Multimodal Technologies and Interaction

MDPI

Article

Using High-Performance Computers to Enable Collaborative and Interactive Composition with DISSCO

Sever Tipei [1,*], Alan B. Craig [2] and Paul F. Rodriguez [3]

[1] Computer Music Project, University of Illinois at Urbana-Champaign, Champaign, IL 61801, USA
[2] National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Champaign, IL 61801, USA; a-craig@illinois.edu
[3] San Diego Supercomputer Center, University of California, San Diego, CA 92117, USA; prodriguez@sdsc.edu
* Correspondence: s-tipei@illinois.edu

Abstract: Composers do not usually collaborate with other composers but, for the last half century, open works were created that invite performers to implement details left undetermined or even decide the order in which various sections of the composition are to be played. Chance operations were also used in the writing of musical pieces and, in music generated with the assistance of computers, controlled randomness found its place. This article proposes a platform designed to encourage collaborative and interactive composition on high-performance computers with DISSCO (Digital Instrument for Sound Synthesis and Composition). DISSCO incorporates random procedures as well as deterministic means of defining the components of a piece. It runs efficiently on the Comet supercomputer of the San Diego Supercomputing Center and uses the Jupyter notebook environment to integrate the end-to-end processes with a user. These tools, the implementation platform, and the collaboration management are discussed in detail. Comments regarding aesthetic implications of the partnership between one or more humans and computer—considered a bona fide collaborator—are also provided. Possible future developments are supplied at the end.

Keywords: collaborative composition; algorithmic composition; digital sound synthesis; high-performance computing

Citation: Tipei, S.; Craig, A.B.; Rodriguez, P.F. Using High-Performance Computers to Enable Collaborative and Interactive Composition with DISSCO. *Multimodal Technol. Interact.* **2021**, *5*, 24. https://doi.org/10.3390/mti5050024

Academic Editor: Insook Choi

Received: 2 February 2021
Accepted: 25 April 2021
Published: 5 May 2021

1. Introduction

DISSCO (Digital Instrument for Sound Synthesis and Composition) is computationally intensive software for algorithmic composition and additive sound synthesis. In this work we used high-performance computers to enable DISSCO to synthesize and play sounds in near real-time with many simultaneous users. This makes it possible for an arbitrary number of participants to cooperate interactively in creating a musical event—a collaborative composition. The product of this joint effort in DISSCO could be a finished and distinct work or a continuously evolving output.

Currently, in the classroom, for example, student composers using DISSCO can interact and collaborate on sound design and compositions, namely by just showing each other their work, then incorporating suggestions from their peers. That is a multistep sequential batch process managed by one person. Here, with modifications described in the paper, we have formalized some of that interaction and communication onto a high-performance computing platform. Our goal is to enable end-to-end composition, sound design, and performance cycle to occur in near real-time, with multiple, or even many, possibly remote, participants each interacting directly and simultaneously with all the steps from sound design to listening to the composition.

This article will present the platform and how we see the potential for composers to collaborate and interact with each other and the computer, and how the computer can have an expanded role, namely because it might help composers plan differently, greatly speed up the end-to-end process, and even semi-automate some of the experimentation

or performance phases, with guidance from the composers. Using a high-performance platform also opens the possibility of having many users involved, as either listeners, participants, or both.

We are not claiming that what gets composed is necessarily different, or that the nature of creativity is altered, but that the process of collaborating is facilitated so that composers interact with DISSCO, each other, and the computer, in a more effective and possibly enhanced manner.

1.1. Background

Composers do not usually collaborate with other composers. In rare occasions, an unfinished work is completed by somebody else as in the cases of Mozart's Requiem or Alban Berg's last scenes of his opera Lulu. There are also orchestrations of piano pieces and operatic transcriptions for piano created by people different from the original authors. Common to these examples is the fact that, in most cases, the original composer is no longer available to collaborate with the second one.

On the other hand, jazz musicians do collaborate and improvise on already existing tunes, in real time. In such cases, melody, chord changes or both are retained in the background and the overall formal structure is preserved or amplified. How strictly such rules are followed depends on each particular style, though, and the free jazz of the 1960s chose to ignore most of them. At roughly about the same time, Karlheinz Stockhausen and others wrote pieces that employed a set of instructions or graphics without offering any purely musical elements: performers were simply asked to react to each other's actions.

The collaborative and interactive composition project takes advantage of computer technology in order to facilitate the combined effort of two or more participants to build a valid musical construct, a composition. The software also allows for random operations to be introduced at all structural levels of a composition and choices made by the human users are realized only with probabilistic precision. They could produce unexpected and surprising results since the computer has an ability that humans do not possess: to model chance in an unbiased way. The collaboration is then extended and the "electronic brain", as the computer was sometimes called in early computer music, becomes part of the process not only by performing a vast number of operations very quickly, but also as a consequential contributor to the creative effort.

1.2. Indeterminacy: Chance and Randomness in Music Composition

Around 1951, influenced by Far-Eastern thinking, John Cage started to use chance procedures (the ancient I Ching book, a sky atlas, coin or dice throwing, blotches in poor quality manuscript paper, etc.) when composing his pieces. A few years later, Iannis Xenakis produced the first work in which durations, pitch intervals, density of textures, and the slope of glissandi were determined through stochastic distributions (normal, Poisson) or other probabilistic means. In later pieces, he also used Markov chains, random walks, and additional mathematically inspired tools [1]. The term "controlled randomness" came to describe situations in which the rule—distributions—and the ranges within which the events/sounds could exist are controlled by the composer while the actual details are indeterminate.

Concurrently, aleatory pieces made up of standalone modules which can be performed in any order such as Stockhausen's Klavierstücke XI [2] appeared. Umberto Eco, a semiotician and writer, in a 1962 an essay, published Opera Aperta (the Open Work, now a chapter of his book The Role of the Reader) [3], in which he talks about musical works that offer multiple ways in which they can be presented to audiences and be understood by them. A composition then is no longer one particular object but consists of a multitude of variants of an archetype, all equally valid.

Early computer-assisted composition (algorithmic composition realized with the assistance of a computer) made possible the families of compositions like the ST [4] pieces by Xenakis and algorithms [5] by Lejaren Hiller; each generated by the same code (the logic)

reading different versions of the data. Moreover, in the case of a program that includes random choices, changing the seed of the random generator produces a new version of the same piece as in the case of manifold compositions [6]. The following examples show two variants of the same short piece realized with exactly the same code and data but whose computations were triggered by different seed numbers. Figure 1 shows the amplitude (and, indirectly the texture density), on the vertical axis vs. time, on the horizontal axis. The duration of each example is 2 min and 26 s.

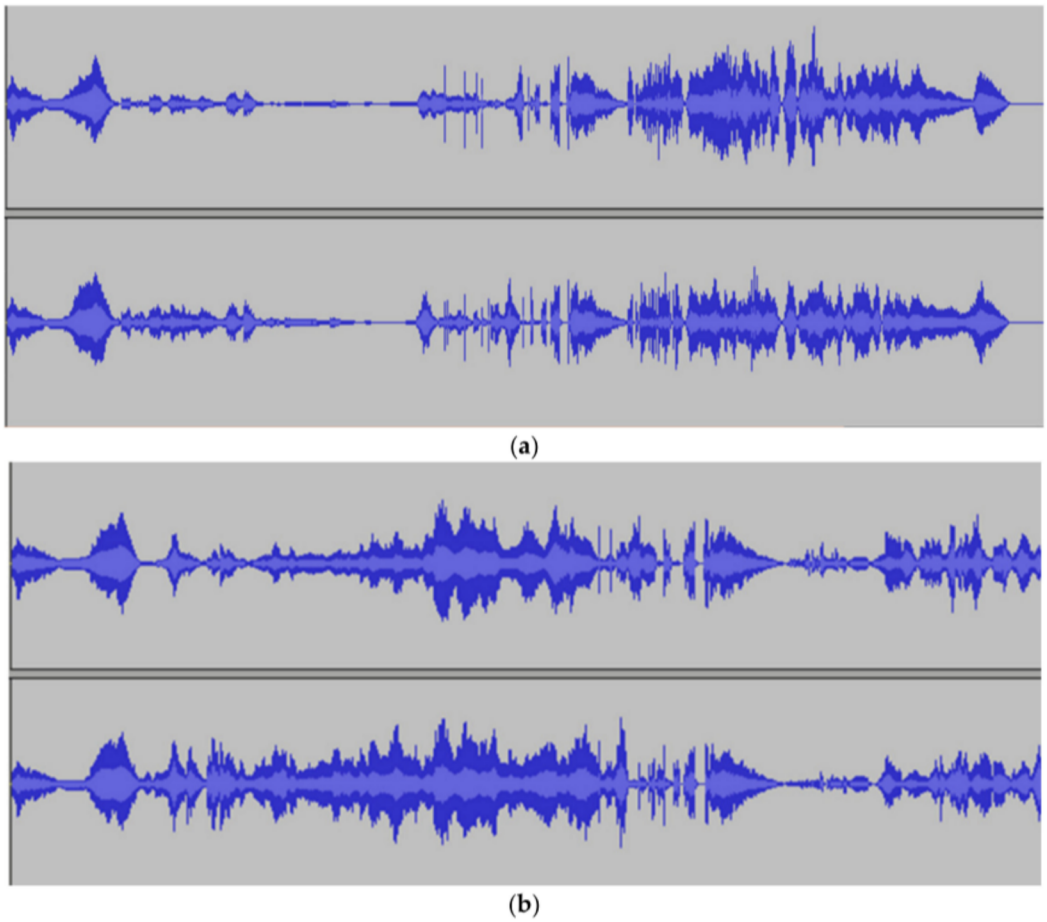

Figure 1. (**a**) Variant of a stereo manifold composition. Left and right channels are shown. (**b**) A different variant of the same manifold. The beginning, the ending, and a small area in the middle are the same while the rest is quite different.

More than 150 different variants of this manifold were generated and performed in public. Twelve other separate manifold compositions were generated and their variants performed during the last two decades.

1.3. Collaboration in Music

The dictionary defines to collaborate as: "to work jointly with others or together especially in an intellectual endeavor" [7] and as shown above, in a computer supported musical collaboration, the computer joins the other participants in a musical endeavor.

The team work can happen synchronously, in real time as in the case of jazz performers who improvise together on stage, or it can take place asynchronously as in the case of sound designers who work on sound files separately, at their convenience, and merge the results at a later time. Likewise, it can take place in the same geographic location or at a distance, with the participants stationed at faraway sites, using the Internet.

Musical collaborations can fall into a number of different categories, a non-exhaustive list of which could include:

- Composition;
- Performance;
- Instrument design or sound design;
- Arrangement or orchestration;
- Notation of scores and editing.

In the case of computer music, instrument design takes the form of sound design, and is particularly suitable for team work.

There are various tools that support collaboration in several of the categories listed above. Many of them support asynchronous collaboration, such as when one partner saves a document (such as a score or a sound file) and another one opens it later to contribute to it. The participants might not even know who the other person is. Other tools support synchronous collaboration, for example, JamKazam [8], which promises the ability to perform together, at a distance, and live, in real time.

As shown by the class use example, DISSCO allows multiple users to present entire works or only fragments, to exchange ideas and solutions and even collaborate. This became obvious when it was used in the classroom.

2. Methods: The DISSCO Software Tool

DISSCO (Digital Instrument for Sound Synthesis and Composition), the software on which the project is based, was developed over of number of years at the Computer Music Project (CMP) of the University of Illinois, Argonne National Laboratory (ANL), National Center for Supercomputer Applications (NCSA), and San Diego Supercomputer Center (SDSC). A first version, DIASS (Digital Instrument for Additive Sound Synthesis), was created at CMP and ANL in the mid 1990s and it was used both as a tool for sonification and in conjunction with other computer-assisted composition software; results were displayed in a virtual environment (CAVE) [9]. In 2004, a first embodiment of the composition component was added followed in 2013 by the attachment of a graphic user interface designed at CMP. More recently, many features were added due to the work of NCSA SPIN (Students Pushing Innovation) program [10] and to ongoing work performed at San Diego Supercomputer Center (see below). DISSCO is a reliable tool in continuous transformation that has been extensively tested as a medium for the creation of original music, performed at international festivals and conferences, and as an education tool, in classrooms at the University of Illinois.

The collaborative composition represents a major new step in enhancing the capacities of the software and in extending the area of potential applications.

2.1. DISSCO Structure

DISSCO has three main modules: LASS, a library for additive sound synthesis; CMOD, a composition module; and LASSIE, a graphic user interface through which the users enter data. DISSCO is written in C++ and is available on GitHub [11]. The backbone of the composition module is the implementation of an acyclic directed graph, a rooted tree, whose vertices or nodes represent "Events" at different structural levels (Figure 2). There is a top event—the entire piece—as well as high, medium, low and bottom events—sections and subsections—placed on vertices further and further away from the root. A parent-children relationship operates between structural levels and an analogy could be established with a set of matryoshka, the Russian dolls nested inside each other. If the top is the entire composition, the bottom events, the nodes at the end of the last edges or branches of the tree, are the places where actual sounds are created.

All events are assigned a start time, a duration and a type. Bottom events require extra information for sound definition which comprise frequency, loudness, vibrato (FM) and tremolo (AM), transients, filters as well as the position of sound sources in space (spatialization) and reverberation. Each sound is linked to a spectrum object in which a

spectral envelope is created when each partial receives a relative strength value and an envelope that trace its evolution in time.

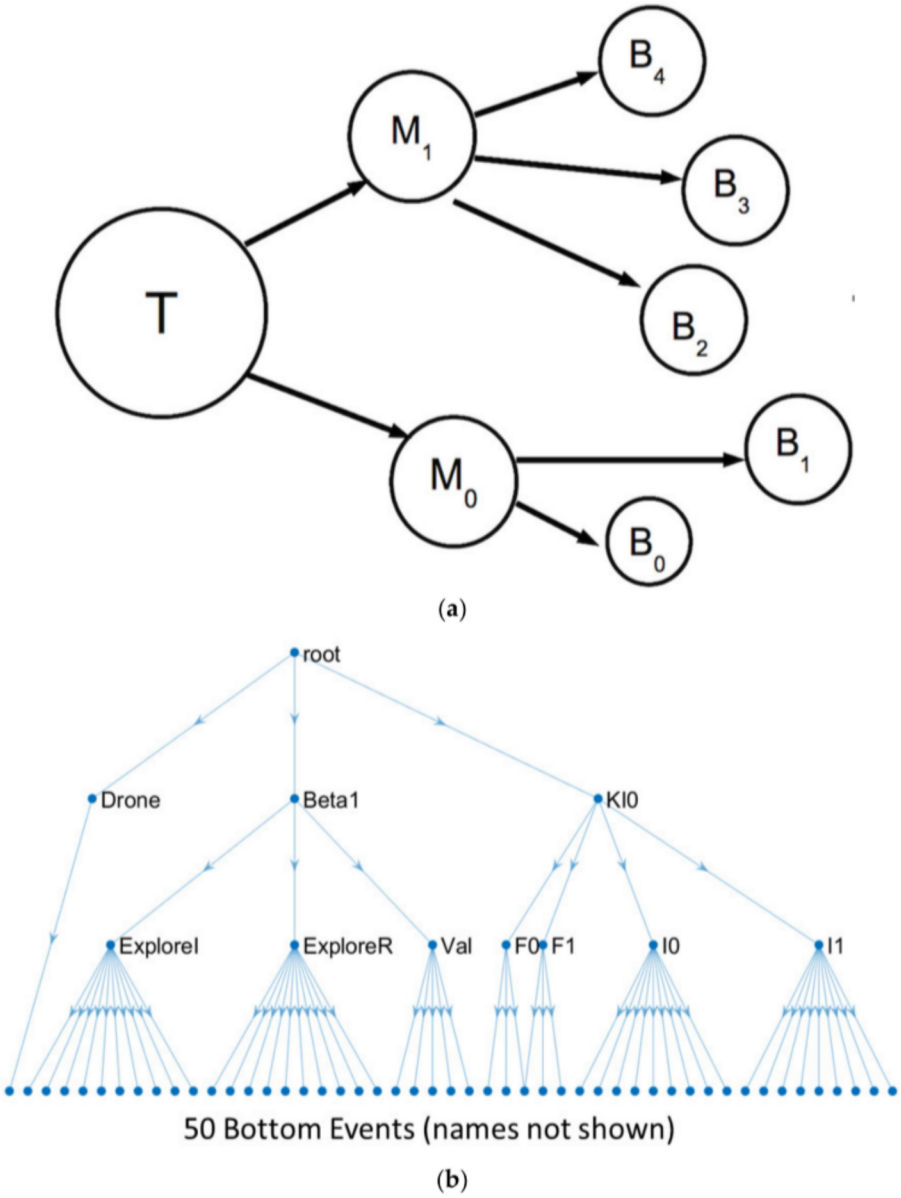

Figure 2. (**a**) A small example of the tree structure concept used in CMOD, with only one intermediate event layer between top and bottom events. (**b**) A moderately complex example of a tree structure from an actual script for a piece called "ghioc". Only the 61 unique edges out of a total of 216 are shown. Event names are shown for the top three layers only. Because of randomness and repetition, over 42,000 sound events are created by CMOD from this script.

DISSCO can produce both audio files and music notation. If sounds are sent from the bottom event to LASS to be synthesized, an audio file is created; if they are sent to the note class, a text file is generated with instructions for LilyPond [12], open-source software for editing and printing music.

2.2. LASS: Synthesis Features

The extra information necessary to define a sound is entered through LASSIE, processed in CMOD by the bottom class, and then implemented by LASS, the "synthesis engine". Users have many options when determining these additional features. For example,

- frequency can be assigned as an equal temperament pitch, as a partial of a given fundamental (just intonation), as a Hertz value or as a power of 2;
- for frequency and amplitude modulations (FM and AM), users control the probability of occurrence and the standard elements of FM synthesis: the magnitude of the distortion or the index factor (amplitude of the modulator) and the oscillation rate (modulator frequency);
- transients of frequency and amplitude add to this list the number of samples affected by the transient;
- spatialization options are stereo and multi-speaker diffusion through polar coordinates or through specific values assigned to individual speakers;
- there are many ways to control reverberation from room size, to a combination of delay, hi-low frequencies ratio, and proximity to the source, to accessing all values for the comb and allpass filters involved (DISSCO uses the model described by F. Richard Moore [13]); and
- there are seven types of general filters available:
    - Low pass
    - High pass
    - Band pass
    - Notch
    - Peaking band EQ
    - Low shelf
    - High shelf

  Almost all these parameters can be either static o dynamic through the use of envelopes.

2.3. CMOD: Utilities

The composition module is based on the proposition that same or similar procedures can be applied at different time scales (meaning: at all structural levels of the composition), from the entire piece to sound attributes. Such procedures are translated into utility functions that can be nested: a function may call another which may call another one in a cascade of successive calls resulting in more and more precise and sophisticated situations.

There are two broad categories of choices available: deterministic and random. The simplest in the first group selects values from a list according to some criteria, usually an order constraint. A family of random functions picks out values from within a defined range or adds a fuzzy area around the possible selection. It is joined by a random density function. More involved, "Stochos Range" distribution specifies dynamic minimum and maximum limits and a distribution within these limits, while "Stochos Functions" (inspired by an example from Xenakis' Formalized Music) stacks a number of envelopes whose combined values add up to 1 at all times, in order to determine the probability of each area comprised between them. While "Stochos" deals with floating-point numbers, "ValueCheck", a close relative, deals with integers and it is also used in "Sieves" (see below).

Higher level tools include Markov chains which build sequences of values (melodies, rhythms) and "Patterns" which apply or distort a given series of intervals grounded in an origin specified by the users.

Finally, Sieves are logical filters built with the help of modulo and Boolean operations, first introduced in music by Xenakis. Part of number theory, Sieves select a subset of values from a continuum, for example, a diatonic scale from the chromatic space. An abstract tool, Sieves can be applied to any sound parameter, not only to frequencies. The users specify modulo numbers, their offsets and the operations (union, intersection, etc.) along with a

dynamic range and a distribution within this range—in operations similar to ValueCheck described above.

2.4. CMOD: Building Children Events

There are three methods through which a parent event can spawn children events: "Continuum", "Sweep", and "Discrete". They all determine the start time, type, and duration of each child event and make sure it rests within the confines of the parent's time span.

Continuum assigns the start time of individual sounds either in orderly fashion, through a deterministic operation such as a specific list, a sieve, et cetera, or by scattering them with the help of random functions.

Sweep ensures that no sound begins before the preceding one has ended thus ensuring the creation of a continuous sequence (melody-like). These two methods accept both integer and floating-point values for start times or durations.

Discrete builds a three-dimensional matrix (start time, duration, type) and ensures that children with incompatible types do not collide. For example, two instrumental sound production techniques such as bowing a string and plucking it—two sound types—cannot exist at the same time. This method admits only integer values.

There are also three time units that can be used: seconds, "Elementary Displacement Units" (EDUs) and percentages of the parent's duration. EDUs form a grid of equally distanced points from which a utility such as a sieve will select the relevant ones. In the frequency domain, one EDU could be the equivalent of a semitone and in the time domain, one EDU could equal 1/LCM, the least common multiple of all subdivisions of the beat that are employed.

2.5. LASSIE: User Input for Sound Design and Composition

The graphic user interface LASSIE is written in C++ and implemented using gtkmm, the connection between C++ and the GTK library. It creates an XML file (a DISSCO script) that is read by CMOD and reflects all the features available in DISSCO that the users can control.

In a collaborative execution, participants are creating their own version of the project script file, always updating the latest script file from among all users. Collaborators have to get a lock before editing the file. The file lock is released when a user saves his/her file and it is up for grabs by somebody else. A Jupyter notebook, described next, is continually looping through the steps of running CMOD and playing audio files, and will look for the latest script file (from any user) before starting CMOD again. Thus, the amount of coordination can be up to the users, or the notebook can be modified to impose an order.

2.6. Jupyter Notebook to Control DISSCO and LASSIE

A Jupyter notebook [14] is an editing environment for Python (or other languages) that allows segments of code and mark-up text to coexist in one file. The environment provides extra functionality to run code cells, which can just be chunks of a program. The Jupyter environment can be executed within a single machine, like a personal laptop. It can also be executed over an http connection in which the code is run on a host computer and the output is displayed through a browser, even without a dedicated web server. In this sense, Jupyter provides a lightweight, graphical interface for shared distributed computing environments, like high performance computers, in which there is no fixed URL address.

Python with Jupyter has several modules that specifically provide audio and graphic widgets for user interaction, which includes the ability to modify or enhance code. Given that the present DISSCO users are tech savvy with some level of coding knowledge, Python with Jupyter was selected for this implementation of the collaborative environment described below. In the future, the project could be expanded to a broader class of users, dedicated servers could be acquired, thus using a more restricted, but easier to access, client-server environment.

3. Methods: Collaborative DISSCO Platform and Environment

Originally, DISSCO modules were executed separately and in sequence from LASSIE to CMOD/LASS, with the user managing files between modules and for audio playback. In order to create a collaborative and interactive environment, a Jupyter notebook is used to tie together all the pieces (Figure 3). The notebook helps control the flow of data between modules and presents the audio files for play, thereby enabling more dynamic, interactive, and continuous execution, as described next.

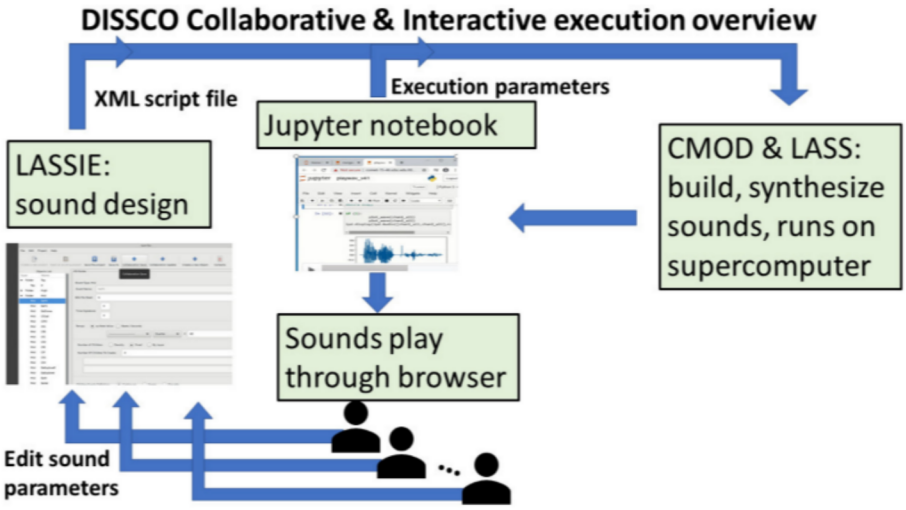

Figure 3. Platform overview also shows the data and workflow. Sounds are synthesized as "slices" of the entire score and played in near-real time for one to many users on their personal computers. All users can also modify the sounds throughout the process.

3.1. The Human-Computer Interaction

The human-computer information flow can be regarded as an iteration of an evolving loop:

design-> build-> synthesize-> play-> design-> build-> synthesize-> play . . . et cetera.

Functionally, a DISSCO user creates events as different nodes on a tree (see Figure 2), where all nodes have start times and durations that may or may not be completely random, but also depend on the boundaries set up by the parent node, which may also span the entire piece. The sound events may or may not have other randomness in their sound wave parameters as well. In this sense, a composer is potentially working with the entire piece as one potential unit, or even the entire set of all possible pieces, given the parameters. However, the user does not have to think about linear sequences, perhaps only about patterns and relationships.

The computer running DISSCO must translate events into sound sequences. In fact, facilitating this translation for multiple users is what makes this platform implementation very different than organizing an online group of musicians with physical instruments. The CMOD program reads through the events tree, applies random functions, and realizes a set of start times and durations for each sound, and then synthesizes sounds with LASS. For an interactive and streaming playback, the program will produce the score in temporal sequence, as opposed to sequence of tree events. Because the synthesis is computationally expensive, the operations are parallelized to organize sounds into sequence as soon as possible. To make it interactive and efficient, it can now do all of that in almost real-time, or as close as possible to when it should be played, and then output sounds in small slices of the piece to stream over the Internet.

With streaming, the composer has more flexibility in how they interact with DISSCO. They can work with the piece as a global unit, hear the piece as one complete score, and develop or edit the piece as such. Or, they can work with only 1 or 2 score slices, that cover

a few to many seconds at any point in the piece, and iteratively develop the lower (in the tree) sound events that occur in those slices—without necessarily knowing which of the higher nodes are involved, or if the sound events are also repeated at other times.

Moreover, because we have exposed DISSCO to take some input in real-time, one could also expand the computer's role by programming the computer to handle meta-processes. For example, one could have the computer build-synthesize-play the entire tree of events, or just certain branches. In this sense, a composer who is a programmer might also consider those possibilities for designing events in DISSCO.

3.2. The HPC Platform

There is another aspect to the human-computer interaction, which is the mechanics and logistics of execution, such as acquiring high-performance computing (HPC) resources and setting up execution parameters. Here we describe the advantages and options for running on an HPC platform.

As described above, the events can be short (milliseconds) or long (10s of seconds or even minutes), and they are organized as a tree of hierarchically ordered types, where the last leaf nodes are the final and fully specified sound events. CMOD translates the tree into overlapping sequences of events in time. The ultimate phase of the process takes place in LASS, the additive synthesis module, and consists of the physical instantiation of single sine waves. These partials are then mixed into complex sound waves, which are represented as arrays of float values, and mixed together into audio tracks. The tracks are further mixed into audio files (wav or aiff format) which can be rendered through an audio device on the users' computer.

The remote, multi-user, collaborative setting adds a layer of complexity which is mediated by the computer and is both enabling and restrictive. It is enabling since it nurtures an exchange of ideas between collaborators and a way to test them expeditiously or almost immediately. At the same time, it can be restrictive because remote users have to be coordinated. Unlike editing a shared Google Doc, the edits are not the end result itself, and cannot be instantaneously fed back. For example, to hear the result of changing a randomness parameter might require replaying the entire piece or a substantial portion of it.

The Comet supercomputer, which is part of the NSF funded XSEDE network [15,16] is used. Comet is a shared distributed computing environment with over 1900 compute nodes available, each with 24 CPU cores and 120 GB RAM. We typically run DISSCO with 1 to 8 nodes.

There are two main computational burdens that motivate the use of high-performance computing resources. One is that DISSCO pieces, especially those with granular sounds and high sampling rate (e.g., 44,100 Hz) are computationally expensive and benefit greatly from multi-node and multi-core parallelization. In the example DISSCO piece described above (Figure 2b), which has about 60 bottom events, and more than 42,000 sound events created, there are approximately 10.9 million random functions calls. Using 44,100 Hz, for an approximately 13 min piece, there are about 35 million float values per track in the computer sound file. Adding up all the durations for all 42,000 sounds results in about 10,300 s of sound calculations, which imply that about 450 million float operations were executed. This would take many hours on a standard laptop computer—negating the possibility for interactive execution.

We benchmarked DISSCO with several pieces and found that, some of the larger pieces that took hours on one node can be reduced to minutes with many nodes in parallel. Table 1, 2nd column, shows performance time for a medium size score with about 216 events, about 42,000 sounds, that is about 13 min long, depending on randomness.

The second computational burden for running DISSCO in near real-time is to translate the non-sequential tree into small slices of the sequential output for streaming. As the leaf nodes of a tree are processed, sound events for that leaf are realized and made immediately available to be synthesized. However, those sounds' start times are not in sequence and need to be sorted. A global sort is impossible at this point because all sound events have

not even been created, and it would be time consuming to wait. Instead, a local sort is performed within each CPU core with a smaller set of sounds. The strategy here is that every core holds onto sounds in small sets that are sorted in place by start time, so that they can be locally processed in that core at an appropriate window of time before they are played. In this sense, the local sequence is an approximation of the global sequence that works in practice, as long as you have enough computer nodes and CPU cores.

Table 1. Performance in batch vs. real-time for a medium size piece on Comet supercomputer. The wall clock time is the user experienced time (as measure by a clock on the wall) for a job to start and finish.

Number of Computer Nodes (Each with 24 CPU Cores)	Wall Clock Time for All 42 K Sounds (Where No Sorting Required)	Number of Sounds Built and Synthesized in Real-Time (~13 min Score; Sorting by Start Times Is Required as Sounds Are Built; Sounds Are Only Synthesized in a Time Window Just before Their Start Times)
1	30:30 (min:s)	1425 (out of 42,000)
2	18:15	3184
4	9:45	13,178
8	5:30	27,402

In general, the more nodes we use the more sounds will be generated and synthesized before they must be played. Otherwise, if sounds are queued for synthesis after a time-window has passed it will be skipped. In other words, if a user asks to stream 10 s chunks of the score, then the program has about a 9.5 s time-window for rendering, mixing, and loading into the browser in order to produce the next 10 s worth of sounds. This will allow for some transmission delay and result in near real-time build-synthesize-streaming. However, if a time-window size is too small, say less than 2 s, then the overhead of taking out slices, mixing, and streaming can limit the ability to render all the sounds that were expected. Using Comet, it was found that having a reasonable window size to render sounds, and having enough nodes, will enable pieces to be mostly processed and streamed in nearly real-time, see Table 1, 3rd column. It is worth noting that sound design is more likely the slower process, as the user must navigate the LASSIE interface and has many options for sound parameters.

4. Collaborative DISSCO Implementation Results

As detailed above, DISSCO offers a large array of options; it was designed as a "Rolls Royce bulldozer", a flexible instrument that offers sophisticated control along with brute force: unrestricted number of event nodes (structural sections), of sounds or of sound components (partials). Thus, for a collaborative implementation, there are several practical considerations for setting up, managing, executing, and organizing a collaborative composition in DISSCO, which we describe here. There is also a demonstration video available in Supplementary Materials (see below).

4.1. User Requirements

Perhaps the main, overarching need is that at least one user needs to be proficient in running DISSCO, Unix, Comet, and the information that the notebook uses to control DISSCO. This user would be a designated host that coordinates execution. In the current implementation, other users need to have some basic knowledge of DISSCO, access to Comet, and appropriate permissions. However, in the future, it could be that only the designated host needs access.

- Designated host: The designated host can acquire one to many Comet compute nodes for a reasonable amount of time, to run the Jupyter notebook, CMOD, and LASS. The host also acquires one node to execute LASSIE (see Figure 3). The host should specify

all run-time parameters for Comet execution and for the notebook execution. The host, or other users, can also run optional pre- and post-cleanup or saving of files.

- Other users: All users need to access one compute node of Comet to run their own session of both the Jupyter notebook and LASSIE if they plan to edit scripts as they hear them.
- Planning the collaboration: DISSCO was initially conceived around the notion of pre-compositional work, making decisions ahead of time about structure and materials. Such groundwork facilitates the execution of the musical piece while being amenable to change if necessary. Thus, each user should come to a collaboration with some knowledge or plans or prior notions of what they can do with DISSCO. However, in this implementation, there is also the possibility to experiment and thrash around without strong or definite plans. In this sense the composition task is potentially very different.

4.2. The Script

An initial XML file ("___.dissco") is created via LASSIE, preferably through consultations between collaborators. It could either be fairly complex and detailed, a tree with many branches and leaves, or it could be simple and sparse, comprised of a top event and of only one bottom event. Such a basic script could serve as a platform for experimentation, for instance, in the sound design arena, since concerns about the formal architecture will be absent. In this case, the output is obtained swiftly and the conversation between users proceeds at a fast pace. By contrast, an intricate initial script requires involved pre-compositional deliberations and takes longer to execute and to appraise the result.

A library of envelopes that dynamically control the data must be built from the beginning. They define the behavior of sound and partials in time and are necessary to any attempt to design sounds. Likewise, pre-existing scripts could also be helpful when constructing the architecture of a piece. They are best suited for events of higher level (structural nodes) than for bottom events.

There is only one "___.dissco" file, for example, a shared script. Using LASSIE, collaborators are constantly updating, editing the latest version of the file which contains the contributions of all the other users. Participants have to get a lock before editing the file. When a user saves the file, the lock is released and it is up for grabs by somebody else. Additionally, users can update an "Events-to-Use' list with each CMOD iteration so that only a limited set of events will be built-synthesized-played. Thus, users can appraise small parts of intricate pieces as they are also updating script files.

4.3. Coordinating Users

The Jupyter notebook (see Figure 3) is continually looping through the steps of running CMOD and creating audio files, and will look for the latest script file (from any user). Thus, the amount of coordination can be up to the host, the users, or the notebook can be modified to impose an order.

The length of time before Jupyter starts CMOD again using a revised script may be modified. It could be short in the case of a synchronous collaboration, simulating an almost real time improvisation or it could be of an arbitrary duration in the case of an asynchronous process similar to the time (possibly many hours) needed by a composer to consider new changes or to the extent of time taken by a chess player before making the next move.

A distinct situation arises when the computer is regarded as one of the collaborators. The indeterminacy embedded in the system through randomness will trigger changes in the output independent of the latest modifications of the script operated by humans. If the process is allowed to run continuously, the computer will generate variant after variant of the output, all different to some degree and the humans can stop the operation and assess the outcome at a convenient moment. Because of this, the so-called "electronic brain" is actually a bona fide collaborator, and part of the experiment.

4.4. Composition Results

Two possible outcomes are apparent: a piece, in the traditional sense of a musical object of definite duration which can be performed time and again in the same form and an installation of an arbitrary duration that changes to some extent constantly. The first is more likely the result of an asynchronous collaboration due to the complexity of the process although being the result of a synchronous operation cannot be excluded. The second outcome, an installation, is the result of a synchronous scenario in which both humans and a computer running unremittingly participate.

In both instances the output evolves over a period of time, in the case of a piece during its construction and in the case of an installation both during its assemblage and during its performance. The real life-like aspect could be enhanced if the process started with a directed graph tree with few leaf nodes and branches to which new ones are added in time until a much more complex structure is achieved and, after a while, the tree is "pruned", branches and nodes removed, and the entity is left to die.

At the present time, only humans can direct such transformations, but the computer can provide the everchanging content of the leaves.

4.5. Potential Collaboration Use Cases

This is a research project, an experiment seeking to encourage experimentation. Its primary focus is to create musical events through teamwork. Participants in this effort have to have an interest—and some knowledge—in at least one of the fields of music composition, sound design, or informatics and computation. As stated above, at the present time, the collaboration requires familiarity with the software and ways of running it. By developing friendlier ways of accessing it in the future, such as building collections of pre-existing scripts, larger categories of users can be included.

The original motivating use case came out of an advanced music theory course at UIUC. Students learn to create a short DISSCO piece that demonstrates certain synthesis and compositional techniques. Each student does this on their own and then there is a listening and feedback session held in the classroom where each student plays their piece for the other students. Traditionally, students would provide feedback to the creator, who could then choose to alter the piece, or not, after the session. With collaborative disco, however, the listening session may be a completely interactive session in which, rather than giving verbal comments to the student whose piece is being discussed, the class of students could, rather, inject their ideas for changes in real time, and ask "what about this?" and hearing the results in real time as the piece is performed. Depending on the setting and goals, the exchanges could be coordinated and negotiated, or it could be more free-form and dynamic in which one student creates a "seed" script and all students alter, expand, and grow the script in real-time with minimal constraints.

In addition to advanced music students, the system could be made attractive to non-professionals who wish to experiment. Already, students majoring in fields as diverse as architecture, finance, visual arts or psychology who worked with DISSCO were able to design original sounds and short compositions in less than one semester although they had no previous musical training. It could also be used as an education tool with the teacher in the role of the host asking for input from students about ways to assemble already existing building blocks or how to design sounds. Like the UPIC (Unité Polyagogique Informatique CEMAMu) tool devised by Xenakis [17], it has the potential to stimulate creativity in people without musical training, especially children.

Musical collaboration could also be construed as a game, with some kind of competition to get the lock on the script. Instead of the host arbitrating the process, chance operations or the use of game theory matrices could decide the next contribution. The participant who supplies a more appropriate solution to a particular situation could receive the right to edit the script next.

5. Discussion

The collaborative composition on HPC projects allows the participants to take turns in adding, modifying, or rejecting previous contributions, a situation that resembles both the completion of a work by another person and the dialogue established during a collective improvisation. It is also similar to the exquisite corps [18] method practiced by surrealist artists and poets by which participants add elements in sequence to a painting or poem according to some rule.

Compositions generated with either the assistance of a computer, containing digitally synthesized sounds, or in combination, are usually the result of some degree of collaboration, in any case. Unlike the pioneering times when composers of computer music had to have programming skills in order to produce anything, the present-day complexity of the task makes it difficult for one person alone to be, at the same time, an accomplished musician, efficient at writing code and knowledgeable about acoustics. The author is somewhat in the position of a movie director: in charge of an entire team. This project divides that top responsibility between participants and replaces a solitary operation with an exercise in cooperation.

As shown before, the computer also contributes to this activity. Its role has been discussed before and may be presumed as a composer's collaborator [19]. Humans are better than machines at inventing and organizing (musical) structures while computers are better at making random choices, unbiased by human preconceptions. DISSCO has a hierarchical structure and makes deterministic operations available, while, at the same time, many of its features are probabilistic. It was designed as a "black box": the user inputs the data in the form of the dissco script and retrieves the output but does not interfere in the computations and does not modify the results. This is still true once a modified script has been submitted. Between edits, the computer is completely in charge, making choices according to an internally generated sequence of random numbers—a good reason to include it in the partnership.

Then, two questions present themselves: can computers be creative? and who is the author/composer of music originating from a collaboration between more than one person and a computer?

The answer to the first question depends on what definition of creativity one endorses. In a short essay, "Musique et Originalité" [20], Iannis Xenakis distinguishes between three types of originality (which we interpret as meaning creativity). In a succinct paraphrased formulation, they are:

- combining, rearranging existing information into something new
- observing critically, comparing with stored information and discerning rules and patterns
- inventing something from nothing, something that is unlike anything observed before

Although not using the terminology favored by present authorities in this matter, Xenakis comes very close to Margaret Boden's classification: combinatorial ("unfamiliar combinations of familiar ideas"), exploratory, and transformational ("transformation of conceptual spaces in people's minds") [21].

Computers are quite adept at satisfying the first condition and partially good at fulfilling the second one: comparing existing things and, sometimes, even discovering rules. For now, though, the third type of creativity belongs to humans only. What Xenakis means by "unlike anything observed before" is the equivalent of establishing different laws of physics that would create a different universe and, at the present time, as far as we know, computers can not invent such new rules of the game.

The answer to the second question is in the name of the project: collaboration, although presumably, no computer, not even a high-performing one, will receive royalties.

A more serious discussion should focus on the nature of the output. DISSCO was created with the goal in mind of generating manifold compositions, multiple variants of a piece. Like faces in a crowd, they share basic features while displaying unique characteristics. As such, a manifold is not a composition but a composition class, a family

of similar related works, and the composer does not engender a single, well defined object, but conceives an entire species with all actual and potential members.

It is also worth considering that there are some important elements of any collaboration, such as: the objects or products, the medium of exchange, the nature of the teamwork, and timing of interactions. Computers can be used to enhance collaborations in many contexts by servicing these elements. For example, in collaborative design or collaborative work there is typically a shared explicit goal and a primary concern is to achieve some form of success through communication and teamwork. A good computing platform supports flexibility in the way participants communicate and share content [22]. On the other hand, in collaborative art, a final goal is perhaps less important than supporting cognitive processes and artistic styles [23].

For the case of composition, there are of course many ways to collaborate—face-to-face, over virtual meetings on the internet, perhaps with special tools to synchronize remote musicians. In the present case, the computer is a cocreator, producing/synthesizing and generating sound details, as well as providing an interactive environment. The goal of this current implementation is to allow for a variety of exchanges and enable fast processing, no matter the nature of the collaboration.

In summary, the collaborative composition project is a substantial extension of DISSCO. It supports the participation of more than one author in the creative process and removes the limitation of having a unique, exclusive designer of both pieces, architecture and its construction materials—the sounds themselves. The team activity changes the very nature of composing from a rather romantic and lonely operation performed by a prodigious genius into a collective enterprise of active synergy. Finally, the present version of DISSCO works in parallel on a high-performance computer and empowers the collaborators to experiment faster and, at the same time, work with an increased volume of data in close to real time.

6. Future Work

Our next step is to bring the system into the classroom with advanced students to observe it in actual use and gain feedback as they test out collaborating and interacting with each other on small pieces. We have already done this ourselves in a testing and demonstration environment. In the classroom, with new and unbiased users, we anticipate finding new details or modes of interactions that will drive our short-term development and generate ideas for further refinements.

In the longer term, one direction that is we can pursue is that of turning the collaborative DISSCO system into a science gateway [24] and enable use cases beyond the classroom. As the system sits now, it requires someone involved to have an allocation on the high-performance computing system and to serve as a host for the session. A science gateway can shield the participants from needing an allocation on the HPC system. Rather, the principal investigator can apply for such an allocation and then anyone who the principal investigator allows to use the gateway can participate. The gateway can run persistently, thus allowing the users to spin up their own collaborative sessions without the intervention of a specific host person. Additionally, the gateway can be built to be fault tolerant in order to reduce the possibility of participants in a session from breaking the system. It can also provide sample sessions and simple starting points to allow new users to begin right away and learn the system, while providing more savvy users more flexibility.

Another area considered for possible future work is to allow other types of devices and interfaces that participants can use with the system. For example, a session might consist of a heterogeneous mixture of Jupyter notebooks, real time physical interaction employing sensors in a smartphone, some participants with midi keyboards or other midi devices, and some with traditional acoustic instruments or microphones to control the system with audio signals.

Real time visualization of the resulting piece as it is being composed/performed could become a compelling display especially when used in (semi) permanent installations in concert venues, museums, schools, and other facilities. The visualization could be provided

on large screens or could be displayed in a virtual reality system that also furnishes input opportunities to the system. The visualization could also be made more utilitarian in a sense by creating a graph of the state of the system and allowing the participant to click on different parts of the tree and make modifications at that point.

One of the current limitations of the system is the requirement that a knowledgeable person serve as a host and that every time a session is invoked, it involves a technical setup. A further development would create a science gateway which would hide some of that complexity and make it more straightforward to roll out to new users and use cases.

Likewise, currently the collaborators are directly tweaking the DISSCO execution via Jupyter. A future task would be to hide the complexity by integrating other interfaces that are more familiar to a broader range of users.

Supplementary Materials: The following are available online at https://www.mdpi.com/article/10.3390/mti5050024/s1, Video S1: Supplementary Video to demonstrate and introduce.

Author Contributions: Conceptualization, S.T.; Methodology, writing, S.T., A.B.C., P.F.R.; Software, P.F.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Comet at the San Diego Supercomputer Center through allocation TG-ART150003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This project was made possible through the XSEDE Extended Collaborative Support Service (ECSS) program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xenakis, I. *Formalized Music*; Pendragon Press: Stuyvesant, NY, USA, 1992.
2. Stockhausen, K. *KlavierstückeXI Nr. 7*; Universal Edition: Vienna, Austria, 1956.
3. Eco, U. *The Role of the Reader*; Indiana University Press: Bloomington, IN, USA, 1979.
4. Xenakis, I. *ST Pieces*; Boosey and Hawkes: London, UK; New York, NY, USA, 1967.
5. Hiller, L. Music Composed with Computers—A Historical Survey. In *The Computer and Music*; Lincoln, H., Ed.; Cornell University Press: Ithaca, Greece; London, UK, 1970.
6. Tipei, S. Conceiving Music Today: Manifold Compositions, DISSCO, and Beyond, Recent Advances in Acoustics & Music. In *Proceedings of the 11th WSEAS International Conference on Acoustics & Music: Theory & Applications (AMTA '10); WSEAS Mechanical Engineering Series*; World Scientific and Engineering Academy and Society (WSEAS): Stevens Point, WI, USA, 2010.
7. Merriam-Webster. Collaborate. Available online: https://www.merriam-webster.com/dictionary/collaborate (accessed on 26 January 2021).
8. JamKazam. Available online: https://jamkazam.com (accessed on 26 January 2021).
9. Cruz-Neira, C.; Sandin, D.; DeFanti, T.; Kenyon, R.; Hart, J.C. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Commun. ACM* **1992**, *35*, 64–72. [CrossRef]
10. NCSA SPIN. Students Pushing Innovation. Available online: http://spin.ncsa.illinois.edu/ (accessed on 26 January 2021).
11. Gousios, G.; Vasilescu, B.; Serebrenik, A.; Zaidman, A. *Lean GH Torrent: GitHub Data on Demand, International Working Conference on Mining Software Repositories (MSR)*; Association for Computing Machinery: New York, NY, USA,, 2014.
12. LilyPond. Available online: http://lilypond.org/ (accessed on 13 March 2021).
13. Moore, F. *Elements of Computer Music*; Prentice Hall: Englewood Cliffs, NJ, USA, 1990.
14. Juypiter. Available online: https://en.wikipedia.org/wiki/Project_Jupyter9 (accessed on 16 January 2021).
15. Towns, J.; Cockerill, T.; Dahan, M.; Foster, I.; Gaither, K.; Grimshaw, A.; Hazlewood, V.; Lathrop, S.; Lifka, D.; Peterson, G.; et al. XSEDE: Accelerating Scientific Discovery. *Comput. Sci. Eng.* **2014**, *16*, 62–74. [CrossRef]
16. Wilkins-Diehr, N.; Sanielevici, S.; Alameda, J.; Cazes, J.; Crosby, L.; Pierce, M.; Roskies, R. *Proceedings of the High Performance Computer Applications 6th International Conference, ISUM 2015, Mexico City, Mexico, 9–13 March 2015*; Revised Selected Papers; Gitler, I., Klapp, J., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 3–13; ISBN 978-3-319-32243-8. [CrossRef]

17. Raczinski, J.; Marino, G.; Serra, M. New UPIC System Demonstration. In Proceedings of the International Computer Music Conference, Glasgow, UK, 10–15 September 1990.
18. Breton, A.; Breton Remembers. *Exhibition catalogue*, *Le Cadavre Exquis: Son Exaltation*; La Dragonne: Galerie Nina Dausset, Paris, France, 1948.
19. Tipei, S. The Computer a Composer's Collaborator. *LEONARDO J. Int. Soc. Arts Sci. Technol.* **1989**, *22*, 189–195. [CrossRef]
20. Xenakis, I. *Musique et Originalité*; Nouvelles Editions Seguier: Paris, France, 1996.
21. Boden, M. *The Creative Mind: Myths and Mechanisms*, 2nd ed.; Routledge: Milton Park, Milton, Abingdon, UK, 2004.
22. Davis, N. Human-Computer Co-Creativity: Blending Human and Computational Creativity. In Proceedings of the Ninth Artificial Intelligence and Interactive Digital Entertainment Conference, Boston, MA, USA, 14–18 October 2013.
23. Idi, D.B.; Khaidzir, K.A.M. Critical perspective of design collaboration: A review. *Front. Archit. Res.* **2018**, *7*, 544–560. [CrossRef]
24. Wilkins-Diehr, N.; Zentner, M.; Pierce, M.; Dahan, M.; Lawrence, K.; Hayden, L.; Mullinix, N. The Science Gateways Community Institute at Two Years. In Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18), Pittsburgh, PA, USA, 22–26 July 2018; ACM: New York, NY, USA, 2018. [CrossRef]