Towards a Ubimus Archaeology

Victor Lazzarini, Damián Keller

Towards a Ubimus Archaeology

Victor Lazzarini[1] **and Damián Keller**[2]

[1] Maynooth University
Maynooth, Ireland

[2]Amazon Center for Music Research - NAP
Universidade Federal do Acre - Federal University of Acre

dkeller@ccrma.Stanford.EDU

victor.lazzarini@mu.ie

Abstract. *In this paper, we attempt to lay out a new area for ubimus, which not only connects with a number of existing interests within the field, but forges new possibilities for collaboration with other disciplines. We explore the idea of ubimus archaeology from four separate but overlapping perspectives: music software, creative resources, creative models, and creative ecosystems. Following a widely-laid out characterisation of these areas, we provide a practical exploration of music software archaeology. This is focused on the rescuing of three classic music programming systems: MUSIC V, CMUSIC, and MIT-EMS Csound.*

Ubimus deals with musical material resources, activities and by-products. One of the targets of the field is the advancement of design thinking, avoiding the pitfalls of fixation on design approaches restricted to an isolated piece of hardware, an instrument, a musical genre or to the constraints implied by some practices constructed around each of those items. An important aspect of ubimus research is the collection of data through field deployments of working prototypes or systems. While this is non-trivial for current technology, it is even more difficult for legacy technology. Problems include lack of access to hardware, lack of support for software, deficient documentation, limited practical usage and limited understanding of the social implications of music practices in contexts that in some cases are incompatible with current forms of music thinking.

1. Music Software Archaeology

Is ubimus archaeology a branch of archaeology, music or computer science? If we follow Roe's [Roe 2019] reasoning, what defines the target of archaeological work is its material of study. As it is the case in music, all the archeological work done today at some point involves computers. So why would you call a field computational archaeology rather than plain archaeology? According to Roe, the object of study of the computational archaeologist is the computer program. Nevertheless, he stresses the difference between the ultimate targets of archaeology and the goals of computer science. Archaeologists use material evidence to make inferences about the past. In contrast, software archaeologists acting within the realm of computer science are mostly interested in the inner workings of software.

Despite the apparent differences, the description of software archaeology provided by Hunt and Thomas [Hunt and Thomas 2002] and the later developments of the field make it clear that understanding the context and motivations of technological design decisions may help to expand the software-oriented archaeological knowledge. If considered as a subfield of the digital humanities, software development is not only shaped by utilitarian (aka objective) goals, it is also influenced by cultural and ideological trends that may impact key design decisions. For instance, should an implementation project target open-software tools and open licences? Should it target high portability or should it prioritise ease of use? How is personal data handled? At a time of unrestricted corporate usage of personal data, these decisions have profound consequences. Therefore, software archaeology also needs to engage with the context for which the computational tools were projected.

An aspect of the archaeological field underlined by Roe has a strong resemblance with musical research trends. Roe mentions that "analytic computational archaeology makes inductive inferences from data, whilst generative computational archaeology makes deductive inferences from simulation". Analytical musical approaches tend to deal with the musical products after the creative act (but see also the exploratory approaches described below), while the music-compositional perspectives tend to involve generative techniques. The forward-looking strategies serve to prepare the terrain for the deployment of new technologies. And the backward-looking techniques help to evaluate the mismatch between the models and the collected data.

Thus, computational archaeology is firmly grounded within the digital humanities and shares some characteristics with music computing. Although the initial proposals of a software-archaeology practice could be construed as having utilitarian objectives that are mostly applicable to software-reconstruction tasks, by placing this proposal as a digital-humanities initiative the contextual aspects are highlighted. Software archaeology not only needs to engage with computer code, it also needs to take into account the culture in which the code was produced. The design decisions (materialised as software) reflect multiple negotiations among the stakeholders with contextual factors that lie beyond simple, immediate or utilitarian purposes. These factors point to ethics, aesthetics and cultural forces that constrain and shape computing.

Having set ubimus-oriented software archaeology within the context of the digital humanities, let us now deal with the potential targets of this new field. Our proposal gathers previous work in diverse contexts of musical research, including computationally oriented musical analysis, applications of prototyping and DIY design, and recent developments in ubimus theory and practice. We focus on three targets: the design and deployment of archaeological ubimus resources, the adoption of generative models as archaeological targets and the reconstruction and documentation of creative ecosystems that bring to life specimens of a lost musical past.

2. Archaeology of Creative Resources

There have been various proposals to apply computational strategies to recover traces of music making. Their methods, targets and implications for replicable music research are diverse and sometimes incompatible. Musicological approaches traditionally engage with aspects of a single musical work. This perspective has been applied in the reconstruction

of part of the 20th-century electroacoustic repertoire, including pieces such as Articulations (Ligeti 1967), Cartridge Music (Cage 1950), the ST/10 series (Xenakis 1962), Songes (Risset 1979) and many others. Despite its relevance as a working unit, the musical work cannot be adopted as the only target of ubimus-archaeological pursuits. Some forms of music making establish clear temporal and spatial boundaries to define musical experiences, but others do not. Ubimus methods strive to be genre-agnostic, i.e., the concepts and methods attempt to engage a wide range of musical practices. Given the diversity of artistic formats, settings and stakeholders encompassed by ubimus practices, other working units beyond the artwork need to be considered.

An alternative to the artwork as a target for archaeological excavations can be built around the notion of creative resource. This concept was introduced in ubimus practices to avoid the corset of the instrument as the exclusive tool for music making. Resources encompass found sounds, images, audiovisual renditions or distributed hardware that may be accessible either synchronously or asynchronously. Several characteristics of the creative resource set it apart from the acoustic instrument. Resources may be persistent or volatile [Keller 2014]. This means they can be created, shared or discarded during the activity, as it is the case in the ubimus practice of live patching [Messina et al. 2020] or in the web-based improvisations that use distributed sonic resources [Yi and Letz 2020] [Stolfi et al. 2019]. They can also be designed and deployed as reusable infrastructure, remotely accessible for community usage [Zawacki and Johann 2014]. Its persistence or volatility defines the temporal and the spatial boundaries of the creative resource, which may or may not conform to the model of synchronous, co-located music making.

Oliver's Silent Drum [Oliver 2016] provides an example of a single creative resource deployable within multiple artistic contexts. According to Oliver and Jenkins [Oliver and Jenkins 2008], in contrast to the acoustic drums, the Silent Drum Controller does nor emit audible sounds when struck. The prototype captures gestures to trigger sounds or to store the parameters for future usage. It supports up to 22 simultaneous data streams. Feature extraction is applied on continuous gestures to obtain discrete events. Apparently, this prototype fits the description of an enhanced, augmented, smart or hyper instrument as laid out by Machover and collaborators in the late 1980s [Machover and Chung 1989]. However, the deployments reported by Oliver hint that 'instrument' is not an appropriate label for its actual usage.

Oliver mentions seven artistic contexts in which the Silent Drum design was reappropriated. The ensemble Qubit replicated the prototype and composed a work for guitar, saxophone and Silent Drum. Students at the University of Lethbridge and the Texan Christian University composed scores for the Silent Drum Controller. Thai pop musician Panlertkitsakul modified the Silent-Drum design by adding elastic heads to two opposite faces of a cube-shaped box, thus calling it The Box. This prototype is played like an accordion and it is connected to a larger setup targeting techno-oriented performances. Collaborating with the Associação Recreativa Carnavalesca Afoxé Alafin Oyó, Ricardo Brazileiro modified the traditional Afro-Brazilian drum Ilú by reusing the tracking hardware and software of the Silent Drum Controller. Rather than dealing with fine motor articulation of hands and fingers, the performance artist Van der Woude's Veerkracht (Resilience) project targeted the modification of the Silent Drum prototype to play 'inside' the drum. This dance project engages the movements of the performer as a triggering

mechanism. Colombian musician Daniel Gómez's Yemas discards the body of the drum and simply stretches a fabric between two tables to represent the playing surface of the Silent Drum. The three dimensions of the hand movements are captured through a Kinect device.

The seven deployments described by Oliver highlight the flexibility of this interaction proposal, affording changes in functionality, shape and sonic qualities. He states that "as we move towards more fluid and participatory mechanisms of exchange, computer music instruments and, by default, music point to the consolidation of a global culture of appropriation and transformation that transcends the constructed boundaries of Western music and its instruments." We take the notion of a global culture with a grain of salt. The circulation of knowledge, particularly when it relates to technology, is neither evenly nor fairly spread among the central and the peripheral countries. This gap is also enforced by the cultural hegemony exerted by the large urban centres over the remote and 'exotic' locations within the countries. Thus, while erasing the artificial boundaries imposed by the acoustic-instrumental music discourse is a beneficial consequence of the ubimus archaeological initiatives, a complementary target – strongly skewing globalisation – entails enhancing the local cultural identities and fostering the coexistence of diverse cultural ecosystems. Given that degraded sonic materials circulate in some virtual communities and that they may have negative impacts on all forms of music making [Huron 2008, Truax 2015], tracking how the introduction of technology influences the level of resilience of the local cultures seems to be a pressing need before we reach a point of no return.

3. Archaeology of Creative Models

Creative resources adopted as archaeological targets may help to analyse the impact of a piece of technology on music making or may be used to compare deployments of tangible and intangible assets in various contexts – as exemplified by Oliver. Creative resources are useful units of study. But when the target is a complete musical outcome they may fall short. A complementary concept introduced by ubimus research is the relational property [Keller et al. 2015]. Relational properties emerge from usage and are usually observed through their byproducts. They are not properties of an object or properties of an agent. An advantage of targeting relational properties as complements to isolated resources is that they provide information in action. For archaeological purposes, this information may be inferred through the analysis of the sonic outcomes of the activity or it may be partially captured through visual, haptic and other streams of data. These techniques furnish insights into patterns of behaviours fostered or deterred by creative support systems. Therefore, they may also be useful as music-analytical tools [Keller and Ferneyhough 2004, Marsden 2012].

Analysis by modeling involves the reinstantiation of an algorithm by means of the archaeological traces left by its usage. These traces include but are not limited to source code, musical data, sonic renderings and various forms of documentation of the creative processes. Keller and Ferneyhough (2004) propose analysis by modeling for works that present widely varying results for each sonic realization of invariant underlying processes. This is the case in Xenakis's algorithmic music. Their approach allows to explore the parameter space of the model and the parametric combinations and interactions among models. The range of possible outcomes provides a qualitative picture of the

underlying processes. Since the published score is only a particular instance of the algorithmic mechanisms, a database of parametric combinations may be more informative for analytical purposes than the final rendition presented by the composer.

Xenakis employed his ST (Stochastic Music) program to compose a number of works, including ST/10-1 080262. The program was coded by M. F. Génuys and M. J. Barraud in FORTRAN IV on an IBM 7090 [Xenakis 1992, 145-152]. Xenakis utilized ST to generate data in text format. Then proceeded to transcribe the output to musical notation. ST provided a list of parameters such as attack time, instrument class, instrument (i.e., the selection of instruments within a given class and the playing technique), pitch, duration, dynamics and glissandi. Based on this information and on the transcriptions featured in the score, Keller and Ferneyhough were able to implement a temporal quantization model and a timbre distribution model to emulate a range of behaviours found in the piece. Two perceptual processes play an important role in auditory processing of sound textures: sequential and simultaneous stream segregation. The results indicate that grain density and timbre-based auditory streaming define whether fused granular streams could be created with the instrumental palette employed by Xenakis in ST/10. The authors raise the question of whether fused textures could have been obtained by employing slightly higher densities of events and a carefully designed timbral distribution.

Archaeologically recovered models feature some advantages over restored resources. As shown by the ST/10 exploration, through the remnants of historical deployments the output of a working prototype may be compared with the available data. This technique is aligned with the concept of scientific replicability. Consequently, the usual tools for validation can be employed. Nevertheless, a note of caution is necessary. There is an ongoing discussion in the ubimus community regarding the limits and relevance of replicability in creative practice. As highlighted by ecologically grounded creative practices [Keller 2000, Keller and Lazzarini 2017], the exact replication of a sonic output tends to reduce the creative potential of the method. So some level of uncertainty may help to boost creativity (see [Aliel et al. 2018] for applications of this perspective in comprovisation). Another feature of sonic models is their potential to update their behaviour, given environmental changes. For instance, some parametric updates may be tied to subjective choices or they may incorporate the computational analysis of the available resources, thus opening the door to the usage of automated methods that do not depend on human decisions (see [Sinclair 2018] for an implementation, and [Coelho de Souza and Faria 2013] for a creativity-oriented discussion of its usage).

4. Archaeology of Creative Ecosystems

Excavations of creative resources range from fairly straightforward cases, when they involve persistent objects (comparable to the bones recovered by zooarchaeologists), to more difficult cases, when they target volatile resources. Volatile resources may become tangible and shareable through various means, including capture, transcription and rendition in formats that may combine digital and material elements (see, for instance, the use of creative surrogates in Keller et al. 2019). With the demise of the printed score as the ideal representation of the musical piece and the demise of *the piece* as the most relevant representation of the musical experience, the replicability of a creative process increasingly depends on intangible assets (see [Bressan 2020] for the first steps toward a ubimus-oriented musicology). So model-oriented archaeological excavations may be an

alternative when the musical experience is based on volatile resources that have not been rendered as tangible assets. Model-oriented software archaeology encompasses digital and material resources to enable the rendition of sonic outcomes that are compatible with the creative target of a past experience. But what happens when the goal of the excavation cannot be identified as a creative resource or as an algorithmic model? This section deals with this question.

Ecologically grounded creative practice has been applied in artistic practice, in educational settings and in software design [Keller 2000, Keller and Lazzarini 2017, Lazzarini et al. 2020]. The influence of ecological thinking in creative practice has yielded the concept of ubimus ecosystems. Ubimus ecosystems encompass infrastructure, creative resources, stakeholders and the dynamic relationships emerging from their interactions, also known as relational properties (see previous section). Ubimus ecosystems may be based on distributed infrastructure – such as the internet of musical things – or they may use intangible assets as key tokens for exchanges among the stakeholders [Ferraz and Keller 2014]. The latter types of organizations are harder to recover and may demand working on inferences from traceable by-products. A characteristic of creative ecosystems that sets them apart from creative resources and models is their unbounded behavioral domain. Resources and models can be identified by their material qualities or by the profile of their sonic outcomes. Complex creative ecosystems may yield behaviours that are not prebuilt or projected. Therefore, an analysis of their output is not enough to identify their potential. Some knowledge of their inner workings is also necessary. An illustrative example is furnished by the music programming systems [Lazzarini 2017].

Music programming systems provide domain-specific languages dedicated to sound synthesis, processing, and the manipulation of events (at various levels from the micro to the macro). They tend to support music making in a variety of ways, with a level of transparency beyond other types of music software. Consequently, music programming languages have been adopted for artistic practices that encompass multiple aesthetic trends and have been employed both for simple utilitarian tasks and for non-trivial creative goals. Given the open nature of their outcomes, their strong bonds to a community of developers and users, and the demand for permanent adjustments to keep up with new hardware and operating systems, music-programming languages are better described as ecosystems, as opposed to fixed or bounded tools.

The three cases reported in the next section illustrate the shortcomings and advantages of targeting creative ecosystems as the objects of software-archaeological diggings. The three chosen specimens constitute historically relevant bodies of work that feature multiple contributions from continuous community efforts. Thus, they function as a repository of knowledge that is more than the sum of the isolated individual contributions. How relevant or effective is the knowledge embedded in these ecosystems given the current creative demands? Only by restoring and deploying these archaeological specimens we may eventually be able to answer this question.

5. Case Studies

As we enter the seventh decade of computer music, it is worth considering, from a software archaeology perspective, the study and preservation of systems that were fundamental for its development. Rescuing documentation, design notes, learning materials,

and music made with such software is an important part of it, but a key part of the process is to be able to recover the software itself. This would allow a study of its design, from both a computer science and a musical point of view, which may inform our understanding of music software design in general. Besides the question of actually finding the software in some machine readable format, there is the difficulty that some code may be hardware-specific, either written directly as machine code, or in some form of assembly. Understanding the software then may be tied to the knowledge of the architecture of the platform in question. The existence of simulators [Burnet and Supnik 1996], such as the one provided by the SIMH project [Supnik 2021] may be helpful in this process. If, however, the system has been written completely or partially in a recognisable high-level language, the process is reasonably more straightforward, depending solely on the availability of enough source code to allow for some reconstruction.

To illustrate in particular some of the practical points of music software archaeology, we would like to investigate a few cases. Sound synthesis and processing of digital audio signals with a computer was first achieved in 1957 by Max Mathews MUSIC I program running on the IBM704 [Lazzarini 2013], although earlier ad hoc attempts at connecting computer hardware directly to loudspeakers had been made elsewhere. At the time, Bell Labs, where this took place was the only place in the world where a digital-to-analogue converter (DAC) of the type suitable for musical signals was available [David et al. 1959]. Without such hardware, computer music using digital signals was not a practical possibility. The MUSIC I software did not support any form of programming besides taking parameters such as amplitude, frequency, and duration, and produced only single-voice sine waveforms. Following on from this, we have MUSIC II (1958), which added the possibility of other waveforms and more than one sound at a time. The breakthrough took place with MUSIC III (1961) [Mathews 1961, Mathews 1963], which could be considered the first dedicated music programming system. It comprised of a collection of BEFAC assembler macros for the IBM 7094 computer defining a few structural elements, such as the idea of an *instrument*, which was a container for *unit generators*. There were only a handful of these, and the table lookup oscillator was the central element in the system. Such a modular approach was very influential as it preceded the existence of the voltage controlled modular synthesiser by a number of years, providing a model for it. MUSIC III also introduced the principle of the *acoustic compiler*, that is, a translation program designed specifically to create sound-generating programs. These could then be fed parameter lists to produce anything from individual tones to complete compositions.

MUSIC III was superseded by MUSIC IV (1963) [Mathews and Miller 1964], a much more complete system, also written in BEFAC, but now sporting its own language. MUSIC IV included a series of preprocessing routines which facilitated the preparation of parameter lists, or *scores*. A sound processing program was compiled from MUSIC IV code, the *orchestra*, which consisted of one or more instrument definitions. As with MUSIC III, the score would instantiate these to produce the audio output samples in a digital tape, which could then be run through the DAC to produce sound. While it is not possible to recover the original code for MUSIC IV, it is possible to understand fairly well how it operated from the reference manual. Since MUSIC IV provided the model for later systems which exist in source code form, we can also surmise much of it by looking at the common elements that were passed on to this next generation of software.

We can now explore three case studies based on software whose source code is available. These systems are either first- (MUSIC V), or second-generation (CMUSIC, Csound) derivations of MUSIC IV. In this study, we are particularly interested in examining the state of the existing code, whether it runs or not, any existing documentation, and possible modifications made to it. We are not looking to make a comparative study of the software beyond the most salient aspects of their operation.

5.1. MUSIC V

MUSIC V was first developed by Mathews and his team at Bell Labs in the late 1960s as an updated version of MUSIC IV written mostly in Fortran, although it allowed the addition some machine-dependent components to be called from the main program. The version that has been rescued was typed in from a 1975 print out of the source code and adapted to work with the gfortran compiler by Bill Schottstaedt in 2008. This was fully adapted to yield a working program that can be used in a modern operating system. Additions to this restoration were made by Victor Lazzarini in 2009. MUSIC V consists (as MUSIC IV did) of separate *passes*, which may be seen as individual processing programs. These have been kept as three separate Fortran programs, which are used in sequence to produce the sound output:

1. pass1 takes a score file named 'score' and produces 'pass1.data'
2. pass2 takes a 'pass1.data' file and produces 'pass2.data'
3. pass3 takes a 'pass2.data' file and produces 'snd.raw'

Since none of the programs take any arguments, it is often better to have a shell script call them to

```
#!/bin/sh
rm snd.raw
cp -f $1 score
./pass1
./pass2
./pass3
mv snd.raw $2
```

The output is a raw, system-endian, 4-byte floating-point number at 44.1KHz (the sampling rate can be adjusted in the Fortran code).

The rescued program from 2008 did not originally contain all of the unit generators reported to exist in the first version at Bell Labs (and also none of the additions that appeared in various other installations). In particular, it lacked the `IOS` unit generator, which appears in some of Jean-Claude Risset's *Catalogue of Computer Synthesized Sounds* [Risset 1969a], most famously in his example 513, the Risset-Shepard tone instrument, based on circular pitch scales [Risset 1969b, Shepard 1964]. An assumption was made that this is a linear interpolation oscillator, and so it was added to the source code. The code also did not include four table generation routines (GEN 4 - 8), which were used in Risset's catalogue. These were added to the present 2009 version. The FLT unit generator is also not present in the code, but since we have no description of it, we can only guess it is a filter, presumably a two-pole resonator. This unit generator has not been restored yet.

Since MUSIC V was for many years used at IRCAM, it is a possibility that its site installation may have been preserved there, but we have no confirmation of this. Furthermore, it has been said that Jean-Claude Risset used the program until recently, and so his private version might also exist in some form. In any case, as far as the authors are aware of, the Schottstaedt restoration with the additions described above is the only publicly available source for MUSIC V. The code and documents is available at `https://github.com/vlazzarini/MUSICV`

5.2. CMUSIC

CMUSIC [Moore 1990], or perhaps more properly, CARL [Loy 2002], is a large computer music system comprising of several programs designed to work cooperatively within a UNIX shell environment, developed by Richard Moore, Gareth Loy, and others at the University of California, San Diego. For this, it takes advantage of the typical means of input/output (IO) proposed by that operating system, using redirection of the standard IO and piping to connect programs together. The source code is written mostly in Kernigham and Ritchie (K&R) C, which poses some challenges for any attempt to recover it within a modern system based on the ANSI C standard. Nevertheless, it has been mostly recovered in 2009 from an available archive of the CARL-0.02 linux distribution containing all of its source code. As part of this process, it was pruned of few elements that were "outdated, irrelevant, or could not be built" [Lazzarini 2021]. For this restoration, the build system was simplified to a few static Makefiles, which work correctly in modern UNIX-like systems. Part of the code is written in Fortran, but, similarly to MUSIC V, has been made to compile under gfortran. One of the components of the system is a realtime playback program called `todac`, which was based on machine-dependent code. In the restoration of the system, a cross-platform program was written to replace it.

The CMUSIC program is derived from MUSIC V, but it is much expanded and, in particular, it uses the facilities given by the C preprocessor to allow scores to use macros to simplify the code. It consists of a single pass program, which produces audio output as ASCII samples to the standard output. This is, as noted above, in line with the UNIX way of doing things. In order to produce a soundfile or to listen to the result directly, the output needs to be piped into a translation or playback program. The `tosf` program, for example, produces IRCAM-format soundfiles,

```
cmusic toot.sc | tosf -if -of -R44100 -c2 toot.irc
```

At the time of writing, some of the unit generators are producing segmentation faults due to fencepost errors. It is supposed that these may be due to incompatibilities between the original K&R C code and modern compilers, as noted by the many warnings and errors that needed to be silenced/corrected in order to build the software. It is an area for further work. Code and documentation are available at `https://github.com/vlazzarini/cmusic`

5.3. MIT-EMS Csound

Of the three case studies, Csound is the only one that continues to exist in an updated codebase [Lazzarini et al. 2016], which is now completely different to its original sources. Therefore the question is not one of rescuing the software to run existing compositions, as these can be run in the modern version of the software, but more related to use the

source code as a way of studying the internal structure of the original system. As with CMUSIC, the original Csound from the MIT Electronic Music Studios is written in K&R C, although the system is more monolithic and does not rely on the UNIX operating system particularities to work. This might be a reason why it was ported almost universally from the mid 1990s onwards, when the codebase was fully modified to comply with the ANSI-C standard by John ffitch.

Due to its K&R C codebase, it does not appear to be possible to build the MIT-EMS Csound without modifications in a modern system (e.g. with clang). Unlike CMUSIC, which was maintained possibly until the late 1990s, the codebase for the original Csound was superseded around 1993 by its ANSI-C version as noted above, thus it is over thirty years old by now (fully introduced in 1986 and publicly released in July 1987). However, it is possible to build and run the software on a VAX platform with BSD 4.3 UNIX under emulation using SIMH. Due to its licensing, it is currently not possible to make the source code available on a repository such as the ones hosted by github. However, disk images and instructions for the SIMH VAX emulator can be obtained from the authors.

6. Conclusions

In this paper, we examined the ideas around what we have termed ubimus archaelogy, which comprises three inter-related areas: music software archaelogy, archaelogy of creative models, and archaelogy of creative ecosystems. We have given a wide overview of these areas, highlighting the relevant elements within the scope of ubimus. Completing the paper, we provided three practical case studies of music software archaelogy, demonstrating some of the ideas put forward earlier. It is hoped that this first attempt at characterising an archaeological approach to ubimus may inspire some further work in the area.

References

Aliel, L., Keller, D., and Costa, R. (2018). The maxwell demon: A proposal for modeling in ecological synthesis in art practices. *Música Hodie*, 18(1):103–116.

Bressan, F. (2020). Interactive systems and their documentation. In Lazzarini, V., Keller, D., Otero, N., and Turchet, L., editors, *Ubiquitous Music Ecologies*, pages 87–100. London: Routledge.

Burnet, M. and Supnik, B. (1996). Preserving computing's past: Restoration and simulation. *Digital Technical Journal*, 8(3):23–38.

Coelho de Souza, R. and Faria, R. R. A. (2013). Creativity as an emergent property in algorithmic composition. In Keller, D., Quaranta, D., and Sigal, R., editors, *Sonic Ideas*, volume Musical Creativity / Criatividade Musical / Creatividad Musical. Morelia, Mexico: CMMAS.

David, E. E., Mathews, M. V., and McDonald, H. S. (1959). A high-speed data translator for computer simulation of speech and television devices. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*, IRE-AIEE-ACM '59 (Western), page 169?172, New York, NY, USA. Association for Computing Machinery.

Ferraz, S. and Keller, D. (2014). Preliminary proposal of the in-group, out-group model of collective creation. *Cadernos de Informática*, 8(2):57–67.

Hunt, A. and Thomas, D. (2002). Software archaeology. *IEEE Software*, 19(2):20–22.

Huron, D. (2008). Lost in music. *Nature*, 453:456–458.

Keller, D. (2000). Compositional processes from an ecological perspective. *Leonardo Music Journal*, 10:55–60.

Keller, D. (2014). Characterizing resources in ubimus research: Volatility and rivalry. In *Proceedings of the V Workshop in Ubiquitous Music (V UbiMus)*. Vitória, ES: Ubiquitous Music Group.

Keller, D. and Ferneyhough, B. (2004). Analysis by modeling: Xenakis's st/10-1 080262. *Journal of New Music Research*, 33(2):161–171.

Keller, D. and Lazzarini, V. (2017). Ecologically grounded creative practices in ubiquitous music. *Organised Sound*, 22(1):61–72.

Keller, D., Otero, N., Lazzarini, V., Pimenta, M. S., Lima, M. H., Johann, M., and Costalonga, L. L. (2015). Interaction aesthetics and ubiquitous music. In Zagalo, N. and Blanco, P., editors, *Creativity in the Digital Age*, Series on Cultural Computing, pages 91–105. London: Springer.

Lazzarini, V. (2013). The development of computer music programming systems. *Journal of New Music Research*, 4(42):97–110.

Lazzarini, V. (2017). *Computer Music Instruments*. Berlin and Heidelberg: Springer.

Lazzarini, V. (2021). CMUSIC – readme. `https://github.com/vlazzarini/cmusic/`.

Lazzarini, V., Keller, D., Otero, N., and Turchet, L., editors (2020). *Ubiquitous Music Ecologies*. London: Taylor & Francis (Routledge).

Lazzarini, V., Yi, S., ffitch, J., Heintz, J., Brandtsegg, Ø., and McCurdy, I. (2016). *Csound: A Sound and Music Computing System*. Springer, Berlin.

Loy, G. (2002). The carl system: Premises, history, and fate. *Computer Music Journal*, 26(4):52–60.

Machover, T. and Chung, J. (1989). Hyperinstruments: Musically intelligent and interactive performance and creativity systems. In *Proceedings of the International Computer Music Conference (ICMC 1989)*, pages 186–190. Ann Arbor, MI: MPublishing, University of Michigan Library.

Marsden, A. (2012). "what was the question?": Music analysis and the computer. In Gibson, L. and Crawford, T., editors, *Modern Methods for Musicology: Prospects, Proposals and Realities*, Digital Research in the Arts and Humanities, chapter 10, pages 137–153. London: Ashgate Publishing.

Mathews, M. (1961). An acoustical compiler for music and psychological stimuli. *Bell System Technical Journal*, 40(3):553–557.

Mathews, M. (1963). The digital computer as a musical instrument. *Science*, 183(3592):553–557.

Mathews, M. and Miller, J. E. (1964). *MUSIC IV Programmer's Manual*. Bell Telephone Lab, Murray Hill, N.J.

Messina, M., Svidzinski, J., de Menezes Bezerra, D., and Costa, D. F. (2020). Live patching and network interaction: An intercontinental practical approach with kiwi. *International Journal of Digital Media and Interaction*, 3(5).

Moore, F. R. (1990). *Elements of Computer Music*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Oliver, J. and Jenkins, M. (2008). The silent drum controller: A new percussive gestural interface. In *Proceedings of the 34th International Computer Music Conference (ICMC 2008)*. Belfast: Queens University Belfast.

Oliver, J. E. (2016). Design and appropriation in open source computer musical instruments: A case study of the silent drum. *eContact*, 18.

Risset, J. C. (1969a). *An Introductory Catalogue of Computer Synthesized Sounds*. Bell Telephone Lab, Murray Hill, N.J.

Risset, J. C. (1969b). Pitch control and pitch paradoxes demonstrated with computer-synthesized sounds. *Journal of the Acoustical Society of America*, 146:88.

Roe, J. (2019). Computational archaeology, digital archaeology, and associated ologies. Joe Roe Blog. `https://joeroe.io/2019/07/19/computational_archaeology.html`.

Shepard, R. (1964). Circularity in judgments of relative pitch. *Journal of the Acoustic Association of America*, 36(12):2346–2353.

Sinclair, S. (2018). Sounderfeit: Cloning a physical model using a conditional adversarial autoencoder. *Musica Hodie*, 18(1):44–60.

Stolfi, A. S., Milo, A., and Barthet, M. (2019). Playsound.space: Improvising in the browser with semantic sound objects. *Journal of New Music Research*, 48(4):366–384.

Supnik, B. (2021). Simh. `http://simh.trailing-edge.com/`.

Truax, B. (2015). Paradigm shifts and electroacoustic music: Some personal reflections. *Organised Sound*, 20:105–110.

Xenakis, I. (1992). *Formalized Music: Thought and Mathematics in Composition*. Harmonologia series. Hillsdale, NY: Pendragon Press.

Yi, S. and Letz, S. (2020). The browser as a platform for ubiquitous music. In Lazzarini, V., Keller, D., Otero, N., and Turchet, L., editors, *Ubiquitous Music Ecologies*, pages 185–206. London: Routledge.

Zawacki, L. F. and Johann, M. O. (2014). Analogue audio recording using remote servers. In Keller, D., Lazzartini, V., and Pimenta, M. S., editors, *Ubiquitous Music*, pages 83–107. Berlin and Heidelberg: Springer.