ACOUSMATIC COMPOSITION UNDER VERSION CONTROL: TOWARDS A NEUTRAL TRACE

Adrian Moore, David Moore
The University of Sheffield
Department of Music, Sheffield, United Kingdom
a.j.moore@sheffield.ac.uk, d.moore@sheffield.ac.uk

ABSTRACT

No cautious, creative person starts a project nowadays without a back-up strategy...it is wise to maintain a living archive of all work. [1]

Acousmatic composition begins with source material that is processed and reflected upon. Given a set of concrete sources and a set of build instructions it is theoretically possible to compile a composition from a limited number of critical initial sounds. This is analogous to the build process for software. By modifying existing compositional tools such that decisions and parameters are recorded, we construct a set of build instructions that reproduce the finished work from original source. Using version control with this methodology to track and record development it is possible to explore a trace of the composition process and use some of the features of the latest version control systems such as branching, tagging and merging. We demonstrate a methodology for composition that enables the composer to use version control effectively and in so doing enables us to more closely examine the composition process.

1. INTRODUCTION

This project grew from the contrasting work of the two authors: Adrian Moore had been interested in probing the composition process, looking at his own practice, how it had developed, looking for general trends, looking at how language and materials work but also how they are linked to process, method and working environment. Dave Moore had been developing software solutions to a variety of musical research questions ranging from sound diffusion to real-time interaction with instruments. For all his projects he used a version control system. Indeed, student courseware was stored as a read-only repository that students check-out rather than download thus allowing staff to keep the archive ‘live’. Version control is more ubiquitous than many realise as it can be seen working in any wiki. Dave wondered whether version control could be used to document the composition process. He created a prototype piece of software that sits in between a version control system and a number of programs that we use for sound manipulation. One might liken this to a ‘track changes’ toolset for composers. Even at this early stage of development it offers another toolset for experienced composers with possible export to the classroom for student composers. With some development we hope to create a tool that will prove useful compositionally and enable researchers to analyse the composition process.

2. WHAT IS VERSION CONTROL?

Version control automates the task of managing large numbers of project files often across distributed networks. It is most often used in computer software project development. The initial commit is version 1. All subsequent changes are added incrementally to the repository. Current versions commonly in use include RCS, CVS, Subversion and Git. Whilst these systems work extremely well with text files, there is a problem with version control for sound. Sound files are typically large binary files that do not compress well and although most modern VCS tools will store this data and version it, the process is inefficient. For example, a simple sound reversal will touch every single sample in the sound file and the average VCS will have little choice but to store the new file in its entirety. So can we look at our compositional process and tailor it such that the VCS can manipulate our instruction set as text? In order to consider this, we need to examine briefly the composer’s use of software.

3. COMPOSITION PRACTICE 1: PROBLEMS AND QUESTIONS

3.1. Typical practice for Acousmatic composition

Figure 1. A cyclic based listening approach.

Acousmatic composition can be examined simply from this cyclic listening based approach. During early stages of composition, the composer develops sound from initial sources with software, often in a free and open fashion (a ‘doodle’ stage) relying heavily upon a ‘gut’ reaction...
to results. Studies of composition processes have always found that note-taking during this process is time consuming and halts the flow of creativity (especially when one is working in the electroacoustic medium). When asked to reflect upon a completed work or section, one normally takes a much broader view - formal rather than structural. Moreover, we tend to add value or tangential information not necessarily conceived of at the time. There’s no harm in doing this but we are interested in capturing what was going on at the time of composition. It is reasonable to assume that we can deduce with some degree of accuracy what was in the mind of the composer by analysing the processes being used. Given a proliferation of software, graphic front ends and limitless ‘doodle-space’, it is often counterproductive for the creative artist to be confined to one particular environment. It is our aim to incorporate this flexibility but at present, we rely solely upon text-based software and this poses a number of barriers to the above mentioned freedom.

### 3.2. Issues arising from multitude of compositional approaches available to composers

- Serendipity is hard to document (and difficult to recreate!)
- A composer may be rigorous but their motives are often hidden
- Composition is often hindered by note-taking
- You only begin to probe what you do when you are asked to ‘do’ in a slightly different way. e.g. moving from graphical interface to commandline.
- Analysis of a completed work is often based on what is heard and rarely related to what was done
- Self-reflection without documentation relies too heavily upon intuition

### 3.3. Composition solutions: wrapping available tools

An earlier example highlighted the difficulties of version controlling a simple sound reversal. Instead of storing the results of the transformation we could store the instructions necessary to repeat the process. Assuming that the process is guaranteed repeatable then this transitional file can be discarded until needed for subsequent processes. If we can document the precise processes taking place throughout the composition then it’s feasible to recreate all transforms from a script. Since the transforms become transitional files we need only version the source material and transformation script. This tool already exists: make has been used for software development for some time. It reads a makefile which specifies all the constituent files and dependencies required to build a final program or library. Therefore we need to construct an environment that enables us to:

- Archive our work and help us probe our decision making processes
- Store all commands/scripts alongside the sound
- Quickly add comments, search and tag
- Work in conjunction with a version control system allowing us to snapshot and version our composition as we move forward

We currently use Csound and SoX. Both are text based and can be easily incorporated into our database, allowing additional data perhaps about construction or future development of the soundfile, perhaps describing the file over and above a file’s name. If we are to tag sounds in some way we need to work out what that data might be. As mentioned earlier a substantial amount of the ‘why’ is stored in the ‘how’ so long as someone conversant with the tools is probing the database. An interesting question that keeps occurring however is, ‘does this assume a general approach to electroacoustic composition?’

### 3.4. Identifying classes of transform

Whilst the transformation stage can essentially be seen as linear for development and combinatorial for mixing, other transforms are possible and pose additional complications.

- One to one transforms: a single input file produces a single output target (reverse, filter, reverb etc.)
- Combining transforms: multiple input files produce a single target (mixing, compositing)
- Splitting transforms: single input files produce multiple output targets, (split, channel split, split on zero crossings, frequency band split etc.)

One to one transforms are simple to reproduce through make dependencies: only one file is dependent so we need only store the transform info and the output target identifier. Combining transforms are relatively simple also: many files are required but a normal make rule can describe this. Splitting transforms are more complicated: a make rule produces a single target and cannot describe the multiple targets of a split. To deal with this our approach is to use a pre-analysis phase to create independent rules for each of the multiple outputs. For example, a stereo to mono channel split is normally achieved with a single transform that creates two files. In our system we record two transform targets for left and right respectively - the process is redesigned in such a way as to reproduce either of the targets independently. This is useful since subsequent transforms on one of the mono parts can be recreated in isolation.

### 4. COMPOSITION PRACTICE 2: DEFINING AESTHETIC PURPOSE

The problem of composition is that our knowledge and experience is nuanced to a degree that makes explanation very difficult. To say that the explanation exists and is
to be discovered in the art work is not a solution. In order to probe the link between our audition of a sound, our interpretation of it and our understanding of just how we are to use it, we suggest that composers might polarise their description of sound so as to uncover degrees of nuance, thereby more clearly defining the continua upon which sounds reside. We also suggest that composers similarly polarise their transformations and use these in a resistive fashion, working against the sound. It would be useful to probe the description of sounds against implicative manipulations. This theory assumes that numerous routes are possible but suggests that general trends can be found. Can we distinguish between an objective response to sound and a subjective, strategic and poetic intention?

Take a simple yet incomplete set of descriptors.

- reality (artificial/natural, real/unreal/surreal)
- size (big/little, heavy/light etc.)
- colour (light/dark, bold/blurred etc.)
- appearance (clean/dirty, clear/cloudy, dry/wet etc.)
- character (good/bad, brave/cowardly, enemy/friend, exciting/boring)
- scene (position such as near/far)
- energy (cold/hot, excited/calm, extreme/moderate)
- growth (departing/arriving, ascending/descending, forward/backward, etc.)

Applying these to real world situations we could generate processes as in Table 1:

<table>
<thead>
<tr>
<th>descriptor</th>
<th>process (per sound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gritty (implies broad spectrum)</td>
<td>filter (lessen), reverberation (dissipation)</td>
</tr>
<tr>
<td>Dull spectrum</td>
<td>stretch spectrum</td>
</tr>
<tr>
<td>High pitch</td>
<td>make low through pitch transposition</td>
</tr>
<tr>
<td>Low pitch</td>
<td>make high through pitch transposition</td>
</tr>
<tr>
<td>Short</td>
<td>make longer (reverberation, granulation, time stretching)</td>
</tr>
<tr>
<td>Discrete</td>
<td>make continuous through repetition and granulation</td>
</tr>
<tr>
<td>Continuous</td>
<td>make discrete through envelope shaping</td>
</tr>
<tr>
<td>Static (Mono)</td>
<td>Add motion either through panning or granulation</td>
</tr>
<tr>
<td>Spatially very dynamic</td>
<td>make mono</td>
</tr>
</tbody>
</table>

Table 1. Descriptor to process.

And when working with more than one sound we might formulate hybridised versions as in Table 2:

<table>
<thead>
<tr>
<th>descriptor</th>
<th>process (per multiple sounds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sound A works with Sound B</td>
<td>mix (vertical) or montage (horizontal)</td>
</tr>
<tr>
<td>Sound A is rich dynamic, Sound B is pitched</td>
<td>hybridise through convolution</td>
</tr>
<tr>
<td>Sound A is rich dynamic, Sound B is pitched</td>
<td>hybridise through filters (resonance or comb)</td>
</tr>
<tr>
<td>Sound A is sustained, Sound B is pulsed or angular</td>
<td>envelope follow</td>
</tr>
</tbody>
</table>

Table 2. Hybridisation procedures.

Similarly, when working with spatialisation our natural appreciation of sound may signal processes as in Table 3:

<table>
<thead>
<tr>
<th>descriptor</th>
<th>process (spatialisation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low bass</td>
<td>static position in the subwoofer</td>
</tr>
<tr>
<td>High frequency, wispy material</td>
<td>quick motion, above our heads</td>
</tr>
<tr>
<td>Thematic material</td>
<td>generally front and centre</td>
</tr>
</tbody>
</table>

Table 3. Spatialisation procedures.

There is potential for interesting research here. As has been stated, the majority of research into composition is Ex Post Facto. Storing a neutral trace of the working process (even if only a part of it) might prove a more interesting point of departure - analysing the ‘why’ through documenting and probing the ‘how’.

5. SOUND COMPOSERS MAKE (SCMAKE)

Our research centres on automating the process of creating the makefile (although in practice we have our own xml based format). *scmake* is a suite of tools built at the University of Sheffield to enable sound composers to effectively utilise a version control system. *scmake* includes facilities for building and managing a database of sound transformations while keeping track of the build dependencies. *scmake* is based on the premise that the entire composition could be remade from source on demand. All transforms (synthesis, sound edits and processes) are recorded as the composer uses them and are reproducible as and when the composer needs them. *scmake* uses SHA256 checksums to guarantee that a repeat transform is precisely the same as originally produced: the composer can rest assured that the mix s/he made yesterday is the same as the one reproduced today. However,
when coding csound it therefore becomes important to seed randomisation. The storage of all transform data allows some interesting analysis of the compositional process. We can in fact graph the chain of transformations used through simple means (Graphviz export).

![Graphviz export](image_url)

**Figure 2.** Example of Graphviz export.

### 5.1. Build tree operations

`scmake` considers the build tree as a graph of dependencies and dependants with each sound transform identified by a node in the tree. Transforms are transitional so a number of operations to manipulate the tree are provided. From any given node moving ‘up’ the dependency tree traverses towards transforms that depend on the chosen node and likewise we can traverse ‘down’ the tree towards transforms that are required by the chosen node. With these traversals in mind we consider some operations that can be performed.

- **Prune**: delete any transitional files from the selected node upwards removing database records.
- **Clean**: delete any transitional files from the selected node upwards leaving the database record intact.
- **Make**: make a single node, and implicitly create all required nodes along the way.
- **Make dependants**: make a single node and all nodes that have used it directly or indirectly.

Operations not yet implemented but considered:

- **Copy transform chain**: take a new starting node and re-apply all transforms up the chain creating a new chain.
- **Export transform chains**: export a given set of transforms into a second database.

### 6. RETURNING TO VERSION CONTROL

With version control comes additional security of logged storage which can be used quickly and efficiently. Not only is the use of this software essential in this project but it enables creative work, written texts, diagrams and other information pertinent to any project to be stored coherently. It offers the potential of being used outside its intended scope of distributed software development and affords a number of interesting ‘diversions’ that composers might find extremely useful. For example, we may branch away in our VCS and indeed take multiple paths. Each may develop in a different manner. The fact that one branch went further than the other and that we made a decision to stop is important. The Git VCS captures time elements as we make drastic modifications to our working scripts and represents a permanent memory through commits. It affords reinsertion through merge.

### 7. CONCLUSIONS

This research has demonstrated that version control can be effective to the composer but it highlights the need for new strategies in working with sound files. A compositional tool, `scmake`, has provided the means to explore these strategies. Work in this area will be ongoing with the expansion of the tool set to include graphic editing of regions from sources, and most importantly a basic mixing program. To investigate the composition process, sole use of this paradigm will be explored by composers interested in seeing the project develop.

### 8. REFERENCES