I am a computer science researcher with a background in human-computer interaction (HCI) and computer music. Throughout my graduate studies, I have explored how computational systems can facilitate collaboration, communication, and participation in the context of creating artifacts while mitigating the challenges inherent to users' real-time involvement in these processes. I have created and deployed a number of interactive systems that support creative activities in applications such as programming, writing, music performance, and UI design. The novel computational tools and methods developed in my research make the creative process and artifacts in progress more visible to users, helping them participate in the creation of artifacts they use.

The interactive systems that I build involve *live collaboration***,** in which users of an artifact are involved in creating it. I use the term *live* when the process of creating an artifact is visible to users in real time. For example, at a live music concert, music is the *artifact*, musicians are the *creators*, and audience members are the *users*. However, this is not always the case. The creation of an artifact and its consumption often take place asynchronously. For example, readers (users) read a book (an artifact) only after the author (its creator) has finished writing it. The same is typically true of food cooked at a restaurant or a painting exhibited at a museum. In the systems that I have explored, the level of immediacy and the visibility of the creative process to users constitutes *liveness*. Furthermore, these systems actively involve users in the creation process; creators can collaborate with users in real time, in a manner that resembles cooking classes or workshops where people can create their own artifacts.

Live collaboration is beneficial for users and creators alike: making the process of creation visible to users encourages them to engage with the final artifact; additionally, creators can receive responses from a continuous closed feedback loop with users. However, live collaboration in which users participate in creating an artifact occurs in only limited settings (e.g., workshops, classrooms) because of the costs involved in creating suitable environments. In my research, I have created computational systems that support live collaboration with users, focusing on three topics (Fig. 1): (1) the challenges inherent to collaborative creation in live contexts and computational tools that can address them; (2) methods that can reduce the barriers of entry to live collaboration for non-experts or broaden the applications of live collaboration; and (3) means for preserving the liveness of the creative process when an artifact is to be consumed asynchronously, affording users more expressivity and access to information only available in real-time history. These topics will be discussed in the following sections.

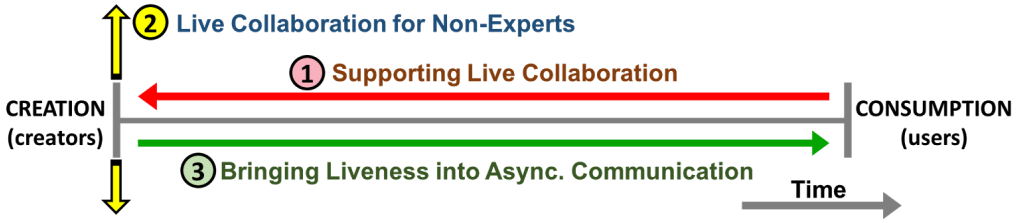

Figure 1. An artifact is typically consumed much later than the time that it was created. My research explores live collaboration in which users can be involved in the creative process. There are three subgoals that I present: ① to identify and address the challenges in computationally supporting live collaboration, ② to develop tools and methods that can reduce the barriers of entry to live collaboration for non-experts and broaden the applications of live collaboration, and ③ to bring liveness into asynchronous communication when we cannot collaborate live.

SUPPORTING LIVE COLLABORATION BETWEEN USERS AND CREATORS

Over the course of my graduate studies, I have worked on a thread of research that seeks to create novel modes of live collaboration for creative and artistic domains.

Programming Environments for Collaborative, Live Programming

Artists' experimental takes on musical aesthetics pose intriguing computational and engineering challenges in the systems that support them for their artistic expression. I have created computational techniques that mediate collaboration between musicians and help musicians explore novel computer music, such as programming language that generates real-time music notation [3] and live-coded digital musical instruments [4]. In the experimental

music performance "Live-Coding the Mobile Musical Instrument", two programmers (creators) collaborate to write a mobile music instrument (an artifact) on a tablet *live on stage*, while a musician (a user) performs on the musical instrument that is being developed over a wireless network [4]. In this extreme setting, a pair of programmers develops a single interactive program—the mobile musical instrument—in real time while a musician plays it. The programmers needed to have a clear understanding of the live program's state as it was being changed by their collaborators' code, as well as how the musician interacted with the program; these challenges were not adequately addressed by pre-existing tools. From the user's perspective, the instrument's usability abruptly changes as new code is executed. To address this challenge, I developed a programming environment that shows the program state in real time, including how the user interacts with the program, within the code editor [5] (Fig. 2, video:https://youtu.be/Ct_0-hu6Q6Q). In addition, the programmers can freely improvise without worrying about naming conflicts in declaring variables and functions, as each programmer has their own namespace. They can also share objects (e.g. variables and functions) selectively, should the programmers wish to work with some objects collaboratively. To minimize interference caused by changes in the program's state during the performance, the programmers can let the user control the timing of updates to the program. The exploration of this experimental case encompasses and addresses the challenges of potential applications of collaborative programming in real time: the program's state is visible to the programmers without the potential risks of breaking each other's code. I suggested a similar programming environment for crowdsourced software engineering, wherein expert programmers can self-coordinate and be aware of their collaborators' work live [6].

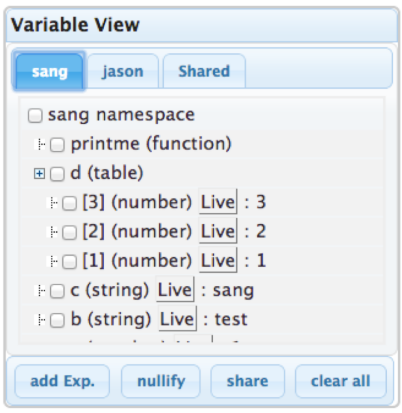

Figure 2. Individual programmers live program state are displayed, and they can share selected objects with collaborators.

Qualitative User Study on Real-Time Collaboration Between a Requester and Crowdworkers

An established approach to crowdsourcing involves breaking one large task into smaller *microtasks* that non-expert crowdworkers can solve independently, without the need to communicate directly with the requester. However, microtasking fails for creative tasks as there are no well-defined steps into which the task can be decomposed; additionally, subtasks are often interdependent. Real-time collaboration between requesters and crowdworkers can address these challenges. This type of continuous interaction is increasingly frequent and is broadening the applications of crowdsourcing; currently, though, we lack a deep understanding of interactivity between requesters and workers. To better understand how requesters speak to collaborate with crowdworkers in the setting, I conducted a qualitative user study with a crowd-powered tool for user interface prototyping where a requester (a user) asks crowdworkers (creators) to create a user interface prototype (an artifact) [1].

The results revealed challenges that requesters face when collaborating with crowdworkers. First, differences in expertise can lead to challenges in terms of both communication style and language: expert requesters tend to provide their requirements asynchronously, making it challenging for workers to recall all the request, while the language used often includes domain-specific terminology. Second, having workers and a requester work together on a shared artifact, as done in typical real-time groupware, led to the loss of the visual context and implicit requirements of the original request. Third, asymmetrical means of communication, where requesters can speak aloud while crowdworkers can only type, produced confusion in their communication, since typed messages can be displayed continuously on-screen while speech is heard once and lost thereafter. These findings are not tied to the context of UI design, and are immediately applicable to systems that employ a collaborative setting in which a requester and workers directly communicate to work on a shared artifact. I suggested design implications and tools for similar interactive systems, such as real-time groupware and live social media that has similar asymmetry in communication (e.g. Twitch, Facebook Live) [2].

REDUCING THE BARRIERS TO ENTRY OF LIVE COLLABORATION

Taking this approach a step further, I have explored ways to include non-experts in the creation of artifacts that require domain expertise in live collaborative settings. Specifically, I have developed interactive systems that allow non-experts to create an interactive UI prototype and facilitate audience participation in a musical performance.

SketchExpress: Remixing Animations for Crowd-Powered Prototyping of Interactive Interfaces

I created *SketchExpress,* a crowd-powered prototyping tool to augment early sketches with interactive behaviors created by non-expert crowdworkers [7]. In this crowd-powered system, requesters verbally describe desired interactive UI behaviors while crowdworkers collaboratively demonstrate these behaviors in real time based on the given description (Fig. 3). However, manual demonstration is problematic: some behaviors, for example, are too complex to demonstrate manually and the demonstrated behaviors are ephemeral. To resolve these challenges, SketchExpress provides a novel interaction method that lets crowdworkers demonstrate these behaviors and remix their demonstrations to preserve the interactivity of the manual demonstration and improve the fidelity of the animation. The generated behaviors can be replayed by UI designers. SketchExpress allows workers to create a complex animation within 2.7 minutes on average, with a 27.3% increase in quality (recall) compared to the manual demonstration. Once created, the artifact (sketch) contains expressive, reproducible behaviors and users can interact with the sketch with a single click. We were able to mitigate the challenge of real-time collaboration by incorporating asynchronous interaction techniques, such as record-and-replay and remixing (post-processing), which alleviates the real-time pressure of manual demonstration. (video: https://youtu.be/A_Pngz1mbDs)

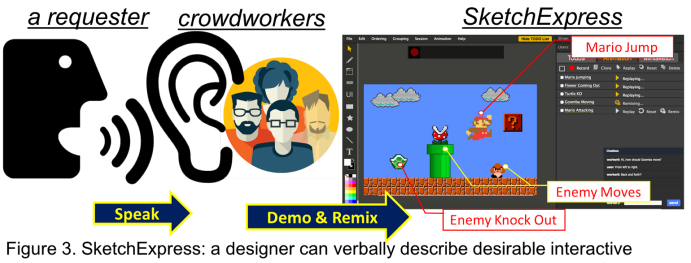

Figure 3. SketchExpress: a designer can verbally describe desirable interactive behaviors. Non-expert crowd workers can create interactive behaviors in a few minutes using **the demonstrate-remix-replay** method. The final sketch will have a set of animations that demonstrate multiple behaviors.

Crowd in C[loud]: Audience Participation in Music Performance

Musicians have long sought to create musical performances which involve an audience as part of the music-making process, as this is an effective way of engaging the audience. For example, in popular music, musicians induce audience participation by making sounds directly such as singing, clapping, or stomping feet (e.g. Queen's "We Will Rock You"). Audience participation is not just a method for music performance, but can be expanded to different contexts, like the theatrical arts, classrooms, and public events. I have developed computational systems for audience-participation music pieces in which an audience *is* the only musician, collectively generating sound for a music piece at a concert hall using smartphones [8]. There are many requirements in creating a large-scale participatory system through which audience members can perform a piece of music: (1) the system needs to be immediately accessible to non-experts, (2) the system needs to sustain the audience's interest in participation for the designated period, and (3) the system needs to allow musicians to orchestrate the audience members to perform a musical piece without interrupting their participation. The system that I developed for the piece *Crowd in C[loud]* provides the audience members with an ad-hoc social network within the musical instrument, using the metaphor of online dating. Each individual's short composition, which is looped, serves as a profile, and participants browse through each other's

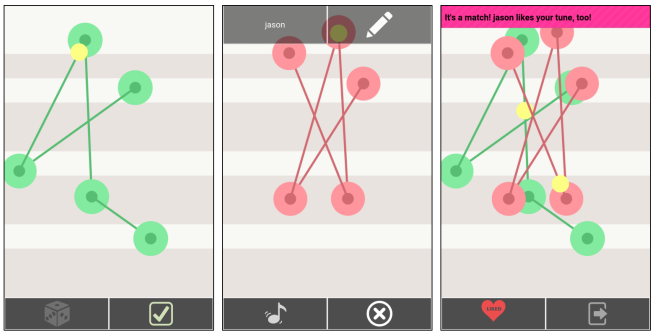

Figure 4. a web-based musical instrument for the audience participation music piece *Crowd in C[loud]*. The short composition (left) serves as a profile in a social media and users can browse other people's profiles (middle) and play together (right). The social interaction among audience members help sustain their interest for the duration of the performance.

profiles (Fig. 4). This allows audience members to socially interact with one another and to experience a process similar to how musicians explore various musical ideas, instantly facilitating collaborative music making at public events. As large-scale audience participation, hundreds of short tunes played on smartphones can create an ambient sound in a predetermined chord (C major, for example) and a musician on stage can remotely change the mapping (i.e. range of pitches) of the musical instrument to follow the chord progression of the composition. The key component that mitigates the challenges of participating in a live music performance is the asynchronous interaction of using social media in participation, as opposed to playing musical instruments, a time-sensitive activity requiring a high degree of synchronization with other musicians. (performance video: https://youtu.be/8nnrKJ4Ap0c)

BRINGING LIVENESS TO ASYNCHRONOUS COMMUNICATION

In the systems described above, the immediacy and the visibility of the creative process to users constitutes liveness—but when we cannot present an artifact to users in real time, how can we maintain the benefits of liveness? I searched for answers to this question through systems that preserve liveness in asynchronous communication. Writing is an expressive process guided and adapted by thoughts that evolve over time. The dynamic process of creating a written artifact can be informative and expressive, and the real-time history of a written work contains information not available in a static copy. For example, watching an instructor live-coding a program in a classroom reveals the temporal order of how the program is constructed and reflects the programmer's dynamic thought process, which the instructor wishes to convey. However, existing methods of archival, such as file saving, source version control, and even undo stack trees, do not archive the real-time history of how a written artifact has evolved.

In my *Live Writing* project, I developed a web application that extends existing web-based text editors. The application records writing activity at a high resolution of detail, such that the writing process can be reproduced in real time and replayed later [9]. In the previous example of live coding in a classroom, the *Live Writing* system allows instructors to create such a replay, which can be shared asynchronously with students via a link. Furthermore, the system can play an important role in the writing process for writers themselves, enabling them to recover context when resuming a writing task or when writing collaboratively. I have explored the effects of *Live Writing* in supporting collaboration in live coding [9] and in academic writing [10]. Currently, the application is deployed publicly. (demo: http://echobin.com). The idea of *Live Writing* has been explored as a form of performing art, in which a musician performs an audiovisual piece by writing a poem live on stage in a web-based editor I developed [11]. A visualization technique is developed to deform typography dynamically in response to any real-time data, e.g. audio or sensor readings, using GLSL shader language[12]. This piece won the International Computer Music Association (ICMA) Music Award 2016. (Fig. 5, videos and demo: https://livewriting.github.io)

Figure 5. *Live Writing : Gloomy Streets.* Writing a poem on stage live in front of the audience becomes an audiovisual performance. The piece uses temporal typography to enhance communication with the audience.

I also explored the possibility of preserving liveness through asynchronous collaboration in a programming context. Codeon, my recent project in collaborative programming, implements an asynchronous on-demand support system [14]. In Codeon, programmers can ask programming questions to expert programmers without leaving the integrated development environment (IDE), and the question generated includes a replay of the question being asked and the code highlighted during the speech act, simulating in-person communication in a pair programming session. The in-lab user study showed that the system reduced time and effort in seeking help for programming questions by 70% compared to state-of-the-art tools. (video: https://youtu.be/gDW3hK3YlP0)

FUTURE WORK: COMPUTATIONAL TOOLS FOR EFFORTLESS SELF-EXPRESSION

My future research vision springs from two questions: how can we scale the collaborative, expressive and live nature of music making to broader computational systems for the general public; and how can we computationally reproduce the power of music to be an effective medium that helps people connect to others? For now, only a small set of people can reach the level of skill needed to fully and fluently express themselves—not just through music but in most creative domains, including programming, writing, and design. My research goal for the next decade is to provide users with computational tools that help them easily express themselves and help them connect with others through their expressions augmented by the tools. My research on computational systems that support live collaboration is applicable to these research goals. I propose a few immediate projects as follows:

Intelligent Writing Environment. The real-time history of writing can be seen as a signal that changes over time, and this signal can represent the current state of the writing process, similar to eye-tracking data. For example, the length of a document changes in a different way when the writer adds all-new content to it versus when the writer is making minute changes while proofreading. Similarly, a program's code revision patterns are different when a programmer is debugging code versus when they are refactoring the code. I plan develop writing and programming environments that understand writers' real-time activity, recognize meaningful incidents, and provide relevant just-in-time support. The *Live Writing* project provides a source of data with which I can begin to study such patterns for research and later the tool can be extended as an intelligent writing environment.

Programmable Interactivity in Social Media. The emergence of novel social media like *Slack* and *Snapchat* accounts for users' diverse needs of communication in different settings. However, communication over social media is monotonic compared to in-person interaction. Their needs to actively communicate with their social network can be addressed by having programmable interactivity in their postings so that they can interact with their audience in more expressive ways than text alone allows. In that sense, I envision social media as an intriguing programming platform for end users to create interactive media with the unique access to their social network. The desires to connect to people with more interactivity will motivate general public to learn computational thinking.

[1] **Lee, S. W.**, O'Keefe S., Krosnick, R., Keelean, B., Park, S. Y., Lasecki, W. S. "To Whom Am I Speaking? An Exploratory Study on Real-Time Collaboration in a Crowd-powered Sketching Tool." *(Currently in review)*

[2] **Lee, S. W.**, Chen, Y., Lasecki, W.S. "The Needs for Real-Time Crowd Generation of Task Lists from Speech." *(Work in progress)* in AAAI Conference on Human Computation and Crowdsourcing (**HCOMP**) 2017.

[3] **Lee, S. W.**, Freeman, J. "Real-Time Music Notation in Mixed Laptop-Acoustic Ensembles." *Computer Music Journal* (**CMJ**) 2013.

[4] **Lee, S. W.**, Essl, G. "Live Coding The Mobile Music Instrument." The International Conference on New Instruments for Musical Expression (**NIME**) 2013.

[5] **Lee, S. W.**, Essl, G. "Communication, Control, and State Sharing in Networked Collaborative Live Coding." The International Conference on New Instruments for Musical Expression (**NIME**) 2014.

[6] **Lee, S. W**., Chen, Y., Klugman, N., Gouravajhala, S. R.,Chen, A., Lasecki, W.S., "Exploring Coordination Models for Ad-Hoc Programming Teams", *(Work in progress)* in the ACM Conference on Human Factors in Computing Systems(**CHI**) 2017.

[7] **Lee, S. W.**, Zhang, Y., Wong, I., Yang Y., O'Keefe, S., Lasecki, W.S. "SketchExpress: Remixing Animations for More Effective Crowd-Powered Prototyping Of Interactive Interfaces." The ACM Symposium on User Interface Science and Technology (**UIST**) 2017.

[8] **Lee, S. W.**, Carvalho, A. D., Essl, G. "Crowd in C[loud]: Audience Participation Music with Online Dating Metaphor using Cloud Service." The Web Audio Conference (**WAC**) 2016.

[9] **Lee, S. W.**, Essl, G. "Live Writing: Asynchronous Playback of Live Coding and Writing." The International Conference on Live Coding (**ICLC**) 2015.

[10] Blackwell, A. F., Cox, G., **Lee, S. W.** "Live Writing the Live Coding Book." The International Conference on Live Coding (**ICLC**) 2016.

[11] **Lee, S. W.** Live Writing: Gloomy Streets. (*Performance*) **CHI** Art Exhibition 2017.

[12] **Lee, S. W.**, Essl G. Web-Based Temporal Typography for Musical Expression and Performance. he International Conference on New Instruments for Musical Expression (**NIME**) 2015.

[13] Chen, Y., **Lee, S. W.**, Xie, Y., Yang, Y., Lasecki, W. S., Oney, S. "Codeon: On-Demand Software Development Assistance." The ACM Conference on Human Factors in Computing Systems (**CHI**) 2017.