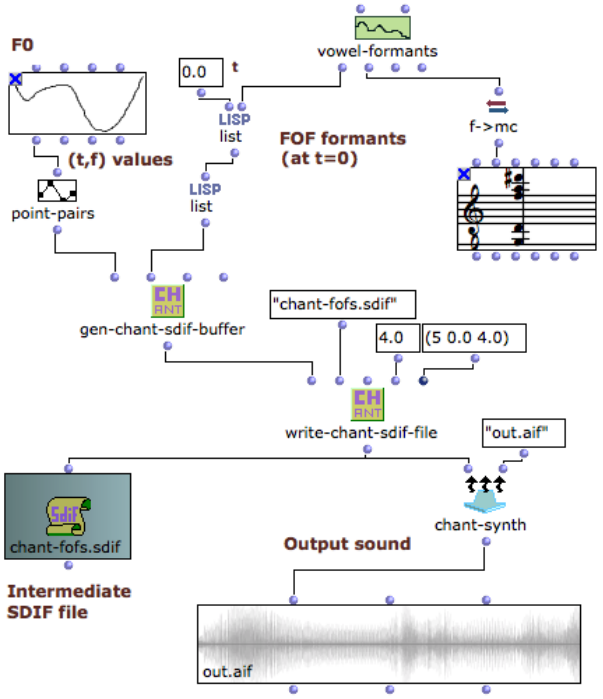

Figure 2. OM-Chant: Generation of SDIF data for the control of CHANT in OpenMusic.

OpenMusic processes using OM-Chant generate a sequence of SDIF frames containing values for the different parameters of the synthesizer's modules.[7] SDIF control frames can be arbitrarily distributed in time, either very sparsely, in which case CHANT will interpolate between

[5] Solutions to generate more elaborated time functions in music or synthesis environments have been proposed, see for instance [2, 10].

[6] Real-time sound processing environments often perform such "continuous" control of the synthesis parameters but do not allow high-level musical formalization of the related processes [18], and are therefore not addressed in the current discussion.

[7] The specification of the SDIF control structure for CHANT is available at http://sdif.sourceforge.net/standard/sdif-standard.html. See also [6] for an overview of SDIF data manipulation tools in OpenMusic.

the specified values, or precisely sampled, describing specific and fine parameters evolutions (see Section 2). The control approach emphasized in OM-Chant therefore follows the CHANT paradigm and circumvents the usage of timed events in the control of sound synthesis: Processes of arbitrary complexity can be programmed to generate the continuous evolution of the different synthesis parameters.

5. CHANT EVENTS: AN IMPLEMENTATION IN OMCHROMA

Discrete events specified with time localization and extent are convenient for compositional control, but not straightforward to integrate in the OM-Chant toolkit and conceptual framework [7]. The consideration of timed events within a continuous time flow and the description of their morphology and articulations imply specific decisions. An extension of the OMChroma system to the control of CHANT synthesis, which we created recently, introduces these concepts in the compositional environment.

OMChroma is a system for the control of sound synthesis integrated in OpenMusic, where the main control structures are time-tagged matrices representing compound sound events. The different components of an OMChroma matrix describe parameter values for different instances of a synthesis module [1]. This system was initially designed to control Csound or conceptually similar sound synthesizers based on the "event" concept (the matrix components generate the different "score statements"). It was however meant to be modular and easily extensible to other sound synthesizers. This new extension will indeed bring new synthesis possibilities to OMChroma, while providing a structured framework for the temporal organization and control of the CHANT processes.

5.1. CHANT Event Classes

Five new classes have been created in OMChroma to represent "CHANT events". Each class is attached to an element of the CHANT patch, and its instances will determine and control the evolution of this element's parameters during a given time interval. *CH-F0* represents "FOF fundamental frequency events", *CH-FOF* represents "FOF formants events" (the combination of *CH-F0* and *CH-FOF* values determines the full parameters for the FOF generator), *CH-FLT* represents "filter events", *CH-NOISE* represents "noise events", and *CH-SND* represents "sound events" (audio input playback from a sound file).[8]

CHANT classes have an *onset* and a *duration* slot, and can be considered as events organized in a global temporal context. Events can have an arbitrarily long duration, and continuous evolutions can be specified for all or part of the synthesis parameters during the event time interval.

Each event is responsible for the generation of a set of time-tagged SDIF frames. A sorted list of CHANT events therefore allows to generate the SDIF file containing the full control sequence for a given CHANT patch, then synthesized using OM-Chant tools (see Figure 3).

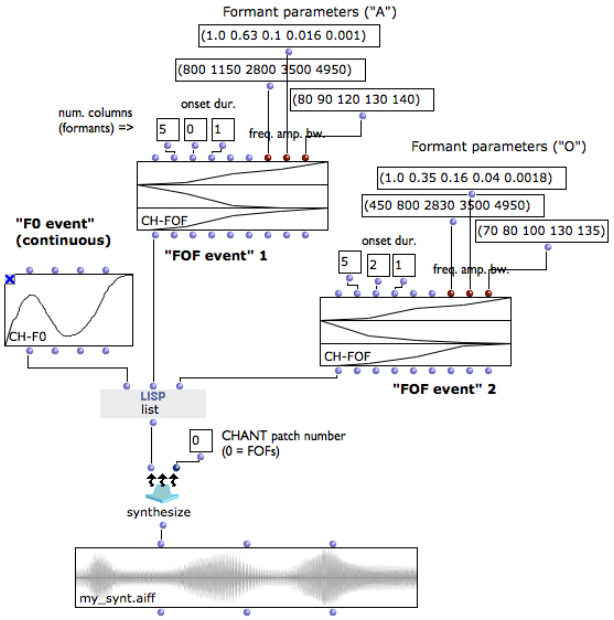

Figure 3. CHANT synthesis in OMChroma: Using high-level structures to represent CHANT events.

5.2. Continuous Control

Two control processes from the original CHANT program have been implemented in OMChroma for the setting and modulation of different synthesis parameters and the general design of the internal morphology of CHANT events: the *jitter* and the *vibrato*.[9] Both effects can be applied to the initial values either by addition or multiplication (see Figure 4). Another control utility available is the *pulse-train* function, which generates impulses with precise tuning of constant or continuous values of the period, amplitude, attack and release, etc. This function is principally meant to generate amplitude envelopes (e.g. for noise generators, see Figure 5).

5.3. Temporal Representation using the Maquette

The OpenMusic *Maquette* can be used to represent compositional objects and processes embedded in time structures, including the parameters of a synthesis process [5]. It constitutes a convenient interface to represent CHANT synthesis events in a temporal context and better handle and control their temporal characteristics (see Figure 6).

[8] Note that only *CH-FOF* and *CH-FLT* are actual matrices, whose columns represent the different "components" or formants of a FOF or filter bank and whose rows represent their different parameters. *CH-F0* and *CH-NOISE* are one-dimensional descriptions, and *CH-SOUND* is a simple reference to a file on the disk.

[9] *Jitter* and *vibrato* produce respectively aleatoric variations and sinusoïdal modulations with given frequencies and amplitudes around initial, constant or continuous parameters.

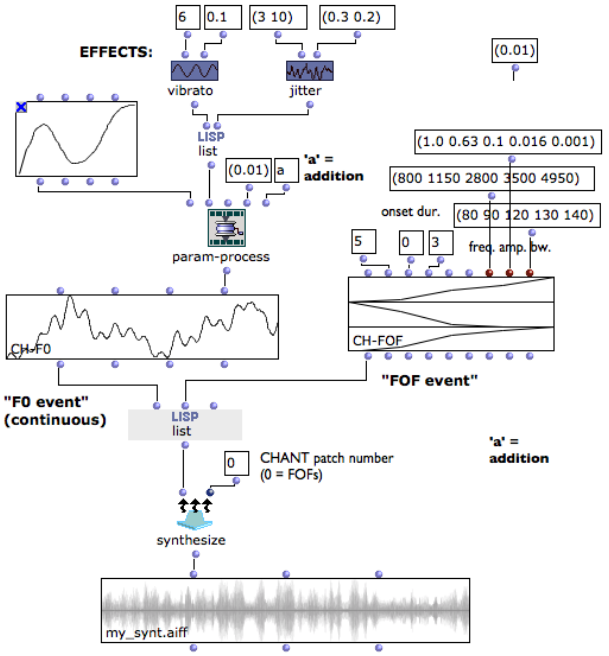

Figure 4. Applying a vibrato and a double jitter to the F0 from Figure 3. Effects are applied by addition at a sampling rate specified in *param-process*.

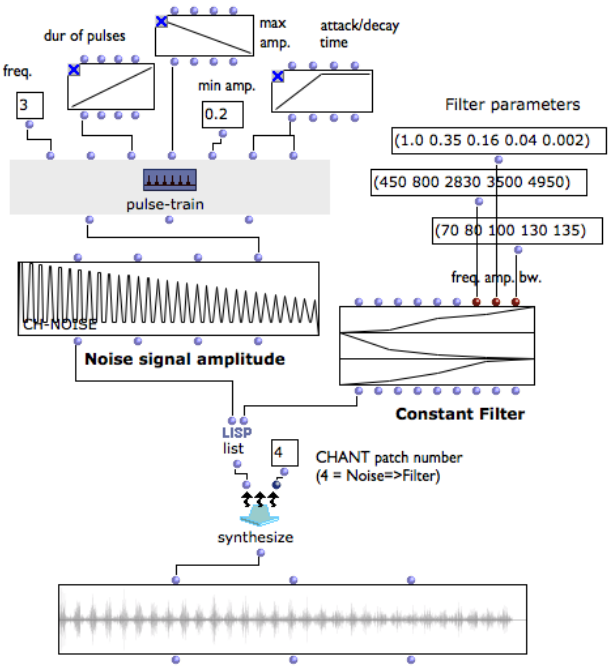

Figure 5. Using *pulse-train* to generate the amplitude envelope of a "noise event".

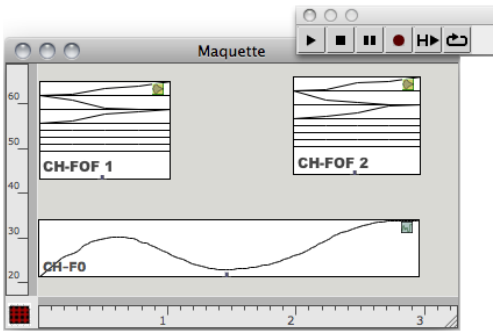

Figure 6. CHANT objects in the OpenMusic Maquette. This process is functionally equivalent to the patch in Figure 3. Each box represents an OpenMusic object or patch producing a CHANT event.

In a maquette, the onset and duration of the events are represented by the position and extent of their containing box. Computation of the whole temporal structure can be triggered, allowing to synthesize and hear the resulting sounds.

6. EVENTS PROPERTIES

Musical events start at a specific time and generally run for a given duration. In order to obtain the steady state of a set of parameter during a given time interval, the CHANT events duplicate their values at two moments t_b and t_e determined by their beginning time and duration (see Figure 7). Between these two moments, the default synthesizer behavior will produce a stable sound (interpolation between two identical states).

Concretely, this simple process occurs at writing the data frames corresponding to the CHANT object in the SDIF file, that is, just before synthesis. If no "continuous" evolution exists inside the events, two SDIF frames are produced for each one of them at t_b and t_e.

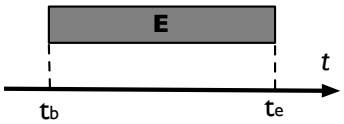

Figure 7. Representation of an "event" in the continuous time line.

Sequences of events may however leave possible ambiguous intervals in the time line (see Figure 8). In this case, CHANT would in principle interpolate and produce a continuous transition between the two events (i.e. between t_e1 and t_b2, from the final state of E1 to the initial state of E2).

In order to obtain silence between the successive events, another optional attribute of the CHANT events makes them duplicate again their set of parameters from t_b and t_e respectively to $t_b + \varepsilon$ and $t_e - \varepsilon$, with amplitude values

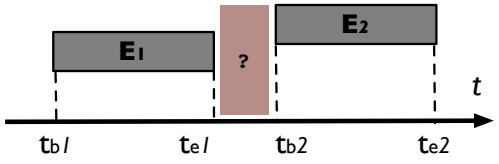

Figure 8. Representation of a sequence of events in the continuous time line.

set to 0 at t_b and t_e (see Figure 9). In this case the value ε will determine a fade-in/out time during which the amplitude is interpolated from 0 to its value in En between $t_b n$ and $t_b n + \varepsilon$ and from the value in En to 0 between $t_e n - \varepsilon$ $t_e n$. Between the successive events, the synthesizer's parameters are still interpolated but the amplitude is zero.

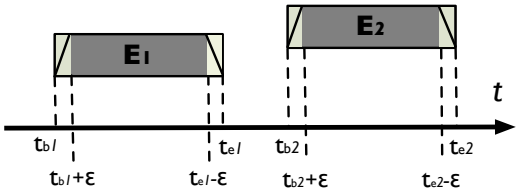

Figure 9. Generating silence and fade-in/fade-out between CHANT events.

The case where two events overlap is also worth being considered. In a configuration such as the one in Figure 10, the overlapping of two events (with steady internal parameters) would produce the following default behavior: State E1 at $t_b 1$, then interpolation from E1 to E2 between $t_b 1$ and $t_b 2$, interpolation from E2 back to E1 between $t_b 2$ and $t_e 1$, and finally interpolation from E1 to E2 between $t_e 1$ and $t_e 2$.

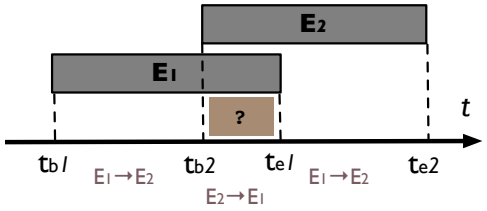

Figure 10. Overlapping events.

A simple trick allows to produce a more intuitive result: by switching the parameters values at $t_e 1$ and $t_b 2$, the parameters become stable between $t_b 1$ and $t_b 2$ and between $t_e 1$ and $t_e 2$. On the overlapping interval (between $t_b 2$ and $t_e 1$) the synthesizer will interpolate between the states E1 and E2, which is a more likely desired transition behavior (see Figure 11).

This method may work in some cases, but some problems remain. If continuous evolutions are specified within the events, it is simply not applicable: Parameters from E1 and E2 would be interleaved in the control flow, which would produce unpredictable (and probably undesired) effects in sound synthesis. A solution in this case is to

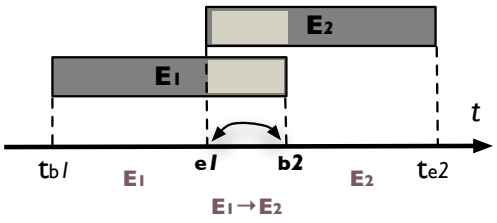

Figure 11. Generating a transition on the events overlapping interval.

rescale and "shrink" the parameters curves from the original event duration to the "steady" intervals only, then to perform the interpolations between ending and initial states of the events. Other solutions could simply be to "cut" the evolution at the time were overlapping/transition starts, or to define a way to continuously go from one set of parameters to the other (as a kind of fade-in/fade-out transition). Due to the multiplicity of choices and subjective decisions involved, the possibility to program and personalize the process seems necessary in order to allow for the definition of specific behaviors, the personalization of interpolation profiles, or the application of individual transitions for the different synthesis parameters.

7. CONCLUSION AND FUTURE WORKS

In addition to giving access and renewed control possibilities for this synthesizer, the use of CHANT in OpenMusic allowed us to investigate new aspects in the domain of the control of sound synthesis, and to propose adequate tools integrated in the computer-aided composition environment. The OMChroma objects presented in this paper, in particular, might allow to achieve advanced and powerful control over both high-level composition and fine tuning of the continuous evolution of compositional structures driving CHANT synthesis.

Event superimposition and overlapping are probably the main issues remaining to address. The control of transitions between successive events could be implemented either as an additional functional characteristics of the CHANT event objects or as independent structures superimposed to a sequence of events.

Even though, CHANT synthesis remains inherently monophonic: Transitions are always pairwise and can be performed between two overlapping events only. In the current state of the synthesizer, it is not possible either to imagine several simultaneous "voices" made of different FOFs and independent input frequencies.Voice polyphony can however easily be simulated by generating different files and mix them using the sound processing tools available in OpenMusic [4]. The maquette interface would moreover provide a hierarchical view and control over accordingly structured processes.

Another direction to explore concerns the spatialization, or distribution of the different components of the

CHANT patches (FOFs, filters, etc.) over a given number of audio channels. This feature is supported by the CHANT synthesizer and can be controlled for each individual synthesis component.

8. REFERENCES

[1] C. Agon, J. Bresson, and M. Stroppa, "OMChroma: Compositional Control of Sound Synthesis," *Computer Music Journal*, vol. 35, no. 2, 2011.

[2] D. P. Anderson and R. Kuivila, "Continuous Abstractions for Discrete Events Languages," *Computer Music Journal*, vol. 13, no. 3, 1989.

[3] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer Assisted Composition at IRCAM: From PatchWork to OpenMusic," *Computer Music Journal*, vol. 23, no. 3, 1999.

[4] J. Bresson, "Sound Processing in OpenMusic," in *Proceedings of the International Conference on Digital Audio Effects*, Montréal, Canada, 2006.

[5] J. Bresson and C. Agon, "Temporal Control over Sound Synthesis Processes," in *Proceedings of the Sound and Music Computing Conference*, Marseille, France, 2006.

[6] ——, "Processing Sound and Music Description Data Using OpenMusic," in *Proceedings of the International Computer Music Conference*, New York / Stony Brook, USA, 2010.

[7] J. Bresson and R. Michon, "Implémentations et Contrôle du Synthétiseur CHANT dans OpenMusic," in *Actes des Journées d'Informatique Musicale*, Saint-Etienne, France, 2011.

[8] R. B. Dannenberg, P. Desain, and H. Honing, "Programming Language Design for Music," in *Musical Signal Processing*, C. Roads, S. T. Pope, A. Piccialli, and G. DePoli, Eds. Swets and Zeitlinger, 1997.

[9] R. B. Dannenberg, P. McAvinney, and D. Rubine, "Arctic : A Functional Language for Real-Time Systems," *Computer Music Journal*, vol. 10, no. 4, 1986.

[10] P. Desain and H. Honing, "Towards Algorithmic Descriptions of Continuous Modulations of Musical Parameters," in *Proceedings of the International Computer Music Conference*, Banff Center for the Arts, Canada, 1995.

[11] F. Iovino, M. Laurson, and L. Pottier, *PW-Chant Reference*, IRCAM, Paris, France, 1994.

[12] M. V. Mathews, Ed., *The Technology of Computer Music*. MIT Press, 1969.

[13] X. Rodet, "Time-domain Formant-wave Function Synthesis," *Computer Music Journal*, vol. 8, no. 3, 1984.

[14] X. Rodet and P. Cointe, "Formes: Compostion and Scheduling of Processes," *Computer Music Journal*, vol. 8, no. 3, 1984.

[15] X. Rodet and A. Lefevre, "The Diphone Program : New Features, New synthesis Methods and Experience of Musical Use," in *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.

[16] X. Rodet, Y. Potard, and J.-B. Barrière, "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General," *Computer Music Journal*, vol. 8, no. 3, 1984.

[17] D. Schwartz and M. Wright, "Extensions and Applications of the SDIF Sound Description Interchange Format," in *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.

[18] M. Stroppa, "Live electronics or... live music? Towards a critique of interaction," *Aesthetics of Live Electronic Music - Comtemporary Music Review*, vol. 18, no. 3, 1999.