

SERIAL RSS SOUND INSTALLATION AS OPEN WORK: THE *BABELCAST*

Christopher Ariza

Department of Music
Towson University
8000 York Road
Towson, MD 21201 USA

ABSTRACT

Using Umberto Eco's notion of the open work as a point of departure, this paper describes the context and technologies of the *babelcast*. The historical and theoretical context is first introduced. A brief description of two key technologies, athenaCL and RSS media syndication, follows. The various formats of the *babelcast* are described. Finally, the production process of the *babelcast*, employing Python, athenaCL, Csound, and FFmpeg, is detailed.

1. INTRODUCTION

In 1962 Umberto Eco, considering specific contemporary art works, defined as "open" those works that "... reject the definitive, concluded message and multiply the formal possibilities of the distribution of their elements" [8]. Eco provides examples of such works in many media, describing musical works of Karlheinz Stockhausen, Luciano Berio, and Henri Pousseur, as well as the dynamic mobiles of Alexander Calder [8]. In particular, Eco considers musical works that resist a definitive, concluded message by employing dynamic construction in performance: these works "... are brought to their conclusion by the performer at the same time as he experiences them on an aesthetic plane" [8]. Eco considers such works "quite literally 'unfinished'" [8] in that the performer provides the final musical deployment.

While Eco [7] and others (e.g. [3], [6], [19], [16], [14], [4], [12]) employ this notion of the open work as a metaphor for the plural and receiver-dependent interpretation of works in a variety of media, the description provided above isolates a specifically literal and material class of openness. (Alicja Helman identifies this as the "narrow," rather than the "broader," sense of the open work concept [10].) Such materially open works, where the work is literally indefinite and without conclusion, are the focus of this paper. Others, often emphasizing the use of indeterminacy in the creation or performance of a work, have similarly referenced Eco's open work concept in the context of this material openness [5], [15].

Eco does not consider alternative types of musical works that may also be considered similarly open. A serial, web-based Really Simple Syndication (RSS) sound installation offers a new kind of open work. Rather than a single piece, such a work offers a series of installments that are unified by procedural methods but distinguished by raw materials. The combination of a computer-aided algorithmic composition (CAAC) system with RSS distribution permits the creation and

deployment of this new type of open musical work. The *babelcast* is an example of such a work.

The *babelcast* serial RSS sound installation (a type of podcast) is a response to the superabundance of audio and visual media. Political media, in particular, is the focus of this work. Political media is unique not only in its highly-redundant presentations (where a single statement might be recorded and distributed by numerous press organizations), but in its specifically theatrical and plastic nature. Political-speech, not specific political agenda, is the subject of the *babelcast*. The subtitle of the *babelcast* is "information abduction and reduction"; the removal of context (abduction) and the algorithmic fracturing and recombination (reduction) function to provide an alternative perspective to political speech.

This paper describes the context and technologies of the *babelcast*. The historical and theoretical context is first introduced. A brief description of two key technologies, athenaCL and RSS media syndication, follows. The various formats of the *babelcast* are described. Finally, the production process of the *babelcast*, employing Python, athenaCL, Csound, and FFmpeg, is detailed.

2. GENERATIVE MUSIC SYSTEMS AND THE OPEN WORK

While Eco defines open works as "unfinished" and "works in movement" [8], he does not consider the use of computer-aided or generative techniques. This is likely because such technologies were not widely used in the 1960s. Contemporary generative music systems permit the dynamic creation of numerous variants or instances from a single procedural system. A work can become a system, where musical outputs become performances, instances, or variants of that single work.

As described above, Eco provides pre-computer examples of works that are generated anew with each performance. With the application of the computer, greater automation and specification of this approach is possible. This potential has been recognized by composers and developers of CAAC systems. Gottfried Michael Koenig frequently used the term "variants" (translated from the German) to describe the many possible outputs a CAAC system produces from a common system configuration [13]. Lejaren Hiller used the term "versions" to describe interchangeable movements created from common procedures with "small to substantial but presumably significant changes of input data" [11]. Iannis Xenakis referred to the outputs of a single system as constituting a family: "the

basic law ... generates a whole family of compositions as a function of the superficial density" [20].

As a way of describing a similar family of compositional variants, Sever Tipei introduced the term "manifold compositions." Tipei, however, proposes a rigorous standard: the CAAC system must run a "comprehensive" program, a program that, once initiated "... does not require the composer's intervention to produce final output"; the same data, as input, must be provided for each variant; and each variant may be performed in public not more than once [17]. Tipei describes these works in a manner similar to Eco's notion of an open work, stating that such works "... challenge the common view of the work as a static product, an art object given once for all ..." [17]. Tipei, however, distinguishes manifold composition from human-performed variants in that "... each Manifold member represents a frozen image of a performer-independent random process..." [17]. While Tipei mentions the work of Koenig, Xenakis, and others, he states that "none of them have attempted, however, to mass-produce such variants ..." [17]. Tipei offers his computer generated composition series *Many Worlds*, for percussion, as an example of a manifold. More recently Tipei has created new manifolds, including over 100 variants of the *A.N.L.-folds* manifold and the composition series *Daria* and *decaf* [18].

The mass-production of compositional variants, while not necessarily a valuable aesthetic goal in isolation, is a compositional framework worth exploration. Musical practices of improvisation and interpretation have, in some sense, always produced compositional variance. The use of machine production at low structural levels, however, distinguishes the mass-production discussed here. In order to make such mass production viable, the production process must be highly automated, the medium must be easy to distribute and consume, and the source content must be of suitable variety to maintain long-term listener interest. The *babelcast* attempts to meet all of these demands.

3. ATHENAACL AND RSS MEDIA SYNDICATION

The *babelcast* depends on two key technologies: a scriptable CAAC system for generating and configuring audio content, and an internet-based distribution mechanism specialized for serial content.

The athenaCL system is a cross platform, open source, CAAC system built in Python [1]. While offering an interactive command-line interface, the system can be scripted and controlled from within the Python language. While working directly with low-level objects is possible, high level athenaCL commands can be collected and configured as Python strings. These strings can be executed within an athenaCL Interpreter object. Such high-level commands permit designing monophonic or polyphonic musical parts by mixing generative procedures within athenaCL with generative

configurations programmed in the controlling Python script.

The Really Simple Syndication protocol specifies a light-weight mechanism for publishers to create "feeds." A feed is simply an Extensible Markup Language (XML) file that is regularly updated by the publisher. The feed specifies metadata such as the name of the feed and the source web-site URL. The feed specifies a number of entities, or content items. These content items may be text (such as news stories) or links to multimedia content (such as audio or video). While there are numerous alternative XML-based music and multimedia representation formats, RSS, by only providing links to media content, is simple and portable.

A feed aggregator allows a user to select feeds and view their content. Aggregators provide mechanisms to scan and update feeds automatically. Specialized feed aggregators may be embedded in other programs, such as within a web browser or media player. The Opera web browser (version 8.5 and above), for example, has a dedicated Feeds menu that permits any RSS feed to be stored, updated, and browsed. Apple's iTunes has a dedicated Podcast menu that permits RSS audio and video feeds to be stored, updated, and browsed.

RSS feeds permit any publisher with a simple file server the ability to create syndicated content. Once a user "subscribes" to a feed, the feed aggregator will (if configured) automatically check for new content and retrieve it. In the case of audio feeds (podcasts), a media distribution system similar to streamed content is created. RSS feeds, however, do not require dedicated streaming servers, constant bandwidth requirements, or other expensive network facilities.

The majority of active media-rich RSS feeds seem to be employed primarily for news, education, entertainment, or radio-like narrated sound collections or performances. While some may be innovative, few use the RSS format as a dedicated medium for distributing multi-part installation works. The *babelcast* attempts to expand the type and depth of content employing RSS.

4. THREE FORMATS: THE *BABELCAST*, *BABELCAST-MOSAIC*, AND *BABELCAST-ZOETROPE*

Each episode of the *babelcast* is simultaneously distributed in three formats, each with separate RSS feeds. Each of the three formats share the same audio content and have the same duration. Durations are typically near five minutes. The formats differ in file-format compatibility and the deployment and usage of images and animation. While a single RSS feed might offer all formats together, it is likely that a user will only desire to subscribe to, or have the playback capabilities for, a single format.

The *babelcast* (<http://www.flexatone.net/babelcast.xml>) is an all-audio MP3 format. Audio is encoded using the LAME encoder.

The *babelcast-mosaic* (<http://www.flexatone.net/babelcast-mosaic.xml>) is an audio and still image MPEG-4 (M4A) format. This format, currently dependent on Apple Quicktime, iTunes, or iPod playback, employs a metadata-rich format of MPEG 4 developed by Apple. This audio format embeds chapter markers, images, and links at time-points within the audio content. The markers are conventionally used for displaying album art or sectional headings in “enhanced” podcasts. For the *babelcast-mosaic*, between 30 to 50 chapters are created within the duration of the audio, and each chapter is assigned an image; the result is an aleatoric slide show. Audio is encoded using iTunes (necessary for embedding chapters), images are processed with the Python Imaging Library, and chapters are inserted with the Apple Chapter Tool.

The *babelcast-zoetrope* (<http://www.flexatone.net/babelcast-zoetrope.xml>) is a video MPEG-4 (M4V) format. This format is a standard MPEG video format and can be viewed with a variety of software. Audio and video are encoded with the free, open source FFmpeg. Video is presented in a 240 by 240 pixel square frame at 25 frames per second. Video is created from individual PNG graphic frames, each frame individually created with the Python Imaging Library.

5. THE PRODUCTION PROCESS

Each *babelcast* episode consists of media drawn from a period of days or weeks. Custom Python scripts are used to harvest audio and images from diverse sources. Both audio and images are manually filtered to select political speech and politically-relevant images of world leaders or events. Content relating to sport or entertainment, for example, is removed. While focusing the work on political content, this filtering does not favor specific political agenda: the *babelcast* does not intentionally advocate particular parties or policies.

A controlling Python script automates the production process of all formats of the *babelcast*. The process begins with the creation of the audio file. The audio file serves as a template for the creation of the image and video content of the *babelcast-mosaic* and *babelcast-zoetrope*.

The audio file is rendered by dynamically creating hundreds of athenaCL Textures [1]. Some Textures provide algorithmic treatments of one or more audio recordings of political speech. These audio files might be presented in a manner to be clearly heard, or processed so as to be difficult to distinguish as spoken language. Additional Textures add algorithmic noise layers or pitched drones. The Textures, at the lowest level, algorithmically shape and control event parameters. These Textures employ a wide variety of stochastic, Markov-based, and alternative generative techniques to control the selection of audio file regions, envelopes, filters, rhythms, pitch centers, fading, and panning. The dynamic output of these generative procedures contributes to an open work system: each

generation ensures variance but offers an audible continuity of procedural methods.

At a higher level, a repertoire of methods encapsulate diverse approaches to Texture generation. Each method may define and configure one or more athenaCL Textures. Methods assign durations, start and end times, collections of audio files to process, and additional Texture parameters. A generative system definition specifies the methods to be called and the number of times they are called. This definition is configured for each episode and is encoded as a Python dictionary. The generative system definition also defines a tempo, pitch center, total time, and metadata such as the date range of audio and image source material. The configuration of the generative system definition offers a compositional control that is intentionally exploited. By altering the number and density of Textures, or by altering the base-tempo or pitch from which other tempi and pitch-centers are derived, variety, similarity, or long-range trends between episodes can be created.

After an athenaCL Interpreter instance creates all Textures, athenaCL is used to create a Csound score, and then render that score into an audio file. At this point the controlling Python script is suspended while audio mastering is performed in a digital audio workstation. The controlling Python script is then resumed, and the encoded file formats are created.

The *babelcast-mosaic* is created by taking the total duration of the generated audio file and dividing it into an algorithmically-selected number of regions. All available image files are processed and cropped with the Python Imaging Library into a uniform frame size. These image files are saved as PNG files. An XML chapter definition file is then created, and a randomly-selected image is assigned to each time region. A Python-controlled AppleScript is used to encode the audio file as an M4A within iTunes. With the Apple Chapter Tool, this encoded audio file is combined with the XML file and the associated images to produce the enhanced M4A file.

The video for the *babelcast-zoetrope* has two states. In the first, or main state, one or two images are zoomed and panned, with the visual frame divided in two. Each image occupies one half of the frame. A black mask is applied over one of the halves. This frame may flicker, revealing the image under it. Numerous moving, perpendicular bands may also expose either the image under the mask or the alternate image. In the second, or transition state, numerous half-masked images quickly flash in sequence.

As with the *babelcast-mosaic*, the total duration of the generated audio file is divided into an algorithmically-selected number of regions. Using the Python Imaging Library, static images, as single frames, are processed one at a time in sequence. Frames within each region are reserved for transition states. After all frames have been generated, the source audio and images are combined in FFmpeg to create the final M4V video file.

After the generation of the encoded formats, the controlling Python script updates each RSS file and uploads the newly-generated content to the server. Thus, except for the filtering of source audio and image material, the configuration of the generative definition, and the audio mastering, the *babelcast* is a highly automated process. Without such automation, the mass-production of episodes would not be feasible. The aesthetic relevance and contribution of non-automated processes, versus that of automated processes, is an unwieldy problem not addressed here.

6. CONCLUSION

William Gibson's novel *Count Zero* [9] describes a machine that, from an Earth satellite, sends down to Earth randomly-constructed collages of human space junk. These collages are assumed to be forged works of artist Joseph Cornell. Cornell is a sculptor of assemblage works, works that combine and juxtapose various artifacts of contemporary life into boxes or display cases. In *Count Zero*, the search for the creator of these "forgeries" is a central plot line. The mass-production of compelling aesthetic objects from human detritus by, or with the assistance of, a machine is a secondary theme. This theme relates to the aesthetic of the *babelcast*.

The *babelcast* series is a long-term project. New techniques and approaches can be gradually incorporated into the system over the course of its evolution. In addition to expanding the sound sources and digital audio processing techniques, the integration of sound and image can be improved. The generated audio material, for example, can be analyzed and used to determine time points for image transitions. In order to provide more contrast in musical texture, multiple complete variants can be created, and then these variants can be algorithmically cross faded and alternated.

The RSS Syndication protocol permits artists to create and publish open art systems in addition to art works. Such art systems may employ audio, slide-show or flip-book-style animation, or video. Such works are integrated with their own distribution system, and offer an alternative to the distribution, in various media, of singular works. In the specific case of internet-based distribution, such mass-produced works make digital rights management (DRM) software irrelevant: the system, rather than the episode, is the work, and cannot be easily reproduced. In what might now be seen as the second, networked age of mechanical reproduction [2], the production of open musical systems, distributed via RSS or similar protocols, offers a potentially fruitful alternative to conventional fixed and reproducible media.

7. REFERENCES

- [1] Ariza, C. 2005. *An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL*. Ph.D. Dissertation, New York University.
- [2] Benjamin, W. 1936. "The Work of Art in the Age of Mechanical Reproduction." *Zeitschrift für Sozialforschung* 5(1).
- [3] Bordwell, D. 1980. "The Musical Analogy." *Yale French Studies* 60: 141-156.
- [4] Carr, D. 2001. "A Museum is an Open Work." *International Journal of Heritage Studies* 7(2): 173-183.
- [5] Cordier, P. 1982. "Chemigram: A New Approach to Lensless Photography." *Leonardo* 15(4): 262-268.
- [6] De Marinis, M. and P. Dwyer. 1987. "Dramaturgy of the Spectator." *The Drama Review* 31(2): 100-114.
- [7] Eco, U. 1984. *The Role of the Reader: Explorations in the Semiotics of Texts*. Indiana: Indiana University Press.
- [8] Eco, U. 1989. *The Open Work*. Translated by A. Cancogni. Cambridge: Harvard University Press.
- [9] Gibson, W. 1986. *Count Zero*. Arbor House Publishing Company.
- [10] Helman, A. 1983. "The Present-Day Meaning of a Work of Art." *Artibus et Historiae* 4(8): 155-164.
- [11] Hiller, L. and R. Kumra. 1979. "Composing Algorithms II by Means of Change Ringing." *Interface* 8(3): 129-168.
- [12] Johnson, M. E. 2002. "Charles Ives's (Utopian, Pragmatist, Nostalgic, Progressive, Romantic, Modernist) Yankee Realism." *American Music*.
- [13] Koenig, G. M. 1970. "Project One." In *Electronic Music Report*. Utrecht: Institute of Sonology. 2: 32-46.
- [14] Morrison, S. 1998. "Skryabin and the Impossible." *Journal of the American Musicological Society* 51(2): 283-330.
- [15] Ménard, P. 1988. "Towards a Universal and Intelligent MIDI-Based Stage System: A Composer/Performer's Testimony." *Leonardo. Supplemental Issue 1*: 63-68.
- [16] Paul, C. 1995. "Reading/Writing Hyperfictions: The Psychodrama of Interactivity." *Leonardo* 28(4): 265-272.
- [17] Tipei, S. 1989. "Manifold Compositions: A (Super)Computer-Assisted Composition Experiment in Progress." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 324-327.
- [18] Tipei, S. 2007. "Manifold Compositions: Formal Control, Intuition, and the Case for Comprehensive Software." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association.
- [19] Tyler, R. 1994. "The No Play Matsukaze as a Transformation of Genji monogatari." *Journal of Japanese Studies* 20(2): 377-422.
- [20] Xenakis, I. 1992. *Formalized Music: Thought and Mathematics in Music*. Indiana: Indiana University Press.