COMPOSING WITH SOUNDS: DESIGNING AN OBJECT-ORIENTED DAW FOR THE TEACHING OF SOUND-BASED COMPOSITION

Stephen Pearse
University of Portsmouth
stephen.pearse@port.ac.uk

Leigh Landy
De Montfort University
landy@dmu.ac.uk

Duncan Chapman
Independent Composer
dchapmanhoot@googlemail.com

David Holland
De Montfort University
dholland@dmu.ac.uk

Mihai Eniu
University of Portsmouth
mihai.eniu@port.ac.uk

ABSTRACT

This paper presents and discusses the Compose With Sounds (CwS) Digital Audio Workstation (DAW) and its approach to sequencing musical materials. The system is designed to facilitate the composition within the realm of Sound-based music [1] wherein sound objects (real or synthesised) are main musical unit of construction over traditional musical notes. Unlike traditional DAW's or graphical audio programming environments (such as Pure Data, Max MSP etc.) that are based around interactions with sonic materials within tracks or audio graphs, the implementation presented here is based solely around sound objects. To achieve this a bespoke cross-platform audio engine known FSOM (Free Sound Object Mixer [2]) was created in C++. To enhance the learning experience, imagery, dynamic 3D animations and models are used to allow for efficient exploration and learning. All tools within the system are controlled by a flexible permissions system that allows users or workshop leaders to create sessions with specific features based on their requirements. The system is part of a suite of pedagogical tools currently in development for the creation of experimental electronic music.

1. INTRODUCTION

The Compose with Sounds (CwS) software package (see Fig. 1) was born out of two distint ideas and projects. Landy's highly influential *Making Music with Sounds* [3] made strives in attempting to fill the enourmous gap in literature regarding sonic creativity for novices and school teachers. For several years, Landy had been working on the ElectroAcoustic Resource Site (EARS [4]). In 2006 the EARS website EARS was supported and adopted by UNESCO becoming a node of their DigiArts programme. After a period of collaboration with the body, they requested 'an EARS for kids'. This subsequently became EARS 2 [5, 6]. As children prefer to be working actively whilst learning as opposed to being solely fed information, the EARS 2 pedagogical site needed to be highly interactive. Furthermore, it demanded a creative dimension that could not be integrated into the site. This provided the opportunity to develop a bespoke software package, not only for EARS 2 but for any inexperienced user in the realm of sound-based composition also known as electroacoustic music.

For many years members of the project team have been working across a wide variety of sound projects in educational and community settings. The target age group for much of this work has been UK Key Stage 3/international middle school students aged 11-14.Within these contexts, it has become rather clear that there were no real software packages that present a simple way of working with recorded audio that does not require specialist knowledge or experience on hand. Projects such as Sonic Postcards [7], typically resorted to using Audacity as an audio editor and composition environment. The main drawback of this was that it was very hard for those without experience to fully comprehend what they were working on. Using Audacity also assumed that users were able to navigate their way round the computer to find the sounds they were going to use. In this setting, participants would often get "lost" and lose interest. Software like Audacity presents a vast number of "choices" regardless of the level of experience of the user. Similarly, when participants have been presented with other "entry level" tools (such as Garage Band) large numbers of them would spend the majority of their time "auditioning" sounds and presets, leaving little time to actually compose with them.

The CwS software was subsequently designed to be as intuitive as possible while avoiding the interface or auditioning shortcomings of existing tools. The core target age group of the work remains students aged 11-14 but it is flexible and approacheable enough to be used by students either side of this age bracket. The software aims to support students to a level wherin they can easily move onto other platforms such as Logic, ProTools or even Max MSP and the like, while being easily linked to challenges within EARS 2 projects. Like the aforementioned platform, the software is intended to be multilingual for adoption in cultures around the globe. CwS was first supported by an EU Culture grant, 'Composing with Sounds' with further development being supported by a Creative Europe grant for the 'Interfaces' project [8, 9].

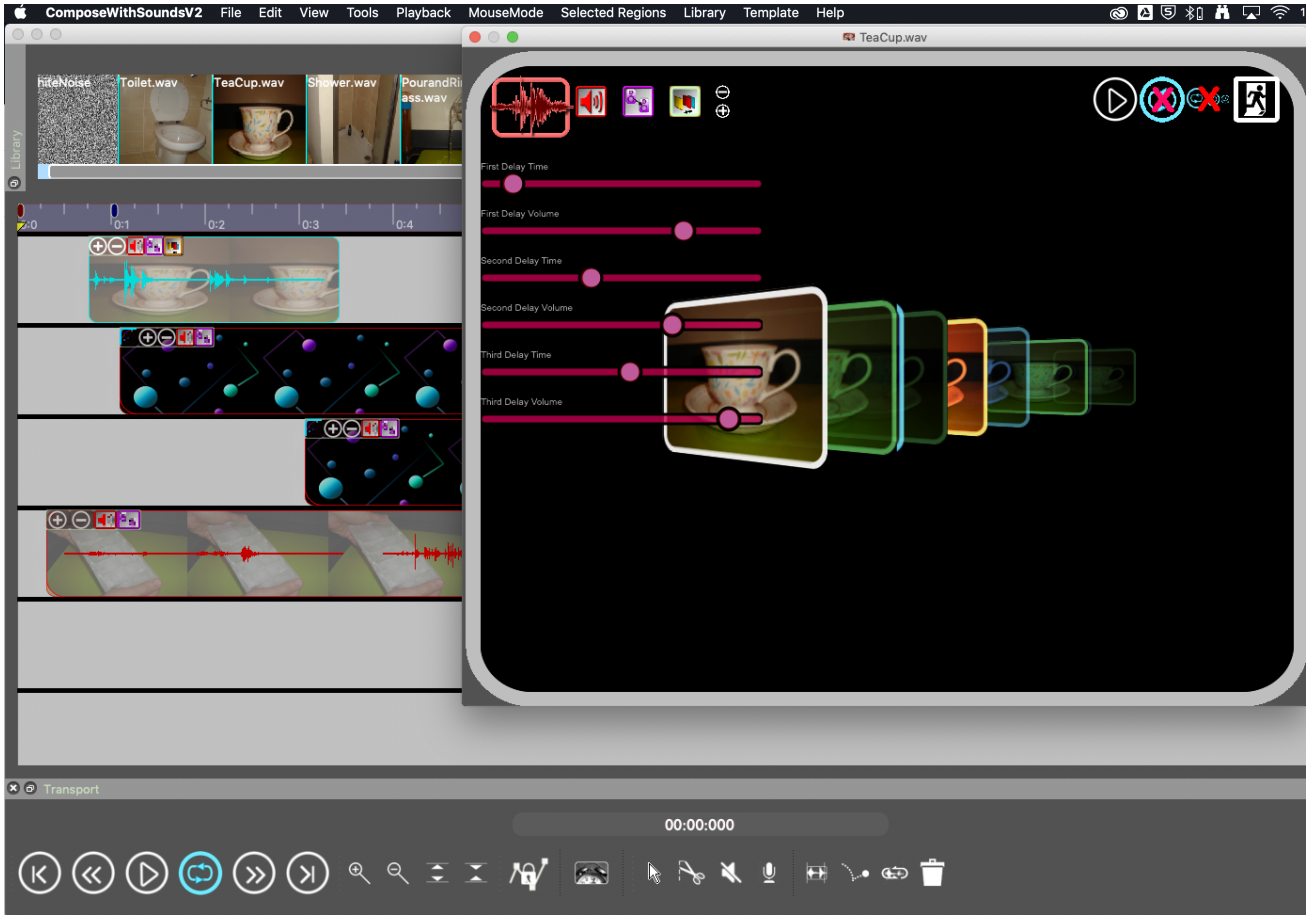

Figure 1. Compose with Sounds Version 2.36

To increase access and potential engagement with the tool and the EARS 2 platform on the whole, the system is scheduled for free release in late spring of 2019 for MacOS 10.9+ and Windows 10.

2. TECHNICAL OUTLINE

To enable the development of CwS, a bespoke audio engine known as the Free Sound Object Mixer (FSOM [2]) was engineered. This open-source audio engine was designed to be flexible and suit different iterations of the Compose With Sound project but also be easy to repurpose for other real-time audio applications. With a project of this scale great care and attention needed to be taken to ensure that the functionality in FSOM was not dependent on the CwS in any shape or form. While keeping the audio and graphic threads separate is widely encouraged in the audio development community, here it is fundamental to the architecture of the project.

Throughout the project, CwS has gone through two significant iterations which entailed a complete rewrite of the graphical user interface. Versions 1 through to 1.35, all utilised the cross-platform windowing library wxWidgets [10] and was made available explicitly for Windows 7 and MacOS 10.7 in 2015. To ensure future support for higher resolution displays and stability on newer versions of MacOS, the entire graphical user interface was re-written to utilise the popular windowing library, Qt [11]. For clarity and integration with other iterations of the software and other projects, all materials that are saved by the system (be it the session itself, library information or template data) are all saved in an easy to read and interpret XML schema.

2.1 Development Cycles

To tailor CwS to the target audience in the best way possible and ensure that it be delivered efficiently, an adapted form of the SCRUM development methodology [12] has been used. Software development iterations are based on development sprints lasting between two to four weeks. This flexibility is a necessity as testing and feedback across multiple levels is continuously required to ensure stable development of the software. Since commencing the development of version 2.0, the codebase has been maintained by a single software developer (the lead author). Due to the size of the codebase and the aims of the project, several iterations of testing and feedback take place to identify bugs and feature requests in each private release prior to wider distributions. The software and each new feature is subsequently tested by three distinct groups, each with finite roles (see Fig. 2). These groups are the developer and students at their institution, the wider project team across the interfaces project and workshop participants. The developer is responsible for testing new functionality; the wider team, bugs and general usability issues

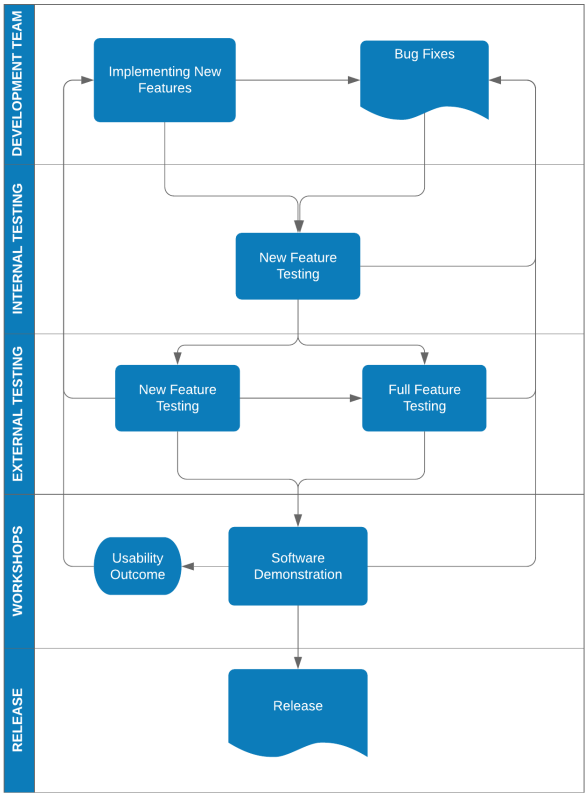

Figure 2. CwS Testing Cycles

and workshop attendees for broader usability issues alongside requesting new features. This approach ensures that if issues or queries concerning the usability emerge, workshop leaders and testers could be informed and guided to test and query functionality.

Many features within the software were subsequently informed through this approach. Perhaps the most notable of these are the graphical tracks alongside the mute and solo tools. As the system does not depend on a traditional track-based architecture (see section 5), sounds can be placed over one another in the sequencer environment with both being heard. Conceptually several users struggled with this methodology and leading to confusion with sounds of different lengths being laid over the top of one another. The software dynamically sorts and alters the rendering z depth of each sound based on their duration with the shorter sound materials being drawn on top of those with longer durations. While this is effective, user feedback indicated that limited graphical "tracks" would be needed to ease the organisation of sessions. After a series of workshops in the summer of 2018, many users requested that these "tracks" offer solo and mute functionality. While this was possible to achieve within the codebase, it was not in keeping with the sound-based nature of the project. As such, mouse tools were created instead for muting and soloing multiple sounds at once within the system.

3. THE SOUND CARD

With sounds being affiliated with imagery, the metaphor of a card or soundcard is used within the software. While

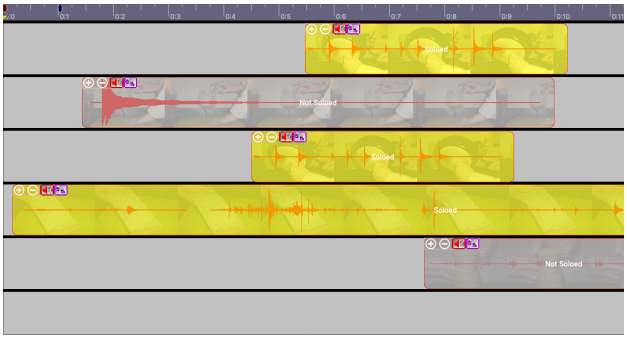

Figure 3. Solo and mute functionality

the pairing of image to sound in this form is encouraged, it is by no means compulsory. At any given moment, a user can alter the view of cards within the system to show the waveform of the sound, the assigned image or both.

A composer's primary means of working within the system is the sequencing and transformation of these cards. The software subsequently contains a suite of standard editing tools to be utilised. This includes the aforementioned solo and mute tools (see section 2), deletion, splicing, truncation, file bouncing and time-stretching [1]. For ease of use, each of these can be undone and applied to multiple cards at any given time. As well as supporting copy and pasting of cards as a user would expect, the system affords different approaches to duplicating and looping cards. Standard duplication is supported wherein a new card is created immediately after the original has finished. When multiple cards are selected at once, the newly duplicated cards commence at the end of the final original card. These new cards retain the same timing differences as the original selections. Cards can also be duplicated to behave like traditional looping functions found in commercial DAW's wherein cards are repeated upon their owen completion. This flexibility in duplication subsequently enables users to easily create structural relationships based on relative timing or more rigid loops if desired.

3.1 Synthetic Cards

Alongside cards made from audio files on the user's computer, the system also contains synthetic/generative cards that can be utilised. At the time of writing, these cards consist of a noise card, an additive synthesis card and a granular synthesis card. The number of controls available for the latter two types was purposefully limited to reduce the threat of creative paralysis when confronted with an extensive collection of controls. The former is limited to four oscillators with pitch, amplitude and shape controls (sine, cosine, noise, square, saw and triangle). The granular synthesis region is based on traditional file-based granular synthesis [14] over live input based tools akin with Mutable Instrument's Eurorack Module, Clouds [15]. This card subsequently provides controls for the grain size, pitch, position in the source file, the rate of spawning, amplitude and playback speed (used to scroll through the sound at varying

[1] The time-stretch tool is based around a custom phase vocoder inspired by the algorithm used in PaulStretch [13]

rates). While random modifiers could have been provided for each of these controls, they have been purposefully removed at this point.

3.2 Card Libraries

As the system is based around the usage of sonic cards, the traditional approach of using an audio pool has been replaced by the use of card libraries. This library system is designed to utilise a clear and concise XML schema alongside a simple file structure rather than serialising the data into binary. While previous research [16] has shown the inefficiency of XML under certain circumstances, it has been utilised here to ensure easier integration with affiliated projects and tools. The system allows users to easily import and export libraries either as standalone entities or as part of saved compositions. A selection of card libraries is provided with the software. A further collection of libraries can be found on the original CwS website [17] and the EARS 2 platform [5, 6] which contains teaching materials for the system. Most significantly, it affords the potential for users to interactively explore the libraries on the accompanying website.

At present, the software does not allow users to record sounds within it. This was an active decision made by the team to ensure that the software is used primarily for sound sequencing and transformation. A proof of concept library creation application for desktop and mobile applications has been created using JUCE [18]. This tool allows users to create cards through recording or importing audio and photos on their device of choice. Libraries can then be exported from this tool.

3.3 Templates and Levels

The system contains a flexible permissions framework that enables or disables access to features within. Akin with Boden and later Magnusson's work on constraints [19, 20], limiting or guiding users through a limited set of tools (initially) forces them to think creatively about what each tool affords. This subsequently encourages them to internally map out creative possibilities that they wish to explore. The approach taken in CwS supports this further as it enables educators to create their own scaffolding structures [21] for audio tools and effects. These structures are known as templates within the software and can be freely created by the user. This augments Bigg's constructivist theory [22], where users can actively seek and creatively apply their own pathway through the tools available. Templates can be applied to any compositional session or library that the system is aware of. When the student in question is ready, educators can subsequently move to the next relevant template. The system is provided with three linear templates that can be used. An outline of what is provided at each level can be seen in Table 1. To aid the delivery of workshops and other guided learning sessions, templates are saved into libraries when they are exported if required. This enables workshop leaders to design sessions that prescribe the sound materials and tools that can be easily deployed.

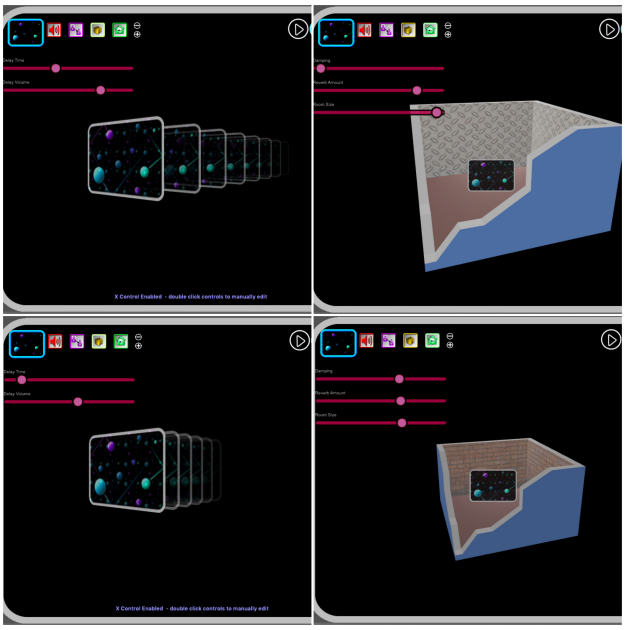

Figure 4. Delay and Reverb representations

4. DIGITAL AUDIO EFFECTS AND TRANSFORMATIONS

The system contains a collection of standard and extended audio effect processors (see Table 1) that composers of sound-based music would expect to find in a DAW of their of choice or would have access to through commercial/free audio plugins. The system does not allow external plugins to be loaded into it for several reasons. In a classroom context, deploying, loading and working with plugins can slow down sessions due to the risk of technical or licensing issues occurring. This is especially the case in today's climate where some plugins require online activation or verification at runtime. When learning a new piece of software alongside a new approach to music and sound, there is the potential for any student to be overwhelmed with the quantity of features available. If this experience requires users to learn the control and interface vocabularies of different plugin manufacturers on top of this, the chance of creative paralysis may increase. Based on feedback from members of the project team, it was decided that a suite of effects should be designed with a consistent interface to ease students development.

Unlike traditional track-based DAW's, each of the effects contained within the software exists entirely on a sound card[2]. To aid the learning experience when using these effects, each effect processor is accompanied by an interactive 3D animation that updates based on the effects settings (see Fig. 4). The animations provided utilise the card metaphor and presents visible transformations that symbolically, and in some cases, literally reflect the effect process. Effects such as reverb and the collection of delay based effects evidence this divide. The reverb effects present the card within a virtual room where the size, brightness and wall texture alter based on parameters such as room

[2] Automation for parametric changes subsequently exists within a sound card itself.

Level 1	Level 2	Level 3
Audio	All Level 1 features	All Level 2 features
Band Pass Filter	Band Reject Filter	Additive synthesis
Delay	Distort	Asymetric Delay
Fade	Envelope	Automation
Gain	Harmoniser	Chorus
High Pass Filter	High Amplitude Modulation	Flanger
Loop	Low Amplitude Modulation	Granular Synthesis
Pan	Low Frequency Modulation	Ring Modulation
Reverse	Timestretch	
Simple Reverb	White Noise	
Transpose		
Truncate		

Table 1. Learning Levels by Template in CwS

size, wet/dry and damping. Delay-based effects present repeating cards drifting into the distance based upon the algorithm being used. Across all of these, the delay time is represented as the distance between the cards and the feedback alters changes in brightness across this spread. The asymmetric (multitap) delay presents each tap with a card whose colour palette has been altered while the chorus and flanger both update the separation of the cards based on the modulation source in real-time.

5. SEQUENCING

The core audio processing architecture within the system deals with discrete regions. These regions contain all of the audio material for a given card in the software. This approach is unlike traditional DAW designs that will process audio as an array/vector of tracks, each with content within, or as an audiograph with audio nodes that are connected. As there is the potential for hundreds of discrete sounds being active at any given point, careful optimisations are needed to ensure consistent performance on computers with widely different specifications. While every region could be stored in some form of an array with each deciding when they should process their audio (based on the current play-head position and whether it is bypassed), this is dependent on a large number of conditional tests and instructions per sample. While this might not be a problem on contemporary hardware, it may introduce a substantial performance bottleneck on older machines that may be found in schools. To overcome this, the audio engine creates a cache of all of the region based events within the current session. Each event has a type, the region it is associated with along with the sample time that it occurs. In the real-time audio thread, a list of active regions is utilised. Only regions within this list are processed when the sequencer is active. In this thread, the event type information is used to decide whether a region should be added, removed or left unchanged in the active region list.

Given the region based nature of this processing, the use of time-based effects (such as delays and reverbs) can be problematic with the decaying effect for a region potentially being cut off as the regions timing events dictate that it has finished playing. To overcome this, CwS uses impulse responses to calculate the decay time of a given card featuring time based effects. This timing information is then used to extend the processing time of the given region to ensure that the decay is never cut off.

6. WORKSHOPS

The software package has been presented at numerous workshops across multiple countries in the European Union. Feedback on software tends to be overwhelmingly positive due to the streamlined user experience through working with cards. Participants and workshop leaders have noted that the software is excellent as an introductory tool for engaging sonic exploration and creativity for four key reasons.

Firstly, the card system enables people to clearly see and arrange the sounds they are working on without needing to use linguistic markers. Some workshop leaders stressed the importance of allowing students to work with drawn images of sounds. In this context, being able to have a card with an image enables a wide range of participants from the very young upwards to intuitively use the software.

Secondly, projects such as Sonic Postcards [7] alongside the work of Holland [23] present a strong case for the further development of student listening activities through field recording expeditions with groups of participants. The card system means that recordings from several groups of participants can be easily combined into a set of sounds for them to compose with.

Thirdly is the level/template system which many facilitators have responded positively to. By not presenting a long list of options to a user, participants can spend more time making music and less time choosing effects or processes. Teachers have welcomed the fact that in CwS, the children are gradually introduced to more tools as they work through the levels. CwS encourages users to explore each effect in turn and develop understanding before they move to the next level with a teacher in support.

Finally, commentators have noted the visual models within the system. Upon illustrating the software and similar tools in workshop contexts, participants are often drawn to similar starting points/processes. Delay (and echo), Pitch shifting (and harmonising), Reverse and Reverb are often the

"go-to" processes. Great care has been taken with the visual models for these processes. Participants have almost universally praised their clarity, reducing the need for lengthy explanations. Several practitioners have highlighted the reverb model being particularly helpful in visually linking the room size and the surface material of the room with sonic changes when using the controls. Such an approach is a noticeable contrast to software packages such as Audacity that is often used in schools by workshop leaders. All of these approaches subsequently encourage focused listening and developmental exploration and creativity with the tools available in the software.

7. FUTURE DEVELOPMENTS

While ongoing production and small enhancements for CwS are planned across Windows and Mac OS, two significant developments are currently underway under the umbrella of the Interfaces project. The first of these is the addition of an audio visual layer to the software that would allow users to sequence music to video. This opens up the system to a broader audience interested in sound for moving image and sound design. The second major development is the implementation of the Compose with Sounds Live (CwS Live) platform. This tool aims to expose students to the world of mixed media compositon and performance via the transformation of live audio input alongside the triggering of recorded materials. The system is also designed to encourage collaborative performance. To do so, the software allows control data to be sent to other instances of the itself on a local area network via UDP.

Acknowledgments

The authors would like to thank their respective institutions and the European Commission's Creative Europe programme for supporting the ongoing development of the project. They would also like to thank Dr. David Moore and David Devaney for their contributions to the Free Sound Object Mixer and the CwS Version 1.0.

8. REFERENCES

[1] L. Landy, *Understanding the Art of Sound Organization*. Mit Press, 2007.

[2] S. Pearse, D. Moore, and D. Devaney, "Free sound object mixer," https://github.com/spearse/FSOM, Aug 2015, accessed: 2019-03-07.

[3] L. Landy, *Making Music with Sounds*. Routledge, 2012.

[4] ——, "Ears : Electroacoustic resource site," http://www.ears.dmu.ac.uk/, accessed: 2019-03-07.

[5] ——, "Ears 2 : Electroacoustic resource site," http://ears2.dmu.ac.uk/, accessed: 2019-03-07.

[6] L. Landy, R. Hall, and M. Uwins, "Widening participation in electroacoustic music: The ears 2 pedagogical initiatives," *Organised Sound*, vol. 18, no. 2, pp. 108–123, 2013.

[7] Sound and Music, "Sonic postcards," http://www.soundandmusic.org/projects/sonic-postcards, accessed: 2019-03-07.

[8] "Interfaces network," http://www.interfacesnetwork.eu/, accessed: 2019-03-07.

[9] "The interfaces project," http://www.interfaces.dmu.ac.uk/, accessed: 2019-03-07.

[10] "wxwidgets," https://www.wxwidgets.org/, accessed: 2019-03-07.

[11] "Qt libraries & apis, tools and ide," https://www.qt.io/, accessed: 2019-03-07.

[12] K. Schwaber, "Scrum development process," in *Business object design and implementation*. Springer, 1997, pp. 117–134.

[13] P. Nascar Octavian, "Paulstretch: Paul's extreme sound stretch," http://hypermammut.sourceforge.net/paulstretch/, accessed: 2019-03-07.

[14] C. Roads, *Microsound*. MIT press, 2004.

[15] E. Gillet, "Mutable instruments — clouds," https://mutable-instruments.net/modules/clouds/, accessed: 2019-03-07.

[16] K. Maeda, "Performance evaluation of object serialization libraries in xml, json and binary formats," in *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*. IEEE, 2012, pp. 177–182.

[17] C. with Sounds Team, "Compose with sounds," http://www.cws.dmu.ac.uk/, accessed: 2019-03-07.

[18] Roli, "Juce," https://juce.com/, accessed: 2019-03-07.

[19] M. A. Boden, *The Creative Mind: Myths and Mechanisms*. Routledge, 2004.

[20] T. Magnusson, "Designing constraints: Composing and performing with digital musical systems," *Computer Music Journal*, vol. 34, no. 4, pp. 62–73, 2010.

[21] D. Wood, J. S. Bruner, and G. Ross, "The role of tutoring in problem solving," *Journal of child psychology and psychiatry*, vol. 17, no. 2, pp. 89–100, 1976.

[22] J. Biggs, "Enhancing teaching through constructive alignment," *Higher education*, vol. 32, no. 3, pp. 347–364, 1996.

[23] D. Holland, "A constructivist approach for opening minds to sound-based music," *Journal of Music, Technology & Education*, vol. 8, no. 1, pp. 23–39, 2015.