

# ABSTRACTION IN A UNIT-BASED AUDIO PROCESSING SYSTEM

*Thomas M. Stoll*

Buffalo, NY, U.S.A.

tms@corpora-sonorus.com

## ABSTRACT

The five principles that most completely articulate the design philosophy of corpus- or unit-based systems for organization and deployment of audio material are persistence, portability, modularity, adaptability, and abstraction. These qualities are explored in the context of one primary abstraction, namely that of the unit, the discrete entity that permeates notions of structure and binds symbolic associations, while grounding all relationships by referencing tangible material. Unit-based software design for audio is discussed along with several examples of its software implementations.

## 1. INTRODUCTION

“A double spooks the world, the double of abstraction...”[11] McKensie Wark’s emphatic interpretation fixates on the abstraction’s power to render knowledge into easily controlled commodities and the privileged hacker class upon whom—at least in his analysis—society depends to transcend, re-purpose, and otherwise re-conceptualize these abstractions. Technological progress brings with it an inevitable abundance of data and technologies developed to distill, categorize, and find patterns in raw information. Enormous amounts of resources are devoted to storing and keeping track of ever-growing stockpiles of information.

In order for computer-assisted music and art to remain relevant in light of these concerns, relationships must be forged between material, its use, and its users: the composers, engineers, and artists. In order to harness the raw power of computation to directly facilitate creative deployment of music as data, the abstraction of such must be undertaken and explored. In order to open the whole of computer-assisted musical creation to—in Wark’s terms—the hacking or creative use of patterns both existent and emergent, digital abstractions themselves must become some of the fundamental tools of composers, programmers, and artists.

In order to discuss the usefulness of abstraction applied to musical or audio material, one must be aware of knowledge gained from an array of interrelated disciplines: information theory, machine learning, computer science, etc. Relevant contributions from these domains shall be briefly summarized following the presentation of the bedrock con-

cept of the unit as it relates to the formation of data and metadata. When considered in aggregate, the following five principles form a framework for understanding collections of data and metadata as musical material, including manipulations, organizational schemes, and representations of both: persistence, portability, modularity, adaptability, and abstraction. Brief examples gathered from working systems and software developed by the author are presented throughout this text.

## 2. BACKGROUND

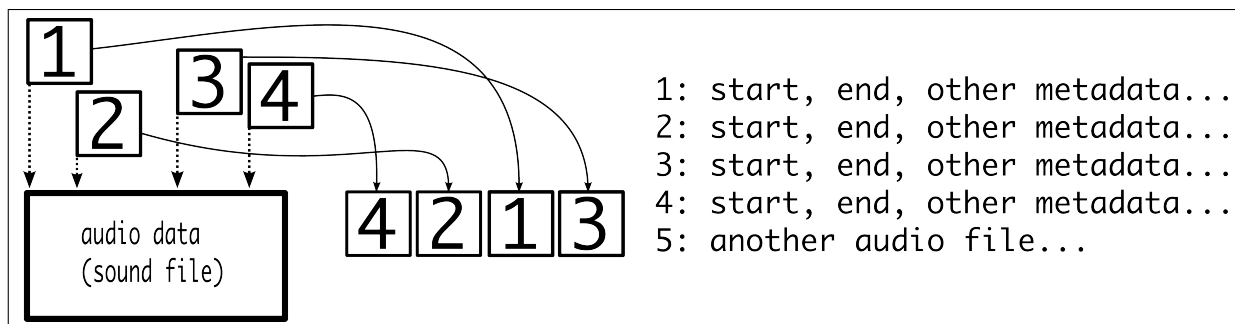
### 2.1. Units, Data, and Metadata

At the outset, a clear distinction should be made between data and metadata. Metadata gathered from direct analysis of musical data is a first order abstraction: data about data. This serves several purposes: to define or summarize the content, to do so using quantifiably less data, and to bring order to a potentially massive glut of information contained in that data. Data itself is information of the purest form, yet upon further analysis large swathes of it might be easily identified as irrelevant, redundant, too distorted, or otherwise unfit for a given task. Furthermore, a plethora of formats and modalities serve to hamper efforts to develop unified and manageable systems of organization. Units (see Figure 1) as fundamental abstractions serve three purposes within a system associating metadata to data: to standardize and define temporal limits; to make discrete what is otherwise continuous, vague, and open-ended; and to reduce material/data/sound to simple entities, thus opening any complexities hidden within to heretofore unimagined possibilities of organization and reorganization.

Implementations of corpus-based processing (CBP and CBP-sc)<sup>1</sup> developed by the author deal primarily with segments of sound data. Many of the techniques covered in this paper could serve as models for organizing non-musical audio, video, choreography, etc. The ability to freely manipulate abstracted materials does in no way guarantee that results may or may not prove interesting, aesthetically pleasing, or useful. Processes generated along the way should, at

---

<sup>1</sup>Unit-based and corpus-based are interchangeable terms; the acronyms CBP and CBP-sc will be used when referencing software implementations.



**Figure 1.** Audio data (sound file) segments are mapped to abstract units which are tagged with metadata.

the very least, engender new methods for representation, understanding, and utilization.

## 2.2. Instantaneous Versus Segmented Data

As musical sound is commonly comprehended as unfolding over time, it is important to make mention of the difference between instantaneous data points and segments that take into account duration. A single measurement of amplitude (a sample) is useless for the purpose of the present discussion. The average of a series of amplitude measurements is more in line with the experience of sound as a stream or flow of signals. This, combined with the fact that temporal limits within the human neurological apparatus (see [9] for a complete discussion) shape the experience of patterns within the changing amplitudes of sound pressure waves, suggests that analysis of sound intended to be useful to a user should first be subject to segmentation. At some point in the not-too-distant future, there may be good cause to consider computer-assisted analysis models that differ in non-trivial ways from those based on human perception, but it seems reasonable to restrict interest to those models that include segmentation.

## 2.3. Databases and Corpora: of Text and of Music

The majority of research and development work on corpora has been done in the text and text-to-speech domains. Several approaches and systems for deploying databases of segmented soundfiles have been developed, most notably mosaicing[13], Mosevius[3], Nodal[5], and CataRT[10]. CBP, an evolving open-source software project developed by the author seeks to build on these examples—most directly on CataRT, on which it was originally based—and provide a general, yet powerful platform for sound and media organization. Within such a platform, units are the fundamental building blocks of freely adaptable systems, open to integrating a wide array of design alternatives, and robust enough to prove generally useful.

## 2.4. Design Patterns: MVC, OOP, and Jamoma

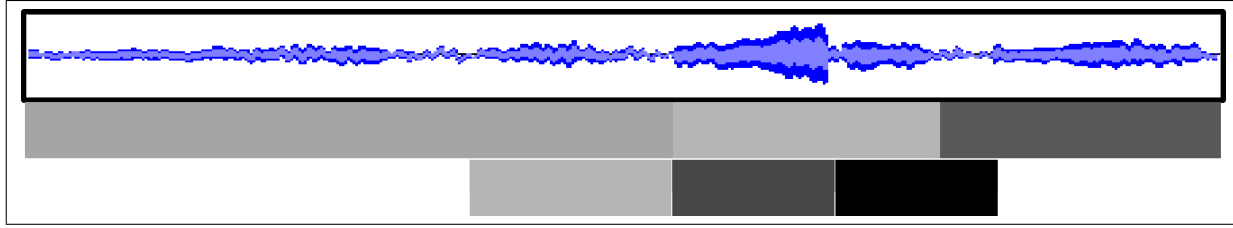
While considering the overall design of a unit-based system, several software design principles prove invaluable to bring focus to complex tasks. The Model-View-Controller (M.V.C.)[8] principle embraced by the designers of Ruby-on-Rails and others has influenced the three-way separation of (meta) data, its visual representations, and the operations that manipulate it. Object-oriented software principles[2] lend the notion of scope and modularity. Jamoma[6], and many of the principles incorporated by its designers, has strongly influenced the modular design and OSC-based intercommunication protocols [7] for CBP that not only move data between modules but also provide control at higher, more intuitive levels.

## 3. FIVE PRINCIPLES OF UNIT- AND CORPUS-BASED ORGANIZATION

Of the five major principles for unit-based system design, abstraction is the most conceptual and complements the remaining four: persistence, portability, modularity, and adaptability. These five concepts are each treated in isolation, but, when combined, complement each other in such a way as to form a more complete sense of principled design. The application of these ideas should allow the designer, composer, or user the opportunity to work at all levels of a system using consistent methodologies, as well as allow for autonomous software agents. In addition these principles are not tied to any specific domain. They may be expressed within various software paradigms or independently as design principles.

### 3.1. Persistence

Digital sound recordings are defined by the data that they embody and the sounds encoded therein. The data might be manipulated in myriad ways, but each process and subprocess should be able to provide verification in the form of a data file containing the audio after any intermediate step. Throughout the processing chain, the sound exists as



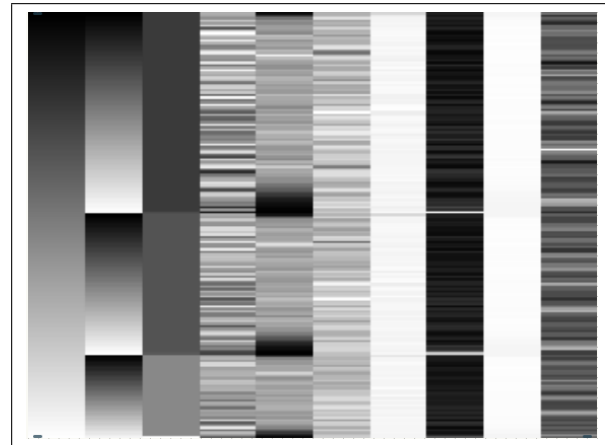
**Figure 2.** A sound file may be segmented numerous ways. These various splittings may or may not overlap.

data in the computer’s memory, and can be saved to disc. Likewise, derived data and metadata is available at points throughout a processing chain, ready to be retrieved and reused.

Temporal boundaries must be defined in order to derive and link metadata. An assortment of segments may be defined; collections of segments may be contained within segments—see Figure 2. In theory, there are no limits, neither hierarchically nor to the number of coexistent and overlapping segmentations of a given sound file. Persistence, defined as the continued existence (and relevance) of stored data and/or metadata beyond the immediate present, may extend indefinitely according to the nature of the storage medium or may expire at some definite time: the end of a performance, the end of the week, etc. Persistence itself is eligible to be encoded, stored, and referenced as a parameter, e.g., an ‘expiration date’, a ‘half-life’, etc.

Even after a unit of metadata loses its associative link to the sonic data that it describes, the essential features captured within it, as vague as they might be, remain encoded in a representation ready for use in some future process, having the potential to be rendered into sound or influence a sound-producing algorithm. In Figure 3, the “DNA”—each row represents a single unit within a corpus in CBP—of a collection of three segmented sound files is still associated with identifying units. The only factor separating data and metadata is the information lost in the distillation process. Given a relatively low degree of loss, the distinction between data and metadata blurs.

Archiving or caching certain data can even lower computational costs over time. If the raw frame data of an FFT analysis is preserved, the user is able to later (re)use this data to derive descriptors or perform other further analytical tasks. The regimented abstraction of multi-stage processes, both analytical and creative, paired with techniques to store and label the associated metadata opens every process to potentially gain from distributed and modular computing paradigms. Furthermore, compositional processes themselves and interactions by the composer or performer therewith might integrate insights from multiple runs of some sequence of operations on the same data so as to refine the result or train a machine learning algorithm.



**Figure 3.** Units retain identities even in the absence of their associated sound data.

### 3.2. Portability

While data in the form of an uncompressed audio file is standardized more or less automatically, metadata must be organized through intervention by algorithm and/or user interaction. There are several standardized data formats that are easily adaptable to unit-based organization of metadata. MIDI is a nearly universally-known method of metadata formatting. Keep in mind that despite how highly structured a MIDI file may be, it is merely a recipe laying out step-by-step instructions for recreating a sonic pattern, not sound itself. Table 1 summarizes file formats that may be used to store and transfer information.

Mostly Unstructured	Text*, comma-separated values
Flexibly Structured	JSON, SDIF*, XML*, other markup languages
Rigidly Structured	MIDI*, other score file

**Table 1.** File formats used to pass metadata among modular components of a unit-based environment. Asterisks indicate formats used in CBP and/or CBP-sc.

The simplest structure consists of lines of text or comma-separated entries that hold cells or lists of metadata. By surrounding metadata with tags that identify and/or categorize

their content, customizable and extensible formats for marking up metadata can be easily adapted. MIDI represents a closed structure that can nevertheless be used in situations where metadata is similar to the kinds of score events commonly represented by MIDI.

The following snippet is an example of unit-based metadata as XML.

```
<heading name="UNITS">
  <unit index="0">0 0 0 1 0 0 3253.107 0 1 185.153 0.855
  -42.888 0.621 0.997 0.163 3.144 0.462 0.287 0.4880</unit>
  <unit index="1">1 0 1 1 0 0 6283.333 0 1 128.587 0.484
  -25.735 0.354 1 0.082 2.804 0.367 0.172 0.0818</unit>
```

XML-parsing functionality found in most computer languages allow for versatile metadata schemes. In the following SuperCollider code, “unit” tags are scraped out of XML and stored in a unit table (“utable”).

```
xmldoc.getDocumentElement.getElementsByTagName("unit").do({
  |tag, index|
  tmp = tag.getText.split($ ).asFloat;
  tmp[0] = tmp[0] + offset;
  this[\utable].add(tmp);
});
```

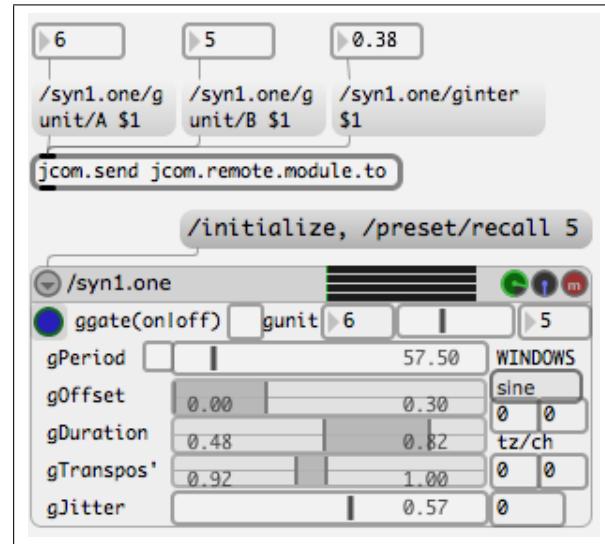
The very act of marking up metadata or storing it in clear formats affords new and unforeseen uses of the information. Use of this metadata might be made in the absence of the data itself and may be subject to further manipulation, second order metadata. Once marked-up and abstracted, this data might be passed freely among software applications and systems. Software might easily be developed that handles multiple metadata formats, autonomously parsing a range of input.

### 3.3. Modularity

Modularity isolates functionality. Not only does this make design, development, and debugging easier, it encourages reuse of code and furnishes opportunities to swap different versions or variations. An automated system acquainted with the limitations of particular potential modules might be able to choose that module which best accomplishes exigent goals. Furthermore, when functionality is clearly compartmentalized, any agent that can do the job may be safely substituted. This has implications for not only efficiency, but also for the ease with which actions take in one domain might be adapted to another. For instance, a module that constructs harmonic material using MIDI data as metadata can be then used as a model for a module that manages search pointers for selection of units that match desired harmonic patterns; notably, both modules would use identical metadata.

Figure 4 shows a Jamoma[6] module for Max 5 that exemplifies modular design. The internal functionality is controlled by messaging or direct interaction on screen. Messages can be connected directly (via patch cords) or remotely (via the “jcom.remote.module.to” mechanism) to the module. This modularity and the flexibility it provides makes the

system controllable at higher levels and therefore extremely flexible.



**Figure 4.** This Jamoma module, characterized by the specific messages to which it responds, re-synthesizes units into a stream of audio.

### 3.4. Adaptability

As structured as unit-based representations of musical material might appear to be, their real potential will only become apparent when the pairing of metadata with data is exploited and extended to carry out artistic or practical tasks. As long as there is a link to real acoustic properties, the use of algorithms to perform musically relevant tasks remains legitimately connected to perceptual results. As a trivial example, previously measured and stored amplitudes averaged over slices of audio data enable an envelope follower to perform its function without constant, costly analysis. Analysis of several parameters on multiple time scales, once stored in accessible formats, might feed information to any later processing stage.

Adaptability works globally as well as at the unit level. Units may be analyzed in groups, while relationships among groups can easily be defined across formal or temporal boundaries. Since a degree of persistence is compulsory for the unit model, and this model might be applied to any and all aspects of the compositional or conceptual design of a piece or performance, data or metadata derived at any time is available and eligible for integration at any future time. Further, this collection and reuse of material or derived material might itself become automated, woven into the very fabric of the corpus-based system at hand.

Metadata is open to redefinition and reorganization. In order to more precisely define its adaptable, malleable nature, consider how it can be classified according to its potential for transformation. Metadata might be categorized according to its potential for manipulation as follows:

1. Translatable: metadata might be reformatted, leaving its basic substance intact.
2. Static: established, but not subject to further update; locked.
3. Dynamic: subject to update and/or manipulation by some algorithmic process or user intervention.
4. Contingent: valid only under certain conditions, i.e. until some transformation takes place that renders a certain relationship invalid or illogical.
5. Virtual: metadata may be disassociated from the data from which it is derived.
6. Target: metadata might describe a search pointer or a unit for comparison.
7. Derived: interconnections are developed by observation of some rule(s) or user intervention.
8. Learned: more complex interconnections might be derived by machine learning algorithms. Machine learning techniques might be employed for their ability to develop associations where no direct analytic process would ever detect a pattern.

This taxonomy, possibly incomplete, outlines a potential for complex and dynamic interrelations between units configured based on properties encoded in parametric metadata. Systems that allow high levels of interaction amongst internal modules must include clear rules for the use of information passed among the parts. Control may both be exercised on global levels and, as in object-oriented programming paradigms, be relegated to actors with less than full knowledge of how other constituents might be using data.

### 3.5. Abstraction

Units and groups of units are powerful abstract entities, because, when combined with metadata, they allow for automated processing and organization. As stated above, metadata makes for “knowledgeable” processing algorithms with a minimum of duplicative or repeated analytic steps. Operations may be selective, forgoing action altogether under certain conditions. This general tendency to avoid redundancy reduces complexity, bandwidth, and throughput for systems as well as modules therein. In addition to providing intermediate results to be cached for future use—for example, frames of FFT data that are saved for potential future

analysis—a slight adjustment to this model serves as a potential basis for parallel and distributed processing networks of the near future.

In the current version of CBP, Ruby[4] bindings shared via Open Sound Control[12] or Adam Murray’s “ajm.ruby” JRuby bindings<sup>2</sup> allow for the storage of all derived metadata as well as the possibility to recall and process such information in meaningful and elegant ways. In the current version of CBP-sc, the SuperCollider implementation, XML and scripted processing of metadata is handled within slang. The following Ruby script is just one example of how unit-based abstractions are in no way dependent on the language or software in which they are encoded. A local copy of a corpus (object) is defined and filled with units whose ‘active’ flags are set. These units are then available to methods that manipulate corpus data, e.g., ‘get\_column’.

```
def initialize(crps)
  @local = crps
  @units_matrix = []
  @barks_matrix = []
  puts "Successfully _linked_to_Corpus: _#{ @local}"
end

def make_units_matrix(descr = 3, val = 1, bits = 16)
  arr = Array.new(bits)
  (bits - 1).downto(0) do |n|
    arr[n] = val.to_i[n]
  end
  @units_matrix = Array.new(@local.get_num_units())
  @units_matrix.each_index do |i|
    if arr[@local.get_unit(i)[descr]] == 1
      @units_matrix[i] = @local.get_unit(i)
    end
  end
end

def get_column(c)
  tmp = []
  @units_matrix.each_index do |i|
    tmp << @units_matrix[i][c]
  end
  tmp
end
```

Finally, abstraction facilitates both new uses for sonic material and the radical repurposing of metadata gleaned from one modality to another. An example already exists of unit-based transcription [1] and the author is working on unit-based sonification of biological data and applications of unit-based processing to sequences of video data. In the experience of the author, the use of abstraction as an overriding design principle influences the compositional processes that make use of corpus-based systems.

## 4. CONCLUDING REMARKS

The only conclusive statement that can be made at this point is that there are a great deal of creative uses towards which these principles can be applied. The purpose of CBP

<sup>2</sup>downloaded from and documented at <http://composition.com/web/software/maxmsp/ajm-objects> (accessed January 5, 2010)

as a software project is to create an open, working, and useful system that can be extended to work across multiple domains, both software and conceptual, where it is used to organize and deploy musical (and other) materials. This software is available for download at [www.corpora-sonorus.com](http://www.corpora-sonorus.com), and offered in the hopes that further applications might be discovered for abstraction of music, sound, and more.

## 5. REFERENCES

- [1] A. Einbond, D. Schwarz, and J. Bresson, "Corpus-based transcription as an approach to the compositional control of timbre," in *Proceedings of the International Computer Music Conference*, Montreal, Canada, 2009.
- [2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley, 1995.
- [3] A. Lazier and P. Cook, "Mosievius: feature driven interactive audio mosaicing," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, London, 2003, pp. 312–317.
- [4] Y. Matsumoto and D. Flanagan, *The Ruby Programming Language*, 1st ed. Addison-Wesley, 1995.
- [5] P. McIlwain, J. McCormack, A. Lane, and A. Dorin, "Composing with nodal networks," in *Proceedings of the 2006 Australasian Computer Music Conference (ACMC)*, Fitzroy, Australia, 2006, pp. 101–107.
- [6] T. Place and T. Lossius, "Jamoma: A modular standard for structured patches," in *Proceedings of the International Computer Music Conference*, New Orleans, USA, 2006.
- [7] T. Place, T. Lossius, A. R. Jensenius, and N. Peters, "Flexible control of composite parameters in max/msp," in *Proceedings of the International Computer Music Conference*, Belfast, 2008, pp. 24–29.
- [8] T. Reenskaug, "Administrative control in the shipyard," in *ICCAS Conference*, Tokyo, Japan, 1973, accessed January 5, 2010 at <http://heim.ifi.uio.no/~trygver/1973/iccas/1973-08-ICCAS.pdf>.
- [9] C. Roads, *Microsound*. MIT Press, 2001.
- [10] D. Schwarz, G. Beller, B. Verbrughe, and S. Britton, "Real-time corpus-based concatenative synthesis with catart," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Montreal, Canada, 2006, pp. 279–282.
- [11] M. Wark, *A Hacker Manifesto*. Harvard University Press, October 2004.
- [12] M. Wright, "Open sound control 1.0 specification," 2002, [http://opensoundcontrol.org/spec-1\\_0](http://opensoundcontrol.org/spec-1_0), accessed April 3, 2010.
- [13] A. Zils and F. Pachet., "Musical mosaicing," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Limerick, Ireland, 2001.