NORTHWESTERN UNIVERSITY

Human-AI Interface Layers: Enhancing Communication of Intent for AI-Assisted Creative Pursuits and Social Experiences

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Technology and Social Behavior

By

Ryan Louie

EVANSTON, ILLINOIS

December 2023

Abstract

Human-AI Interface Layers: Enhancing Communication of Intent for AI-Assisted Creative Pursuits and Social Experiences

Ryan Louie

While early uses of artificial intelligence (AI) aimed to automate repetitive and burdensome tasks, AI shows great potential in assisting users in human domains of personal meaning and importance, from creative pursuits to social experiences. These human domains require that AI have an understanding and sensitivity to a user's personal ideas and knowledge. To assist a novice creator who lacks formal training, a generative AI partner must understand the person's creative ideas. An AI agent can appropriately identify opportune moments to connect with friends/colleagues from afar, only if it has cultural and tacit knowledge of the situations and activities in which people desire to connect. Despite significant advances in AI capabilities, conventional interfaces for AI can make it challenging for people to communicate their ideas and expectations. This dissertation proposes a crucial layer in an AI-powered application's stack called the Human-AI Interface Layer, which reconfigures existing AI capabilities to empower people's effective communication.

They provide (1) intuitive constructs for communicating ideas in ways AI systems can act upon and (2) interactive tools for forming, reflecting upon, and clarifying their intentions.

I developed three Interface Layers to enhance communication of intent to existing AI capabilities. First, users struggle to convey fine-grained semantics and control pieces of a generated artifact, when the AI's interface only allows indirect influence over how the entire artifact is generated. To resolve this, I propose a Steering Interface Layer that partitions and constrains generated outputs to support incrementally guiding AI outputs toward desired directions. Second, when a person's ideas for their AI-enabled creation are conceptually-rich and sometimes vague, people can struggle to translate these high-level concepts to the lower-level constructs for AI systems to understand and act upon. As such, I contribute an Expression Interface Layer equipped with cognitive bridging tools that help people address this semantic gap by fleshing out their high-level concepts and foraging for relevant and precisely operating AI constructs. Third, when an individual provides explicit instructions to an AI agent, the agent's actions during execution can still disappoint when it encounters constraints that prevent implicit expectations from being satisfied. To resolve this, I developed an Execution Interface Layer for recognizing and stating implicit expectations so an AI agent can appropriately adjust after handoff.

More generally, a mismatch might always exist between what people envision for human pursuits and what AI assistants can currently understand about them. This thesis shows that Human-AI Interface Layers for intention formation and articulation can empower creators and designers to communicate their ideas and intentions for AI assistance, despite existing AI's inherently limited interfaces for doing so.

Acknowledgements

If ever there was a group of people so diverse in their qualities but all consistent in their support throughout the years, it would be my thesis committee. My advisor, *Haoqi Zhang*, taught me how I can aspire to be a better version of myself if I want it. It was the lesson I needed to hear when the strengths I already had were only getting me so far. He also always reminded me that I'm more capable than I might realize. That was the truth I needed to hear when I was being self-critical without recognizing all the ways I have achieved great work, in no small part, through my own doing. I'm so grateful for him taking a chance with me as an early scholar; always giving me such ample time and feedback when I needed it; and helping me appreciate the value of conceptual work beyond the design and technical work that I initially interpreted as the only ”real” work.

Darren Gergle taught me how to be rigorous in statistical methods and how to faithfully adapt social science research for use in design and HCI. Most importantly, he taught me that being part of a research community meant feeling entitativity to its community members, which can attained through a commitment to learning individuals' personalities and ideas. *Carrie Cai* taught me what it meant to be a great leader in Human-AI Interaction, and showed me the value of bringing diverse people together who cared about empowering people through the advancements in AI. She taught me that people who aren't formally trained in HCI have intentions to empower humans with AI, too; that making a bigger impact requires teaming up with those people in your shared mission;

and that my perspective as an HCI system builder and thinker is something I should cultivate and share with others.

Jeremy Birnholtz has always taken the time to share his written thoughts on my drafts before the actual presentation or defense. That explicit engagement in my drafts has always invigorated me about how awesome a place academia can be, full of thoughtful individuals who are providing written feedback about what ideas resonate, and how they stir up new questions within them. That deep academic engagement made my research ideas feel seen, appreciated, and improved beyond measure. *Andrés Monroy-Hernández* gave me a deeper appreciation for how to shape my research thinking by keeping a meaningful connection to what is going on in the world at the current moment. He has taught me to be more strategic about which work I put out into the world depending on whether the world is ready for it. I have been emboldened by his encouragement for researchers to pursue systems work that makes for playful and meaningful experiences. Finding camaraderie with others like himself who were seriously working on playful research helped me to accept myself and my research a little bit more.

Whereas a good mentor is supportive and provides direction, it is your peers who are ever-present and inspirational to work with every day. I was lucky to have peers who taught me as we explored research together. Special thank you to the wonderful colleagues in the Delta lab, including Spencer Carlson, Melissa Chen, Katie Cunningham, Kapil Garg. Gobi Dasu, Eureka Foong, Jamie Gorson, Emily Harburg, Garrett Hedman, Evey Huang, Yongsung Kim, Harrison Kwik, Nick LaGrassa, Dan Rees Lewis, Lauren Lin, Kristine Lu, Leesha Maliakal Shah, Caryn Tran, Gus Umbellino, and Morgan Wu. Thank you to the community of students in the Design Technology Research program

at Northwestern. You are energetic, reflective, and supportive beyond measure. Thank you to the members of the Opportunistic Collective Experiences special interest group at Northwestern for allowing me to serve as your mentor and for putting your all into advancing a research agenda with me. Thank you to my collaborators Taewook Kim and Zhenhui Peng for helping me develop my current and future interest in AI for Mental Health. Thank you to my friends and colleagues in the Technology and Social Behavior program at Northwestern. Thank you to Northwestern CS friends who made Mudd a welcoming place to call home.

Aside from all the researchers, the community I made while living in Evanston, Chicago, and Palo Alto has made life so much better during these past years. In between all the work, you filled life with brilliance, warmth, and adventure. A special shout out to the players in the NU Table Tennis Club, which was a constant community throughout graduate school. I am so lucky to have been able to train, compete, and travel at a collegiate level for a sport I've loved playing for 17+ years. More than just a healthy distraction from research work, Table Tennis has been an important space for me to channel my abilities to take on seemingly improbable goals one point at a time. It's also a space where I learned the value of feeling ecstatic *losing a match* when I knew my performance was much better than I have ever done despite the outcome. Staying in touch with these truths has always had a positive spillover to my other personal and professional pursuits.

Without my parents, none of this would be. You have been always so encouraging of me to explore my passions, even if so many of them were whimsical and quirky. Giving

me the permission to go as deep as I wanted in anything planted the seeds for my interests and persistence in becoming a design researcher and computer scientist.

Meeting my partner, Akhila Kovvuri, while attending Northwestern was probably the most fortuitous thing to happen to me. The second fortuitous thing to happen was finding a fantastic research postdoc position that resolves a 3-year, long-distance relationship! I can't wait to shape a future together and continue to bring out our best selves in each other.

While people are the most important part of my life, one organization gave me the freedom to spend my 6th year in graduate school really embarking on this intellectual journey of developing this dissertation. Thank you to the Google PhD Fellowship in Human-Computer Interaction for your support.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Advances in AI are enabling developers to integrate a variety of AI capabilities into user-facing applications. For example, the advent of generative AI models for producing coherent and high-quality outputs has opened up opportunities to integrate them into a variety of content-production applications. Amidst growing concerns around the harms that generative AI might cause in automating content production entirely, more designers and researchers have begun emphasizing AI's usefulness as a co-creator and an assistant to enhance, rather than replace, human creators. While some AI-powered applications use AI in interactive and collaborative uses, other AI advances are used in systems running in the background, that opportunistically surface information and proactively take action. For example, the accuracy of pattern recognition, along with the ubiquitousness of personal data and sensors, has created opportunities to integrate contextual inferences into AI-powered applications for tracking important attributes and surfacing contextually relevant reminders, content, or experiences. Such inferences are based on large labeled datasets of specific categories of interest [**82**], and often leverage pattern recognition algorithms to infer contextual attributes along dimensions of activity [**60, 93**], mood [**108**], and environment [**138, 112**]. Since these AI-powered applications are performing inferences in moments that application designers may not be constantly overseeing, there can be additional challenges in how to ensure their appropriate operation when deployed.

While early efforts developing AI-powered applications have aimed to automate the completion of mundane and burdensome tasks, there is a growing awareness that the focus should be on broadening and enabling engagement in personal pursuits in which people find meaning. Examples of such AI-enabled human pursuits include composing music and socially connecting with geographically distant friends and family. AI capabilities can lower the barriers for users to engage in personal pursuits or participate in these types of social activities by taking on tasks humans may struggle to do themselves. For creative experiences, while the pursuit of music composition typically requires extensive music theory knowledge, generative AI capabilities can aid novice composers in making a harmonious song that expresses their creative goals. For social connection, it can be challenging to find the time or reason to plan a call or initiate a direct message with people who live at a distance (e.g., friends/family whom we no longer see regularly). However, recent research [**103**] has shown the potential to opportunistically connect such people through an intelligent context-aware agent. The agent identifies common situations across geographic distance that coincidentally arise—thus capturing more meaningful moments that without such an agent would have been missed. In other words, using AI capabilities to foster meaningful pursuits—from helping novices explore their creativity, to recreating across distance the feelings of connecting opportunistically with a friend—is an important frontier for the HCI and applied AI research communities.

In these pursuits, people are not interested in enlisting AI to simply create *something sensible*. Instead, they use AI to help them realize their specific creative ideas that without AI assistance would have been difficult or impossible. For example, a novice composer who lacks music theory expertise can partner with generative AI to help them co-create a song.

They want to create a song that not only sounds pleasant and evokes a certain emotion but also reflects their creative ideas for how the different musical elements should come together. In another example, a social experience designer aims to recreate across distance the feelings of connecting opportunistically with a friend by using a context-aware agent to identify coincidental moments when users share similar situations. Instead of using an AI agent to start interactions in moments when end-users share *any similar context*, a designer may want to use the AI agent to realize a specific vision for an opportunistic social experience at a distance—such as facilitating an experience based on geographically-distant friends grabbing a warm meal on a cold day.

To ensure their ideas are realized, people need to communicate them. Effective communication entails several steps. First, people need a modality to articulate their ideas in ways that AI capabilities can understand and act upon. Second, people need to form their ideas to effectively articulate them to AI systems. Ideas that remain vague will result in AI systems that have a surface-level understanding. Those that are articulated incorrectly will result in AI actions that do not align with their expectations. Thus, articulating ideas comprehensively and accurately is paramount for using AI to realize a creative vision.

However, many AI interfaces do not readily support such communication needs. One issue is that an AI interface can only provide a few constructs for controlling an AI capability. For instance, an existing generative AI for music may not allow people to request how they want AI to generate the rest of the song. Another issue is that an AI interface's constructs can be challenging for people to articulate their ideas with, even if the constructs can influence how an AI capability operates. For example, the set of context detectors for monitoring someone's context may not match a designer's idea for a situation they want to recognize for a shared experience.

Researchers in the AI community have made recent progress in improving the interface between AI and humans by creating more communicative and collaborative AI agents that can adapt their actions to the preferences, goals, and behaviors of users. One recent approach that provides people with an intuitive modality for communicating intentions to an AI capability is large language models. For example, generative AI models like ChatGPT [**110**], DALL-E [**114, 113**], Stable Diffusion, Text-to-Music models like MusicLM [**3**] can now understand a user's instruction or prompt to generate text, images, and music accordingly. This has opened up many possibilities for personalized and customized generated content. While LLM-based AI capabilities have made it easier to articulate ideas to AI capabilities, they place the burden on users to form their ideas to communicate. This can create issues if such AI systems jump to conclusions or take action without comprehensively and precisely understanding a user's intended meaning. For example, a generative AI model could automatically produce creative artifacts based on a user's prompt (e.g., a painting that conveys the emotions 'stuck, reflective, sad') without leaving room for the user to clarify or conceptualize how they want to evoke such feelings in the artifact. In summary, even if an LLM-based interface exists for articulating ideas to an AI capability, communication requirements like forming intentions must also be addressed for people to realize *the full extent of their ideas* for AI-enabled creations.

In parallel to the research happening in the AI community, the HCI community has been interested in studying and supporting the interaction between humans and AI systems, and more broadly putting human experience and goals at the forefront of AI application development. A body of HCI research has proposed human-centered methods, guidelines, and strategies, some of which include identifying stakeholders' requirements before implementing an AI application, forming close collaborations with AI experts to assess which interactions and features can be supported by current AI capabilities, sketching out application interfaces populated with realistic AI outputs [**135**]. Of particular relevance are generally applicable human-AI guidelines for designing interactions with AI systems which can be used to evaluate and correct usability and interaction issues with AI-powered features [**6**]. While we certainly agree with these guidelines for what interactions would be helpful, challenges still remain in applying such guidelines to improve the interactions with real AI systems, especially when the default interfaces to an AI capability, such as an API, do not readily support such interactions.

In summary, interfaces of an AI system can lack intuitive or actionable constructs for creators to communicate their ideas. Conventional interfaces can also place the burden entirely on the user to form their intentions more comprehensively and precisely. And while there are suggestions to improve interactions with AI, these guidelines can be difficult to implement if an AI's interface does not support such interactions. Therefore, building AI-powered applications without addressing these deficiencies in the communication interface with AI will result in AI-enabled artifacts and experiences that do not reflect the intended idea. Suppose composers cannot convey their creative ideas through a generative AI interface. In that case, this readily detracts from the joy a composer can receive from crafting an artifact and turning their ideas into reality. Similarly, suppose designers cannot communicate their ideas about what situations would recreate the feeling of sharing a similar experience with a friend. In this case, an AI agent will be limited in how it can identify and facilitate opportunities for end-users to connect meaningfully.

1.1. Thesis and Contributions

This dissertation proposes a missing layer in an AI-powered application's stack called *the Human-AI Interface Layer* to improve the communication interface between people and existing AI capabilities. The Human-AI Interface Layer reconfigures the user interface to an AI capability with intuitive and actionable constructs for controlling an existing AI and scaffolds for helping people to recognize and form their intentions. Rather than relying on new AI capabilities to be trained to understand and operate on a new and expanded set of constructs, a Human-AI Interface Layer approach can expose intuitive and functional constructs for adjusting how existing AI capabilities operate to support the more straightforward articulation of ideas. Moreover, instead of relying on AI systems to take most of the initiative to infer a person's intentions—which may cause AIs to jump to conclusions and take action without a comprehensive or precise understanding of the ideas and expectations—a Human-AI Interface Layer approach provides user interfaces explicitly aimed at giving people more initiative to form, evaluate, and clarify the ideas they will convey to AI.

In this thesis, I describe three example classes of problems faced while conveying an idea and intent to an AI system. These challenges arise due to the interfaces of existing AI systems not meeting people's requirements for having intuitive and actionable constructs for communicating their specific ideas, and support for a back-and-forth process for forming and clarifying intent with an AI system. For each example challenge, I introduce my proposed Human-AI Interface Layer for resolving it. Each layer accomplishes this by augmenting an existing AI system with new capabilities and reconfiguring its interface to meet a user's communication needs.

- First, when the interface to an AI capability only affords indirect influence over coarse-grained objects, such as fully-completed creative artifacts, people can struggle to evoke their fine-grained and opinionated ideas and progressively engage in taking the initiative when partnering with AI to co-create an artifact. To resolve this, I propose a *Steering interface layer* that partitions and constrains generated outputs along semantically-meaningful dimensions and exposes user interfaces for incrementally guiding AI outputs toward desired directions.
- Second, when a person's ideas for their AI-enabled creation are conceptually-rich and sometimes vague, people can struggle to translate these high-level concepts to the lower-level constructs for AI systems to understand and act upon. As such, I argue that an *Expression Interface Layer* can augment user interfaces for using AI capabilities with cognitive support tools that help people flesh out their high-level concepts and forage for relevant AI constructs which accurately represent their concepts, leading to fuller and more precise encodings of their conceptually-rich ideas.

- Third, when an individual provides explicit instructions to an AI agent, the agent's actions and decisions during execution can still disappoint due to encountering constraints that cause breakdowns in the individual's implicit expectations. To resolve this, I developed an *Execution Interface Layer* for communicating about execution constraints to support the individual in recognizing and stating implicit expectations so an AI agent can appropriately adjust after handoff.

Together, these Human-AI Interface Layers act as an intermediary so end-users and designers can articulate, form, reflect upon, and clarify their meaning and intention to the AI system for more aligned output and action. We demonstrate the value of the Human-AI Interface Layer in the context of two domains: (A) novice composers using a generative AI model to co-create musical artifacts; and (B) designers using an intelligent, context-aware agent to facilitate opportunistic social experiences between friends and family who live across distant contexts.

1.1.1. An Interface Layer for Incrementally Steering Generative AI

The first class of problem I consider is in regard to a user's goal to convey fine-grained qualities into different pieces of a generated artifact, despite the interface to an AI only allowing indirect influence over how the entire artifact is generated. I studied a concrete instantiation of this problem in the domain of generative AI tools for music co-creation, where a music creator wants to collaborate with generative AI to create a song that reflects their ideas and emotions. Conventional generative AI tools offer a user interface very similar to the AI's interface: they take as input some starting notes, and then based on these notes, output a completed song or musical piece. However, it's challenging and sometimes unobtainable to use these generative AI tools to create music that evokes a composer's creative ideas: when a composer wants to craft small chunks of the composition at a time, the AI generates the whole artifact all at once which makes it hard to edit and control how different chunks sound; and while a user wants to define fine-grained musical or emotional qualities in the song, they don't have controls to convey how they want the musical notes to be generated. In other words, if a generative AI interface only supports tweaking completed generative outputs as a whole—but does not allow the person to define how different parts, within the whole, should be generated according to different qualities—human creators will not be able to craft an artifact that captures their creative ideas nor derive inherent satisfaction from this creative pursuit.

To address this shortfall, a Steering Interface Layer reconfigures interactions with an existing AI model by exposing a control panel user interface that partitions AI-generated content into chunks and constrains what the AI generates based on several semantically-meaningful dimensions. The Steering Interface Layer was designed to enable creators to incrementally direct the creation process in real time. We demonstrate this through Cococo, a music composition application for using a deep generative AI model augmented by a Steering Interface Layer that supports partitioning generated musical notes to particular voices and time steps; constraining outputs along semantically-meaningful directions (happy/sad; similar/different; conventional/surprising); and auditioning multiple alternative generations. Technically, we contribute a soft priors technique for adjusting the sampling distribution of an AI model in accordance with semantic constructs without retraining the underlying model—thereby allowing tool designers to expose a range of musically-meaningful constructs. Evaluative studies of the Steering Interface Layer show how partitioning outputs allows people to work on a chunk of the whole artifact at a time, which makes thinking about creative intentions and discovering strategies for realizing them more manageable. We find this gives people the control they need to communicate their creative intentions for the AI to follow, which helps them to feel greater self-efficacy, creative ownership, trust, and collaboration with the AI.

1.1.2. An Interface Layer for Expressing High-Level Concepts using Low-Level Constructs

The second class of challenge occurs when the users' ideas for what they are trying to create are at many levels of abstraction removed from the constructs for configuring an AI system. I consider a manifestation of this problem for context-aware AI systems, where AI capabilities can infer lower-level features of context (e.g., locations), but not the activities or experiences end-users could be engaging in. When training new AIs with better-fitting constructs isn't feasible, the burden is on individuals to bridge from their high-level idea to the AI's low-level constructs. Such is the case for conventional programming interfaces for context-aware systems: while an individual can readily define simple situational triggers using low-level objects (e.g., a "restaurant serving soup"; a "park"), they can struggle to express high-level concepts of human situations and the socio-cultural ideas about the activities or experiences they afford (e.g., "family can share warm comfort foods on a cold day"; "a place to have a picnic"), which are several levels of abstraction removed.

To overcome challenges bridging the semantic gap between human ideas and the constructs provided by an AI system, the Expression Interface Layer provides a designer with cognitive bridging tools that address challenges that arise when forming one's high-level ideas and finding relevant and precisely-matching machine constructs for one's concepts. We built an Expression Interface Layer into Affinder, a visual programming environment for expressing concepts of situations using location context detectors that intelligent, context-aware agents can computationally act upon. To prevent concepts from being too narrowly defined, Affinder has prompts for reflecting and expanding a designer's conception of the situation, inspired by analogical design techniques for re-representing ideas through more abstract schemas. To ensure they can find all relevant AI constructs for a concept, Affinder provides an unlimited vocabulary search that uses textual metadata associated with the context features (e.g., Yelp Reviews of the location categories) which provides designers a rich vocabulary for querying for relevant constructs based on their conception. To mitigate concept expression inaccuracies, Affinder includes simulation and repair tools for recognizing misconceptions and ensuring the machine representation precisely operates as the designer intended. We find that the Expression Interface Layer's cognitive supports can help designers richly express a high-level situation by helping them define a more expansive set of concepts for ways to realize it (e.g., eating spicy food and soupy foods are distinct ways to foster the feeling of grabbing warm comfort foods on a cold day). The cognitive supports also help to find relevant and accurate links between their intermediate concepts of a situation and the features that a context-aware AI system can understand and detect.

1.1.3. An Interface Layer for Recognizing and Stating Implicit Expectations for Execution

Even when an individual has expressed instructions for an automated system to carry out, automated systems can take action and make decisions that are disappointing due to uncommunicated user expectations. For example, an automated system can automatically plan an itinerary for a user, but the itinerary it outputs has packed too many activities in one day which a person may not have the energy to do [**137**]. I examine this third class of problem in the domain of context-aware AI agents facilitating opportunistic social experiences: While an intelligent agent can proactively engage end-users when their context meets the situational requirements (e.g., when located at beaches), a contextually-triggered experience might be inaccessible to end-users in a target population if the situational requirement is hard to meet based on the region they live in (e.g. people living in landlocked cities). Such disappointments during execution can be attributed to the user interface for AI agents placing the burden on the individual to state their implicit expectations, with little support for helping them recognize when they've forgotten to state an idea that is important during execution. To make matters more difficult, an AI agent may not provide a means for people to communicate their expectations to the AI agent so it can respect them. For example, a context-aware agent may only provide constructs for defining who can be involved in a multi-person opportunistic, social experience, but few ways to encode expectations about the maximum time in which participation should be left unreciprocated, let alone mechanisms for controlling how an AI agent triggers to respect these expectations.

To improve communication about implicit expectations, we developed an Execution Interface Layer that augments the user interfaces for programming an AI agent to support people in either (1) recognizing how implicit expectations could be impacted by how they've currently articulated their explicit ideas, or (2) stating those implicit ideas through new AI mechanisms that can adapt an existing AI's execution behavior to achieve a balance of explicit and implicit expectations. We developed different tools and mechanisms that improve the communication of implicit ideas so context-aware coordination engines can respect such interaction norms. We created a tool to help people adjust their situational requirements for an experience to balance their explicit and implicit expectations for access and inclusiveness across different regions. Moreover, we developed a decision-theoretic mechanism to determine if launching an AI-powered experience will lead to timely participation. It is based on a model of people's likelihood to participate and a designer's ideas about the importance of timely participation. These examples illustrate how an Execution Interface Layer can complement existing AI capabilities for coordinating opportunistic experiences and ensure that people and AI systems are taking into account the implicit ideas when making decisions.

1.2. Thesis Overview

This dissertation details our contributions to developing Human-AI Interface Layers that address three example classes of problems when communicating intentions and ideas for AI systems to enact. I demonstrate the benefits of these interface layers in the context of two domains: (A) novices using generative AI to explore their human creativity;

and (B) designers using context-aware AI agents to surface and realize the potential for coincidental, meaningful connections across geographic distances.

(1) Chapter 2 will describe the design and development of a Steering Interface Layer, which reconfigures a default interface for generative AI by partitioning generated outputs and providing semantic knobs for controlling the AI model in the domain of music. Through an evaluation study, we found that the Steering Interface Layer significantly improved feelings of self-efficacy in achieving creative goals, collaboration with AI, and agency when co-creating.

(2) Chapter 3 will show the need for an *Expression Interface Layer* that supports a designer's cognitive process in fleshing out high-level concepts for a context-aware experience and linking those to machine detectors that an AI system can use to recognize the situation across distributed contexts. This chapter introduces Affinder, which is a visual programming tool that allows designers to encode their ideas for situations for usage in a mobile, location-based context-aware application. Affinder's core features address cognitive challenges designers face when expressing their high-level intentions for an experience—which are social and cultural ideas that are many levels of abstraction removed from the context-detectors made available by existing context APIs and frameworks.

(3) Chapter 4 will show that human elements for coordinating an opportunistic, social experience, such as social norms around participation, are typically implicit, and need to be stated to ensure AI systems can promote desirable social interactions under real-world conditions. This chapter describes how an interface layer for recognizing and stating tacit expectations for execution can address

breakdowns in social interdependence that occur when coordinating opportunistic interactions that are governed by the uncertain availability of participants.

(4) Chapter 5 provides a discussion of the dissertation as a whole, summarizing key contributions, design principles, and research methods for Human-AI Interface Layers. In addition, we illustrate how the three example classes of challenges can manifest in other domains and discuss how to generalize and extend the approaches to apply to new domains and account for the emergence of new AI capabilities like Large Language Models. This chapter ends by emphasizing the role of Human-AI Interface Layers in preserving human's engagement in these human endeavors where craft, self-expression, and mastery is valued.

(5) Chapter 6 concludes the dissertation with a succinct restatement of the thesis, the three types of Human-AI Interface Layers that were advanced, and the fundamental idea of enhancing people's ability to communicate their goals for personal endeavors despite limitations in existing AI capabilities and interfaces.

1.3. Prior Publications and Authorship

Although I am the primary author of the research detailed in this dissertation, it is also the product of years of collaboration with my primary co-advisors, Haoqi Zhang and Darren Gergle, and my collaborators at Google Research. The work on Steering Interfaces for Generative Models for Music appeared at CHI 2020 [**101**] and was based on my internship at Google Research with Andy Coenen, Anna Huang, Michael Terry, and Carrie J. Cai. The work on the Expression Interface Layer instantiated through Affinder [**104**] appeared at CHI 2022 and was a collaboration with Darren Gergle and

Haoqi Zhang. The initial work behind the Execution Interface Layer chapter was done in collaboration with Kapil Garg, Darren Gergle, and Haoqi Zhang, and at the time of thesis writing is being prepared for resubmission. To reflect my collaborators' contributions, I use the first-person plural ("we") in these chapters.

CHAPTER 2

An Interface Layer for Incrementally Steering Generative AI

In this chapter, we investigate how people can struggle to impart their fine-grained and opinionated ideas into an AI-created artifact when the AI's user interface only allows generating all the content at once, and adjusting how outputs are generated in a coarse-grained and indirect manner. We study this issue in co-creating music with generative AI, where people would like to take initiative and exercise fine-grained control when crafting elements of a creative artifact—yet, the default interfaces to using generative AI models automate many of the steps of the creation process that are important for people to feel greater ownership and create an artifact that reflects their creative decisions.

Broadly, the chapter highlights that a *Human-AI Interface Layer for steering during co-creation* can reconfigure the interaction with powerful AI that automatically generates fully-completed end-products by partitioning and constraining its existing outputs. This supports fine-grained and incremental control of creative intent, thereby preserving the human involvement needed for crafting a personal artifact and engaging in an inherently valuable creative experience.

2.1. Introduction

Rapid advances in deep learning have made it possible for artificial intelligence (AI) to actively collaborate with humans to co-create new content [**109, 31, 53, 91, 58, 83**].

One promising application of machine learning in this space has been the use of generative deep neural network (DNN)-backed systems for creative activities such as poetry writing, drawing, and music creation—experiences that bear intrinsic value for people, but often require specialized skill sets. For example, by completing a drawing that a user has started [**109, 31, 87, 46**] or filling in a missing section of a song [**74, 59**], generative models could enable untrained lay users to take part in creative experiences that would otherwise be difficult to achieve without additional training or specialization [**76, 41, 54**]. In this chapter, we focus on the needs of music novices co-creating music with a generative DNN model.

While substantial work has focused on improving the algorithmic performance of generative music models, little work has examined what types of interactive capabilities users actually need when co-creating with generative AI, and how those interactive capabilities might affect the music co-creation experience. Recent generative music models have made it conceivable for *novices*, who have little or no formal experience composing music, to create an entire musical composition from scratch, in partnership with a generative model. For example, the widely available Bach Doodle [**76**] sought to enable anyone on the web to create a four-part chorale in the style of J.S. Bach by writing only a few notes, allowing an AI to fill in the rest. While this app makes it conceivable for even novices with no composition training to create music, it is not clear what people desire when engaging in co-creation activities like these, or what forms of interaction with AI might they find useful.

In a study we conducted to understand the human-AI co-creation process, we found that AI music models can sometimes be quite challenging to co-create with. Paradoxically, the very capabilities that enable such sophisticated models to automatically generate fully-completed music compositions from scratch can impede human partnership: Users struggled to evaluate and edit the generated music because the system created *too much* content at once; in essence, they experienced *information overload*. They also struggled with the system's *non-deterministic output*: While the output would typically be coherent, it would not always align with the user's musical goals at the moment. These findings raise critical questions about how to co-create with an AI that already matches or supersedes a novice's capabilities to create artifacts: What user interfaces and interactive controls are important, and what interactive capabilities should be exposed by deep generative neural nets to benefit co-creation?

To address these interaction challenges, we developed a Human-AI Interface Layer that augments the standard generative AI interface with **AI-steering tools** that partition outputs into meaningful chunks and constrain how the AI generates content along semantic dimensions. These AI-steering tools were designed to enable novice users partnering with a generative AI to iteratively direct the creation process in real-time. To ground this research, we developed Cococo (collaborative co-creation), a music editor web-interface for novice-AI co-creation that augments standard generative music interfaces with a set of AI-steering tools: 1) *Voice Lanes* that allow users to define for which time-steps (e.g. measure 1) and for which voices (e.g. soprano, alto, tenor, bass) the AI should generate music, 2) an *Example-based Slider* for expressing that the AI-generated music should be more or less like an existing example of music, 3) *Semantic Sliders* that users can adjust to direct the music toward high-level directions (e.g. happier / sadder, or more conventional / more surprising), and 4) *Multiple Alternatives* for the user to select between a variety of AI-generated options. To implement the sliders, we developed a *soft priors* approach that encodes desired qualities specified by a slider into a prior distribution; this soft prior is then used to alter a model's original sampling distribution, in turn influencing the AI's generated output without need to retrain the model

In a summative evaluation with 21 music novices, we found that AI-steering tools not only increased users' trust, control, comprehension, and sense of collaboration with the AI but also contributed to a greater sense of self-efficacy and ownership of the composition relative to the AI. Beyond improving user attitudes towards the AI, the steering tools also enabled new user strategies for music co-creation: participants used the tools to divide the music into semantically meaningful components; learn and discover musical structure; debug the music and the AI; and explore the limits of the AI.

In sum, this chapter makes the following contributions:

- We discover key interfacing challenges that music novices face when co-creating with a typical generative-DNN music interface that automatically generates a fully-completed artifact, including issues related to AI-induced *information overload* and its *non-deterministic output*.
- We present the design and implementation of AI-steering tools that enable users to progressively guide the co-creation process in real-time, contributing a *soft priors* technical approach that encodes desired qualities in a prior probability distribution to influence the AI's content generation, without needing to retrain the model.

- We find in a summative study with 21 users that the steering tools increase users' sense of ownership of the composition relative to the AI, while increasing trust, controllability, and comprehensibility of the AI.
- We describe new user strategies for co-creating with AI using these tools, such as developing new insights into composition strategies, isolating the cause of musical glitches, and exploring the limits of the AI. We also uncover novice considerations of agency and collaboration when co-creating with AI.

Taken together, these findings inform the design of future Human-AI Interface Layers for co-creation.

2.2. Related Work

2.2.1. Human-AI Co-creation

The acceleration of AI capabilities has renewed interest in how AI can enable human-AI co-creation in domains such as drawing [**109, 31, 87, 46**], creative writing [**53, 25**], design ideation [**91**], video game content generation [**58**], and dance [**83**]. For example, an AI might flesh out a half-sketched drawing [**109**], write the next paragraph of a story [**25**], or add an image to a design mood board [**91**]. Across this range of prior work, a core challenge has been developing collaborative AI agents that can adapt their actions based on the goals and behaviors of the user. To this end, some systems design the AI to generate output conditioned upon the surrounding context of human-generated content [**46, 25, 53**], while others leverage user feedback to better align AI behavior to user intents [**58, 91, 31**]. Research has also observed that users desire to take initiative in their partnership with AI [**109**], with controllability and comprehensibility being key challenges to realizing this vision [**6**]. Building on this need, our work enables users to express their preferences to an AI collaborator through a variety of means.

Much of the prior work in this space has focused on the domains of drawing or writing. Efforts examining human-AI collaboration for creating music have been relatively nascent [**57**], particularly with generative DNN music agents of similar prowess. Building on prior work examining AI as a peer in the creative process, our work contributes to the broader literature by investigating human-AI co-creation in music.

2.2.2. Interactive Interfaces for ML Music Models

To support music makers in the composition process, researchers have conceptualized and developed ML-powered interfaces that map user inputs to musical structures so users can interactively explore musical variations. Examples of such designs and systems include those that allow users to find chords to accompany a melody [**126, 56**], experiment with adventurous chord progressions [**75, 51**], control the similarity vs. otherness for retrieval of music samples [**7**], use custom gestural inputs to interpolate between synthesizer sounds [**48**], or turn free-hand sketches into harmonious musical textures [**47**].

More recently, progress in generative DNNs has introduced fully-generative music interfaces capable of performing auto-completion given a seed of user-specified notes [**59, 76, 118**]. Beyond supporting single sub-components, these systems can produce full scores that automatically mesh well with local and distant regions of music. Thus, there is potential to now support users in a wide range of musical tasks (e.g., harmonizing melodies, elaborating existing music, composing from scratch), all within one interface. While recent research has made these fully-generative interfaces increasingly available to musicians and novices alike [**59, 118, 76, 40**], there has been relatively little HCI work examining how to design interactions with these contemporary models to ensure they are effective for co-creation, especially for novices. Our research contributes an integrative understanding of how interfaces to these capable AIs can be designed and used, how these capabilities affect the composing experience, and users' attitudes towards AI co-creation.

2.2.3. Deep Generative Music Models

As their name implies, generative deep neural networks can synthesize content. Research has demonstrated the potential for modeling and synthesizing music, ranging from single-voice sequences [**45**] and multi-part music [**54, 96**], to music with variable parts at each time step [**16**] and music with long-term structure over minutes [**78, 111, 63**].

In contrast to models that (typically) generate music chronologically from left to right, *in-filling* models can more flexibly support co-creation by allowing users to specify regions at any point in the music, then auto-filling those gaps. Examples include DeepBach [**59**] and Coconet [**74**], both trained on four-part Bach Chorales. Researchers have also created models designed to support interaction mechanisms that grant users more control. For example, there are emerging approaches aimed at learning a continuous latent space so that users can interpolate between music [**117**], or explore a space of musical alternatives [**39**]. In our work, we adopt *soft priors* as a general approach that provides additional ways for users to direct their exploration. In contrast to hard constraints, our approach allows DNNs to simultaneously consider the original context (encoded in the model's original sampling distribution) and additional desired qualities (encoded in a soft prior distribution), without needing to retrain the model.

2.3. Formative Needfinding Study

Our research focuses on enabling novices to engage more creatively with music, without the prerequisite understanding of musical theory and composition. Thus, we conducted a 45 minute formative interview and elicitation study with 11 novice music composers to understand 1) their motivations and needs for creating music themselves and 2) challenges in co-creating with AI composing tools. We recruited participants from our institution using mailing lists and word-of-mouth, screening for individuals who had played a musical instrument at some point in their life: 9 participants had five or more years of experience playing a musical instrument; 8 had no formal experience in composition and had informally experimented with musical arrangements using music software or improvising on an instrument; and 2 had tried creating a small composition as part of a music theory class assignment.

2.3.1. Motivations and Needs for Creating Music

Our participants reported the desire to create music to complement or enrich existing personal artifacts or experiences, such as creating an accompaniment to a short personal video or photo album, a composition inspired by a poem, or a theme song for a friend or loved one. Participants who had attempted creating music on their own encountered challenges due to their lack of training in music theory and composition. Oftentimes, they knew something needed to be created or fixed (e.g., adding harmonies), but lacked the expertise to identify the issue, a strategy for solving the problem, and/or the ability to generate viable solutions. These challenges suggest specific ways AIs could aid users and make them more capable.

2.3.2. Challenges in Co-Creating with Generative DNNs

In the second half of the study, we conducted an elicitation to understand challenges when interacting with a deep generative model to compose music. The interface mirrored the generative *infilling* capabilities found in conventional interfaces for deep generative models [**76**], where users can manually draw notes and request the AI to fill in the remaining voices and measures, or erase any part of the music and request the AI to fill in the gap. Overall, we found that users struggled to evaluate the generated music and express desired musical elements, due to *information overload* and *non-deterministic output.*

2.3.2.1. Information Overload. While the deep generative models were capable of infilling much of the song based on only a few notes from the user, participants found the amount of generated content overwhelming to unpack, evaluate, and edit. Specifically, they had difficulty determining why a composition was off, and expressed frustration at the inability to work on smaller, semantically meaningful parts of the composition. For example, one user struggled to identify which note was causing a discordant sound after multiple generated voices were added to their original: *"It was difficult because all the notes were put on the screen already... I can identify places where it doesn't sound very good, but it's actually hard to identify the specific note that is off."* Some participants naturally wanted to work on the composition *"bar-by-bar or part-by-part"*; in contrast to expectations, the generated output felt like it *"skipped a couple steps"* and made it difficult to follow all at once: *"Instead of giving me four parts of harmony, can it just harmonize one? I can't manage all four at once."*

2.3.2.2. Non-deterministic output. Even though the AI was capable of generating notes that were technically coherent to the context of surrounding notes provided by users, the stochastic nature of the system meant that its output did not always match the user's current musical objectives. For example, a participant who had manually created a dark, suspenseful motif was dismayed with how the generated notes were misaligned with the original feeling of the motif: *"the piece lost the essence of what I was going for. While it sounds like nice music to play at an upscale restaurant, the sense of climax is not there anymore."* Even though what was produced sounded harmonious to the user, they felt incapable of giving feedback about their goal in order to constrain the kinds of notes the model generated. Despite being *technically* aligned to context, the music was *musically* mis-aligned with user goals. As a result, participants wished there were ways to go beyond randomly *"rolling dice"* to generate a desired sound, and instead control the generation based on relevant musical objectives.

2.4. Cococo

Based on identified user needs, we developed Cococo (collaborative co-creation), a music editor web-interface for novice-AI co-creation that augments standard generative music interfaces with a set of *AI steering tools* (Figure 2.1). Cococo builds on top of Coconet [**74**], a deep generative model trained on 4 part harmony that accepts incomplete music as input and outputs complete music. Coconet works with music that can have 4 parts or *voices* playing at the same time (represented by **S**oprano **A**lto **T**enor **B**ass), are 2-measures long or 32 *timesteps* of sixteenth-note beats, and where each voice can take on any one of 46 *pitches*. Coconet is able to *infill* any section of music, including gaps in the middle or start of the piece. To mirror the most recent interfaces backed by these infill capabilities [**40, 59**], Cococo provides a rectangular *infill mask* feature, with which

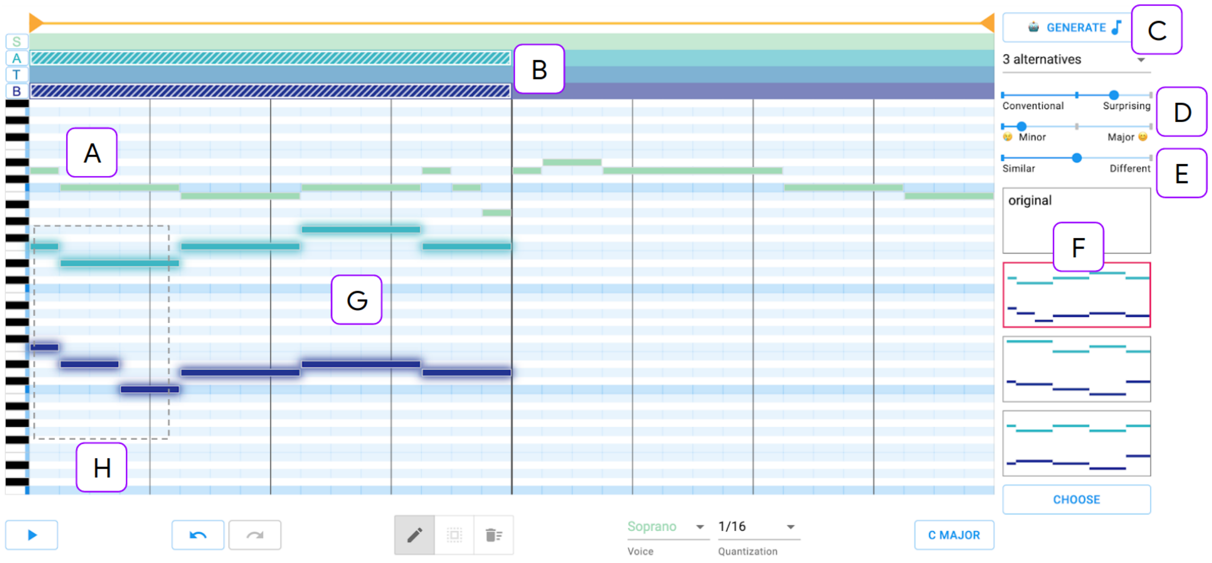


Figure 2.1. Key components of Cococo: users can manually write some notes (A), specify which voices and in which time range to request AI-generated music using Voice Lanes (B), click Generate (C) to ask the AI to fill in music given the existing notes on the page, use Semantic Sliders (D) to steer or adjust the AI's output along semantic dimensions of interest (e.g. more surprising, more minor or sad), use the Example-Based Slider (E) to express how similar/different the AI-generated notes should be to an example selection, or audition Multiple Alternatives (F) generated by the AI: users select a sample thumbnail to temporarily substitute it into the music score (shown as glowing notes in this figure (G)), then choose to keep it or go back to their original. Users can also use the Infill Mask's rectangular selection tool (H) to crop a section of notes to be infilled again using AI.

users can crop a passage of notes to be erased, and automatically infill that section using AI (see Figure 2.1H). Users can also manually draw and edit notes.

Beyond the infill mask, Cococo distinguishes itself with its *AI steering tools*. Specifically, users start an AI-generated iteration by using *Voice Lanes* to define for which time-steps (e.g. measure 1) and for which voices (e.g. soprano, alto, tenor, bass) notes can be generated. Desired musical qualities of the generated notes can be adjusted by using an *Example-based Slider* and *Semantic Sliders*. Finally, users have *Multiple Alternatives* to audition and choose from. Cococo supports an iterative co-creation process because users can repeat this workflow by inputting subsequent, incomplete versions of the composition to inform the AI's next generation. A visual description of this workflow is included in Figure 2.1.

2.4.1. Voice Lanes

Voice Lanes allows a user to specify the voice(s) for which to generate music within a given temporal range. With this capability, users can control the amount of generated content they would like to work with. This was designed to address information overload caused by Coconet's default capabilities to infill all remaining voices and sections. For example, a user can request the AI to add a single accompanying bass line to their melody by highlighting the bass (bottom) voice lane for the duration of the melody, prior to clicking the generate button (see Figure 2.1B). To support this type of request, we pass a custom generation mask to the Coconet model including only the user-selected voices and time-slices to be generated.

2.4.2. Semantic Sliders

Cococo includes two semantic sliders to influence what the generative DNN creates: a conventional vs. surprising slider, and a major (happy) vs. minor (sad) slider. This was based on formative observations that users wanted to control both musical qualities (e.g., how much the generated notes should stand out from what already exists) and emotional qualities (e.g., should the notes together produce happy or sad tones).

Users can make the generated notes more predictable given the current context by specifying more "conventional" on the slider, or more unusual by specifying more "surprising." The conventional/surprising slider adjusts the parameter more formally known as the *temperature* (T) of the sampling distribution [**55**]. A lower temperature makes the distribution more "peaky" and even more likely for notes to be sampled that had higher probabilities in the original distribution (conventional), while higher temperatures make the distribution less "peaky" and sampling more random (surprising). In formative testing, we found that a log scale interval of $[1/8, 2]$ with a midpoint of $1/2$ yielded a reasonable range of results. In addition, we refined the semantic labels of conventional/-surprising based on user feedback to best capture its behavior.

The major vs. minor slider allows users to direct the AI to generate note combinations with a happier (major) quality or a sadder (minor) quality. The limits of this slider include happy and sad face emojis to signal the emotional tones users can expect to control. To generate a passage that follows a more major or minor tone, we define a soft prior that encourages the sampling distribution to generate the most-likely major triad (for happy) or minor triad (for sad) at each time-step.

2.4.3. Audition Multiple Alternatives

Cococo provides affordances for auditioning multiple alternatives generated by the AI. This capability was designed based on formative feedback, in which users wanted a way to cycle through several generated suggestions to decide which was the most desirable. We allow the user to select the number of alternatives to be generated and displayed (with a default of three). A thumbnail preview of each alternative is displayed and can be selected for audition within the editor, allowing the user to hear it within the larger musical context. The musical chunk used as a prior to generation is accessible via the top thumbnail preview (labeled "original") so that users can always compare what the previous version of the piece sounded like, and opt to not use any of the generated alternatives.

2.4.4. Example-based Slider

While prototyping the Multiple Alternatives feature, we found that the non-determinism inherent in a deep generative model like Coconet can lead to two undesirable outcomes: generated samples can be too random and unfocused, or they can be too similar to each other and lack diversity. For example, when the generation area was small relative to the surrounding context, generated results would become repetitive: There were a limited set of likely notes for this context according to the model. As a solution, we developed the example-based slider for expressing that the AI-generated music should be more or less like an existing example of music. Before this slider is enabled, the user must select a reference example chunk of notes, either by using the most recent set of notes generated by AI, or manually selecting a reference pattern using the voice lanes or infill mask. Example-based sliders also use soft priors to guide music generation.

2.4.5. Soft Priors: a Technique for AI-Steering

Many of our AI-steering tools make use of a "soft prior" to modulate the model's generated output. These priors enable users or an AI-steering tool designer to add control to existing generative models without needing to retrain them. The model's sampling distribution

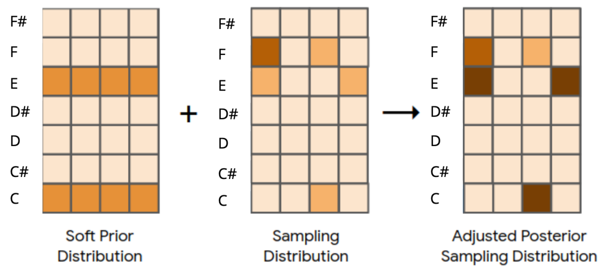


Figure 2.2. Use of soft priors to adjust a model's sampling distribution. Darker cells represent higher probabilities. The shape of the distribution is simplified to 1 voice, 7 pitches (rows), and 4 timesteps (columns). In Cococo, the actual shape is 4 voices, 46 pitches, and 32 timesteps.

is a softmax [**55**] probability distribution over all possible pitches, for each voice and for each time step; high probabilities are assigned to the pitches that are likely given the infill's surrounding musical context. The soft prior approach enables the generation of output that adheres to *both* the surrounding context (encoded in the model's sampling distribution) and additional desired qualities (encoded in a prior distribution). More formally, we use the equation below to alter the distribution used to generate outputs:

$$p_{\text{adjusted}}(x_{v,\,t}|x_C) \propto p_{\text{coconet}}(x_{v,\,t}|x_C)\, p_{\text{softprior}}(x_{v,\,t})$$

where $p_{\text{coconet}}(x_{v,t}|x_C)$ gives the sampling distribution over pitches for voice v at time t from Coconet given musical context x_C (C gives the set of v, t positions constituting the context), $p_{\text{softprior}}(x_{v,t})$ encodes the distribution over pitches specified by the user or AI-steering tool designer (serving as soft priors), and $p_{\text{adjusted}}(x_{v,t}|x_C)$ gives the resulting adjusted posterior sampling distribution over pitches.

The soft priors $p_{\text{softprior}}(x_{v,t})$ are defined so encouraged notes are given a higher probability, and those discouraged are given a lower, but non-zero probability. This setup allows for two desirable properties. First, since none of the note probabilities are forced to zero, very probable notes in the model's original sampling distribution can still be likely after incorporating the priors. Second, even though the priors are specified for particular voice and time steps, their effects can propagate to other parts of the piece. For example, as Coconet fills in the music, it will try to generate transitions that go smoothly between parts with a soft prior and parts without. Together, these make it possible for the model's output to adhere to both the original context and the additional user-desired qualities.

The soft priors technique powers Cococo's example-based slider and semantic sliders. When the user sets the example-based slider to more "similar," we create a soft prior that has higher probabilities for notes in the example. Conversely, for a slider setting of more "different," we create a soft prior that has lower probabilities for notes in the example. The soft prior is then used to alter the sampling distribution according to the equation and Figure 2.2.

The minor/major slider uses a slightly more complicated approach to define the soft prior distribution. To encourage notes from a major (or minor) triad, we construct the soft prior by asking what is the most likely major (or minor) triad at each time slice within the model's sampling distribution. The log-likelihood of a triad is computed by summing the log-probability of all the notes that could be part of the triad (e.g., for a C major triad, this includes all the Cs, Es, and Gs in all octaves). We repeat this procedure for all possible major (or minor) triads to determine which triad is the most likely for a time slice. We then repeat this procedure for all time slices to be generated, in order to create our soft prior for most likely major (or minor) triads; this soft prior is used to alter the sampling distribution to create the adjusted posterior sampling distribution as shown in Figure 2.2.

Cococo is implemented as a React.js web application[1], backed by an open source browser-based implementation [**118**] of the Coconet model. We modified Coconet to include soft priors.

2.5. User Study

We conducted a user study to evaluate the extent to which AI-steering tools support user needs, and to uncover how they affect the user experience of co-creating with AI. To this end, we compared the experiences of music novices using Cococo to that of a conventional interface that mirrors current interfaces for deep generative models (e.g. the Bach Doodle [**76**]). The conventional interface is aesthetically similar to Cococo, but does not contain the AI-steering tools. The conventional interface does include interactive control features via the *infill-mask* feature (present in both conditions) which enables users to crop any region of music to be regenerated [**40, 59**]. We ask in this study: **RQ1**: How do the AI-steering tools affect user perceptions of the creative process and the creative artifacts made with the AI (e.g., perceptions of ownership, self-efficacy, trust in the AI, quality of the composition, etc.) and **RQ2**: How do music novices apply the AI-steering tools in their creative process? What patterns of use and strategies arise?

[1]`https://github.com/pair-code/cococo`

2.5.1. Measures

To answer the research questions above, we evaluated the following outcome metrics. All items below were rated on a 7-point Likert scale (1=Strongly disagree, 7=Strongly agree, except where noted below).

Users' compositional experience is important to support for novice music creators pursuing autotelic, or intrinsically-rewarding, creative activities [**26**], which motivated the following set of metrics. **Creative expression**: Users rated *"I was able to express my creative goals in the composition made using [System X]."* **Self-efficacy**: Users answered two items from the Generalized Self-Efficacy scale [**122**] that were rephrased for music composition. **Effort**: Users answered the effort question of the NASA-TLX [**62**], where 1=very low and 7=very high. **Engaging**: Users rated *"Using [System X] felt engaging."* **Learning**: Users rated *"After using [System X], I learned more about music composition than I knew previously."* **Completeness**: Users rated *"The composition I created using [System X] feels complete (e.g., there's nothing to be further worked on)."* **Uniqueness**: Users rated *"The composition I created using System X feels unique."*

Motivated by the importance of supporting effective, human-centered partnerships with AI [**6, 26, 109**], we additionally evaluated users' attitudes towards the AI. **AI interaction issues**: Users rated the extent to which the system felt *comprehensible* and *controllable*, two key challenges of human-AI interaction raised in prior work on DNNs [**109**]. **Trust**: Participants rated the system along Mayer's dimensions of trust [**106**]: capability, benevolence, and integrity. **Ownership**: Users rated two questions, one on ownership (*"I felt the composition created was mine."*), and one on attribution (*"The music created using [System X] was 1=totally due to the system's contributions, 7=totally due to my* *contributions."*). **Collaboration**: Users rated *"I felt like I was collaborating with the system."*

2.5.2. Method

The 21 participants who completed the study included 12 females and 9 males, ages 20 to 52 ($\mu = 31$). To ensure that they were novices in composition, we required that they had played a musical instrument before at some point in their life, but had none or relatively little experience with composition and music theory. Almost all had either very little experience with music theory (12 users) or a beginner-level understanding of note reading, major/minor keys, intervals, triads, and time signatures (8 users). They had diverse prior experiences with music composition, where 6 had never considered composing, 8 had considered composing but never done it, and 7 had tried improvising or creating music informally. Users were recruited through mailing lists at our institution and came from a variety of professional backgrounds (e.g., designer, administrator, engineer). Each received a $40 gift credit for their time.

Each user first completed an online tutorial of the two interfaces on their own (30 minutes). Then, they composed two pieces, one with Cococo and one with the conventional interface, with the order of the conditions counterbalanced (15 minutes each). As a prompt, users were provided a set of images from the card game Dixit [**132**] and were asked to compose music that reflected the character and mood of one image of their choosing. This task is similar to image-based tasks used in prior music studies [**75**]. Users were observed while composing using a think-aloud procedure. Finally, they answered a post-study questionnaire and completed a semi-structured interview (20 minutes).

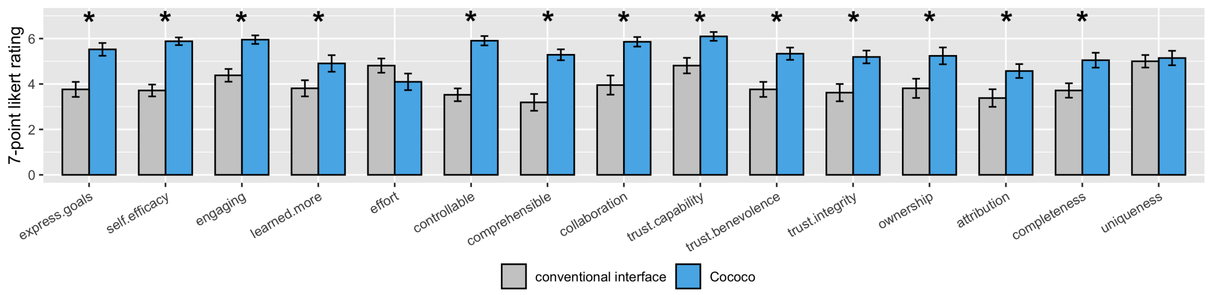


Figure 2.3. Results from post-study survey comparing the conventional interface and Cococo, with standard error bars.

To analyze the quantitative measures, we conducted paired t-tests using Benjamani-Hochberg correction [**13**] to account for the 15 planned comparisons (using a false discovery rate $Q = 0.05$). For qualitative findings, three authors conducted a thematic analysis [**19**] of the observation and interview data.

2.6. Quantitative Findings

Results from the post-study questionnaire are shown in Figure 2.3. In regards to users' perceptions of the creative process, we found Cococo significantly improved participants' ability to **express their creative goals** ($\mu = 5.5$, $\mu = 3.8$, $p = 0.0006$), **self-efficacy** (average of two items $\alpha = 0.86$, $\mu = 5.9$, $\mu = 3.7$, $p < 0.0001$), perception of **learning more** about music ($\mu = 4.9$, $\mu = 3.8$, $p = 0.0003$), and **engagement** ($\mu = 6.0$, $\mu = 4.4$, $p = 0.0001$) compared to the conventional interface. No significant difference was found in **effort** ($\mu = 4.1$, $\mu = 4.8$, $p = 0.1514$); participants described the two systems as requiring different kinds of effort: While Cococo required users to think and interact with the controls, the conventional interface's lack of controls made it effortful to express creative goals. Users' perceptions of the **completeness** of their composition made with Cococo was significantly higher than the conventional interface ($\mu = 5.0$, $\mu = 3.7$, $p =$

0.0116); however, no significant difference was found for **uniqueness** ($\mu = 5.1$, $\mu = 5.0$, $p = 0.6507$).

The comparisons for users' attitudes towards the AI were all found to be statistically significant: Cococo was more **controllable** ($\mu = 5.9$, $\mu = 3.5$, $p < 0.0001$), **comprehensible** ($\mu = 5.3$, $\mu = 3.2$, $p < 0.0001$), and **collaborative** than the conventional interface ($\mu = 5.9$, $\mu = 4.0$, $p = 0.0002$); participants using Cococo expressed higher **trust** in the AI, along the capability dimension ($\mu = 6.1$, $\mu = 4.8$, $p = 0.0008$), benevolence dimension ($\mu = 5.3$, $\mu = 3.8$, $p = 0.0004$), and integrity dimension ($\mu = 5.2$, $\mu = 3.6$, $p = 0.0055$). Users felt more **ownership** over the composition ($\mu = 5.2$, $\mu = 3.8$ $p = 0.0071$), and **attributed** the music to more of their own contributions relative to the AI ($\mu = 4.6$, $\mu = 3.4$, $p = 0.0136$).

2.7. Qualitative Findings

In this section, we describe participants' strategies for co-creating music, how they leveraged the AI-steering tools to work around perceived limitations of the AI, and how the steering tools helped novices "up-level" their existing skills and knowledge, while still retaining a sense of agency and ownership.

2.7.1. Tool-Based Strategies for Composing with AI

Users composed by breaking the task down into smaller, semantically-meaningful pieces, and used the steering tools to support initial brainstorming, to generate alternatives, and to steer the generation until it matched the user's creative intent.

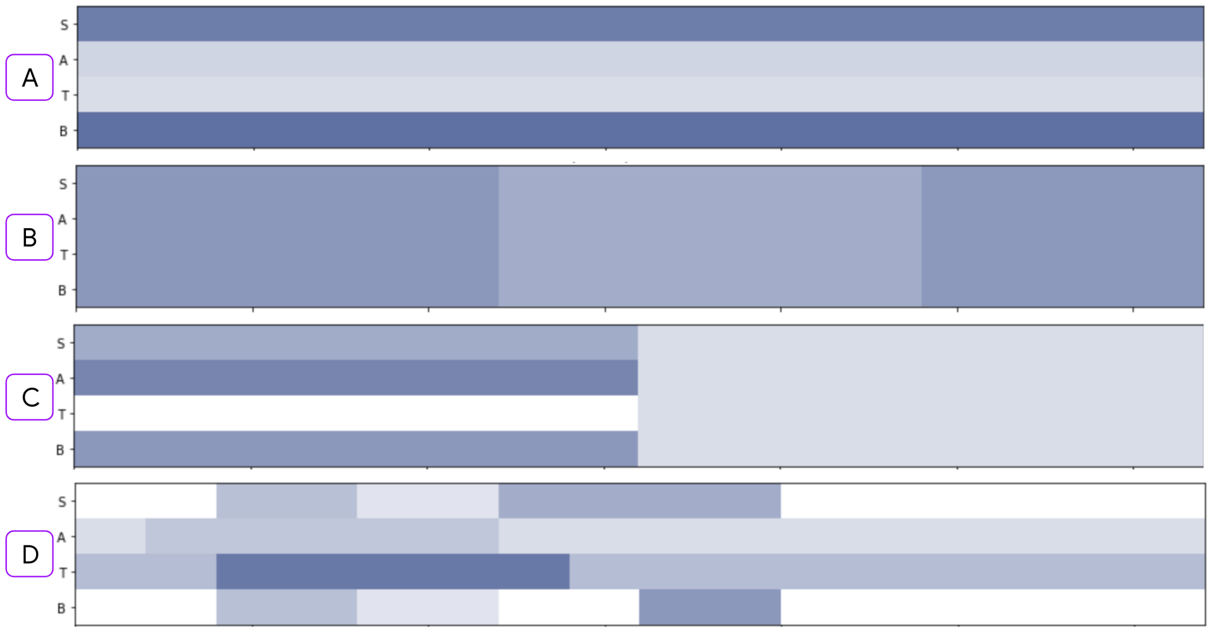


Figure 2.4. Common Patterns of using Voice Lanes, visualized using interaction data from 4 archetypal participants (darker-colored segments were performed by users before lighter-colored segments): (A) Voice-by-voice (most common), (B) Temporal Chunks, (C) Combination of Voice-by-Voice and Temporal Chunks, and (D) Ad-hoc Bits

2.7.1.1. Building Up, Bit-by-Bit. Many participants used the Voice Lanes to develop one voice at a time, in a "brick-building" fashion (Figure 2.4A): *"I'm trying to get the bass right, then the tenor right, then soprano and alto right, and build bit-by-bit"* (P2). This use of the Voice Lanes helped reduce the mental workload of handling multiple voices at once: *"As someone who cannot be thinking about all 4 voices at the same time, it's so helpful to generate one at a time"* (P2). Other participants leveraged the temporal aspect of the lanes (Figure 2.4B), using the AI to generate all four voices for a measure then refining the result. Some tried a combination of the voice-wise and temporal approaches, by working voice-wise in the first half of the song, then letting the AI continue a full measure in the second half (Figure 2.4C).

One participant referred to this piece-wise process as creating intermediate "checkpoints," where they stopped and evaluated the song before more content was generated. This strategy allowed participants to *"intervene after [the AI] generated [content]... stop it in the middle... and change it to feel different, before it kept going"* (P14).

In contrast, in the conventional interface, the AI fully auto-completed the music at once. As a result, participants resorted to "sculpting" and refining the AI's fully-generated music by repeatedly using the Infill Mask. Echoing the results in our need-finding study, some participants found the amount of resultant content overwhelming.

2.7.1.2. Working With Semantically Meaningful Chunks. Similar to composing bit-by-bit, users actively leveraged AI-steering tools to divide the music into semantically meaningful chunks, based on voice or time. For example, many used Voice Lanes to differentiate between the melody and background by using separate voices, or they assigned different musical personas to different voices. For example, one participant gave the tenor voice an *"alternating [pitch] pattern"* to express indecision in the main melody, then gave other voices *"mysterious... dinging sounds"* as a harmonic backdrop (P4).

Participants also divided the music into temporally distinct chunks as a way of illustrating evolution or change. One participant communicated a fight was about to start by requesting more conventional chords in the beginning third of the piece, then used the minor and surprising slider to generate an unresolved feeling in this evolving battle scene in the middle of the piece. In the final section, they used *"prolonged notes [to match] the long stare"* between dueling characters.

2.7.1.3. Generating, Auditioning, and Editing. Participants often employed the AI-steering tools to 1) point the AI in a desired initial direction, 2) audition the generated content, or 3) edit and steer the generated output. The Multiple Alternatives functionality naturally lent itself to this "generate and audition" strategy of music composition. Participants could generate a range of possibilities, audition them, and choose the one closest to their goal before continuing.

When generating content, the Semantic Sliders were sometimes used to set an initial trajectory for generated music: *"There's one... idea in my head.... that's the signal that I'm giving to the computer"* (P3). Some felt that this capability helped constrain the large space of possibilities that could be generated: *"Because I was able to give more inputs to [Cococo] about what my goals were, it was able to create some things that gave me a starting point"* (P8). In analysis of logs, 12 of the 21 participants modified the default values of the slider parameters prior to their first AI generation request.

AI-steering tools were also used to refine AI generated content, nudging in a direction closer to their intentions: *"It was... not dramatic enough. Moving the slider to more surprising, and more minor added more drama at the end"* (P5). Applying the example-based slider, users moved the setting to "similar" to push content closer to an example that embodied their musical goals: *"Work your magic on these notes, but keep it similar so they won't move around too much"* (P1). They set the slider to "different" when the initial AI-generated notes were *"not sounding good"* (P15) or when all the generated options needed to be *"totally scrapped"* (P13) because all were of opposite quality to the sound the user desired.

2.7.2. Tool-based Strategies for Addressing AI Limitations

In this section, we describe ways in which the steering tools were used to discover and directly address AI limitations.

2.7.2.1. Identifying and Debugging Problematic AI Output. By building up the music bit-by-bit, users became familiar with their own composition during the creation process, which enabled them to more quickly identify the *"cause"* of problematic areas later on. For example, one participant indicated that *"[because] I had built [each voice] independently and listened to them individually,"* this helped them *"understand what is coming from where"* (P7). Conversely, if multiple voices were generated simultaneously, participants found it difficult to understand the complex interactions: *"It's harder to disentangle what change caused what... when I make a change, there could be this mixed reaction...it propagates to [multiple] things at once"* (P6). By enabling users to generate bit-by-bit, and incrementally evaluate the music along the way, the steering tools may have enabled novices to better understand and subsequently "debug" their own musical creations.

2.7.2.2. Testing and Discovering the Limits of the AI. The steering tools also enabled participants to discover the limits of the AI. One participant, while using Voice Lanes to generate multiple alternatives for a single-voice harmony, discovered that the AI may be constrained by what's musically possible: *"Maybe the dissonance is happening because of how I had the soprano and bass... which are limiting it... so it's hard to find something that works"* (P15). Here, the Voice Lanes helped this user consider the limits imposed by a specific voice component, enabling them to reflect on the limits of the AI in a more semantically meaningful way. The Multiple Alternatives capability further enabled this participant to systematically infer that this particular setting was unlikely to produce better results through the observation of multiple poor results.

Some participants also set the sliders to their outer limits to test the boundaries of AI output. For example, one user moved a slider to the “similar” extreme, then incrementally backed it off to understand what to expect at various levels of the slider: *“On the far end of similar, I got four identical generations, and now I’m almost at the middle now, and it’s making such subtle adjustments”* (P18). These interactive adjustments allowed the user to quickly explore the limits of what they can expect the AI tools to generate, aiding construction of a mental model of the AI’s capabilities. In contrast, when using the conventional interface, users could not as easily discern whether undesirable outputs were due to AI limits, or a simple luck of the draw.

2.7.2.3. Proxy Controls. Participants drew upon a common set of composition strategies to achieve desired outcomes. For example, higher pitches were used to communicate a light mood, long notes to convey calmness or drawn-out emotions, and a shape of ascending pitches to communicate triumph and escalation.

Users who could not find an explicit way to express these concepts to the AI repurposed the steering tools as “proxy controls” to enact these strategies. For example, some users hoped that the surprising vs. conventional slider would be correlated with note density and tempo. A common pattern was to set the slider to “conventional” to generate music that was *“not super fast... not a strong musical intensity”* (P9), and to “surprising” for generating *“shorter notes... to add more interest”* (P15). Participants also turned to heuristics (such as knowledge that bass lines in music tend to contain lower pitches) to “reverse-engineer” which Voice Lanes to select in an attempt to control pitch range. Multiple tools were also combined to achieve a desired effect, such as using “conventional” in conjunction with the bass Voice Lane to create slow and steady music.

In some cases, even use of the AI-steering tools did not succeed in generating the desired quality. For example, the music produced using the “similar” setting was not always similar along the user-envisioned dimension, and the surprising slider did not systematically map to note density, despite being correlated. Facing these challenges, participants developed a strategy of “leading by example” by populating surrounding context with the type of content they desired from the AI. For instance, one participant manually drew an ascending pattern in the first half of the alto voice, in the hopes that the AI would continue the ascending pattern in the second half.

2.7.3. Novice Up-Leveling, Agency, and Collaboration

Beyond assisting with content generation and editing, the AI-steering tools seemed to help participants extend their music composition knowledge and skills.

2.7.3.1. Learning and Discovering Musical Structure. In the Cococo interface, there is no way to request initial music generation by the AI without first selecting Voice Lanes. As a result, the steering tools implicitly created a more structured workflow, which seemed to be helpful in providing scaffolding for novices: *“With all the controls, I feel more secure..... you have the bars of the [Voice Lanes]... you feel surrounded by this support of the machine”* (P13).

Users better understood how individual musical elements interacted together by re-purposing the steering tools to study isolated effects. For example, one participant described how a workflow of 1) manually composing a seed voice, 2) using the AI to generate a single accompanying voice from that seed, and 3) modifying the seed and repeating this process helped them *"more directly see how the changes [they] made affect things"* (P6). Another participant was *"curious what [Cococo] will put in for alto...[After the alto is generated] it seems to go with the soprano, but there's some dissonance near the beginning"* (P15). By isolating and revealing the effects of a single voice on another, the steering tools allowed participants to "micro-evaluate" the music and discover patterns in how components interact.

The steering tools also helped participants learn how sub-components affect semantic qualities. One user described how they came to understand *"that having that soprano up [at this bar]... gives a total injection of a different emotion,"* which they only realized by using the Voice Lanes to place a single voice within a single bar. Another user learned that *"a piece can become more vivid by adding both a minor and major chord"* after they applied the major/minor slider to generate two contrasting, side-by-side chunks (P12). Thus, while the conventional AI could do everything on its own, partitioning the AI's capabilities into smaller, semantically meaningful tools helped people learn composition strategies that they could re-use in the future.

2.7.3.2. Novice Self-Efficacy vis-a-vis the AI. Novices described how the steering tools instilled a sense of competence, self-efficacy, and agency when composing. For example, a participant contrasted the conventional interface, in which the *"machine is doing all the work,"* to Cococo, where they felt *"more useful as a composer"* (P3). The AI-steering tools also seemed to instill a sense of creative agency. By enabling participants to indicate what type of music was generated, the slider controls *"really help to express [myself] in a way [I] wouldn't be able to do in music notes or words"* (P7). Participants also attributed their sense of agency and ownership to the availability of choice, even if it wasn't exercised: *"There are options, but I don't feel like I have to use them... it's not like the [AI] is telling me 'This is the correct thing to do here'... so I felt I definitely had ownership in the music"* (P9). In contrast, participants indicated that they felt less ownership of the music in the conventional interface because they performed a smaller portion of the work, relative to the AI: *"The more I used the AI... the less I personally compose, the less ownership I felt....I was not as creative, I felt like I got lazier with the music...I relied on the AI to solve problems"* (P9).

While there were indications that the steering tools helped improve feelings of self-efficacy, there were also times when participants questioned their own musical capabilities when they were unable to obtain desirable results. Because the AI generates music given a surrounding "seed" context, users who were dissatisfied with AI output often wondered whether they had provided a low-quality seed, leading to suboptimal AI output: *"All the things it's generating sound sad, so it's probably me because of what I generated"* (P11). In such cases, participants seemed unable to disambiguate between AI failures and their own compositional flaws, and placed the blame on themselves.

In other instances, novices were hesitant to interfere with the AI music generation process. For instance, some assumed that the AI's global optimization would create better output than their own local control of sub-units: *"Instead of doing [the voice lanes] one by one, I thought that the AI would know how to combine all these three [voices] in a way that would sound good"* (P1). While editing content, others were worried that making local changes could interfere with the AI's global optimization and possibly *"mess the* *whole thing up"* (P3). In these cases, an incomplete mental model of how the system functions seemed to discourage experimentation and their sense of self-efficacy.

2.7.3.3. Novice Perceptions of AI's Collaborative Role. The ability to use AI-steering tools also affected how users perceived the AI as a collaborator. When using Cococo, users conceived of the AI as a collaborator that could not only inspire, but also revise and adjust to requests. For instance, one described it as a nimble team who *"could be adjusted to do what I would like for them to do... I had a creative team [if I needed one] or I had a conventional team [if I needed one]... like a large set of collaborators"* (P19). Others appreciated that Cococo was able to yield control to the end-user, and viewed the AI as more of a highly-proficient helper: *"An art assistant, who is extremely proficient, but has a clear understanding of who is in control of the situation"* (P18).

In contrast, participants called the conventional interface a *"brilliant composer"* (P16) they could outsource work to, but who was more difficult to communicate with. When working with the conventional interface, users were optimistic about its ability to surprise them with musical suggestions that they would not have thought of on their own but pessimistic about its *"blackbox"* (P19) persona when communicating and *"take-it-or-leave-it"* (P6) attitude when working together.

These differing views of the co-creation process with the two interfaces led to distinct ideas of where each interface would be most useful. For the conventional interface, participants imagined it to be useful when they feel *"lazy, and need to generate ideas quickly,"* (P2) or when they feel competent to compose most of a piece manually but are open to brilliant, unexpected suggestions. On the other hand, Cococo was useful when the user *"has some [creative goals] in mind that [they] want to build upon"* (P13).

2.8. Discussion

2.8.1. Partition AI Capabilities into Semantically-Meaningful Tools

Our results suggest that the Steering Interface Layer played a key role in breaking the co-creation task down into understandable chunks and generating, auditioning, and editing these smaller pieces until users arrived at a satisfactory result. Unexpectedly, novices quickly became familiar with their own creations through composing bit-by-bit, which later helped them debug problematic areas. Interacting through semantically meaningful tools also helped them learn more about music composition and effective strategies for achieving particular outcomes (e.g., the effect of a minor key in the composition). Ultimately, AI-steering tools affected participants' sense of artistic ownership and competence as amateur composers, through an improved ability to express creative intent.

2.8.2. Bridge Novice Primitives with Desired Creative Goals

Our study also revealed trade-offs between different levels of musical abstraction: whereas novices typically struggle to operate on the lowest level (raw notes), deep generative models operate at the highest level (generating entire compositions). Despite hopes that they can make music composition more approachable to lay novices [**76**], we found in our studies that this highest level of abstraction is surprisingly challenging for novices to work with. Instead, their interactions suggested utility in creating *mid-level* objects and concepts upon which they can manipulate.

Though we created an initial set of dimensions for AI-steering, we were surprised that participants already had a set of go-to primitives to express high-level creative goals, such as long notes to convey calmness or ascending notes to express triumph and escalation.

When the interactive dimensions did not explicitly map to these primitives, they re-purposed the existing tools as proxy controls to achieve the desired effect. Given this, one could imagine directly supporting these common go-to strategies. Given a wide range of possible semantic levers, and the technical challenges of exposing these dimensions in DNNs, model creators should at minimum prioritize exposing dimensions that are the most commonly relied upon. For music novices, we found that these included pitch, note density, shape, voice and temporal separation. Future systems could help boost the effectiveness of novice strategies by helping them bridge between their building blocks to high-level creative goals, such as automatically "upgrading" a series of plodding bass line notes to create a foreboding melody.

2.8.3. Preserving Creative Experimentation in the Age of Increasingly Powerful Generative AI

At the time this research was conducted, a generative AI model for music could not yet automatically produce music conditioned on emotions found in text or imagery. In the present day, however, new generative models for music have demonstrated the ability to generate music from text descriptions [**3, 79**]. Such text-based generative AI models open the possibilities to requesting a generative AI to create songs that span a variety of musical styles, moods and activities, and emotions. While it's possible to prompt music models to generate songs evoking multiple emotions (e.g., "piano piece: stuck, reflective, sad"), we might expect—by generalizing these findings—that directly using emotional prompts would take away a composer's engagement in the process of interpreting what an emotion might sound like and how to use musical concepts to evoke it. In this way, in order to preserve a creator's engagement in the process of directing fine-grained decisions in the creative process, user interfaces for language-based generative AI may need to be reconfigured to encourage the type of iterative steering we advanced in this chapter. One can imagine a more steerable language-based AI that encourages users to use mid-level constructs and describe semantically-meaningful chunks or layers they want to control. For example, such interfaces could expose knobs in the interface for controlling musical attributes, which could be easily translated into prompts for the model to follow (e.g., an sustained electric bass that plays arpeggiated notes; high-pitched bongos with ringing tones). Baking into the user interface these semantics about musical attributes and layers could provide novices affordances for learning what fundamental partitions they can have control over and what musical and semantic qualities they can begin to give them.

However, other aspects of novice user's interaction capabilities are not present in these current text-based music generation interfaces. The brick-building metaphor – which allowed people to listen to a part of the music, adjust the qualities of it before moving onward, is not supported. Since brick-building and incrementally generating pieces of the music supported important outcomes like better understanding of the music and feeling involved at different stages of the process, we might infer that these user outcomes are still undersupported by the default interfaces for text-based music models which still generate the entire song at once. Future work should explore what interactive capabilities or creative workflows do preserve and expand many of the meaningful processes and activities that make music creation joyful, creative, and challenging but rewarding.

2.9. Conclusion

We found that AI-steering tools not only enabled users to better express musical intent, but also had an important effect on users' creative ownership and self-efficacy vis-a-vis the AI. Future systems should expose mid-level building blocks, divulge the AI's capabilities and limitations, and empower the user to define the partnership balance. Taken together, this work advances the frontier of human-AI co-creation interfaces, leveraging AI to enrich, rather than replace, human creativity.

CHAPTER 3

An Interface Layer for Expressing High-Level Ideas using Low-Level Constructs

Another type of issue with trying to use AI capabilities is when the concepts a person wants to express are still many levels of abstraction removed from the constructs made available by an AI system. Such is the case for designers developing context-aware AI social technologies, where the high-level concepts of a situation—such as the socio-cultural meaning of why a similar situation could cultivate feelings of shared experiences—are semantically far from the available AI context detectors made at the level of location categories.

In this chapter, we introduce a human-AI interface layer for expressing high-level ideas using low-level constructs. This Expression Interface Layer reconfigures the user interface for communicating overarching intentions to AI systems by providing a visual workspace with cognitive support tools that helps people flesh out their high-level concepts and forage for relevant and precisely-operating AI constructs. In studies, we show that the Expression Interface Layer's cognitive bridging tools can help designers richly express their high-level ideas by helping them define a more expansive set of concepts for ways to realize it. Additionally, the cognitive bridging tools ensure that the AI constructs used operate as a designer intended. In this way, Expression Interface Layers give designers the tools to engage in an effective process for conveying their high-level, overarching ideas to AI systems, when the semantic gap between their ideas and AI's representations is large.

3.1. Introduction

Context-aware technologies have an enormous potential to be helpful and responsive within many situations that arise during people's daily lives. Increasingly, mobile context-awareness applications are being developed to help end-users think about places they are visiting, and what they can do there. For example, such applications can remind users to engage in personal activities or routines (e.g., buy vegetables, listen to live music) based on relevant places they encounter in their daily lives [**32**] or help users find coincidental moments to engage in shared experiences with other people across distributed contexts (e.g., when family members living apart can share a meal together) [**103**] The proliferation of context-aware applications have been possible due to the advances in better mobile sensors, location-based information sources (e.g., Foursquare, Yelp), and machine learning algorithms—which have made available a diverse set of component detectors that infer semantically-meaningful aspects of a user's context, which we refer to as *context-features* (e.g., whether a user is moving or stationary, whether they are visiting a park, whether their current weather is windy or not). By providing such semantically-meaningful context-features to program with, frameworks for building context-aware applications have made it easier for application designers to define how an application should trigger and act based on a user's current context.

Despite this focus on better component detectors and context-features, it is difficult to encode human concepts of a situation into a machine representation using available context-features. While context-features provided by mobile context frameworks are useful for detecting events or actions at the level of locations and place categories (e.g., a restaurant tagged with the place category ‘soup’; a ‘park’), it’s difficult to use such context-features to describe situations that would support experiences (e.g., ‘enjoying a warm meal on a cold day’; ‘good for tossing a frisbee’) when these concepts are several levels of abstraction removed from the context-features. While context-programming frameworks and trigger-action programming tools [**38, 127**] have made it easier for authors to access a variety of context-features, their processes for programming the situational detectors are limited to creating simple situation detectors at the level of the events and locations which can be detected. Instead, we argue that what is needed are programming environments that explicitly support the cognitive work required to flesh out an author’s concept for how a situation might enable experiences (e.g., ‘soup’ is one category of restaurants for a cold day, but where else?) and translate how available context-features apply to their concept. If designers could encode their human concepts of a situation—such as what contexts are appropriate for engaging in a personal activity, or what contexts support engaging in a digitally-mediated shared experience—it would improve the ability of applications to recognize human situations and facilitate appropriate activities within them.

To resolve these challenges, I developed an Expression Interface Layer that reconfigures how designers can use the existing AI features to express their high-level concepts of situations. This Expression Interface Layer is instantiated in a visual programming environment named Affinder. Using Affinder, a designer can take their ideas for a situation they want to use in a location-based, context-aware application and translate their human concepts of that situation into a logical expression using readily detectable context-features, which we call a *concept expression*. Applications can then use concept expressions to identify the situation across end-users' mobile contexts. For example, a designer of a mobile app that reminds users of opportunities to engage in personal activities or routines can construct the high-level concept "awesome for tossing a frisbee around" by using simpler concepts and detectors such as open recreational areas (disc golf, parks, playgrounds, beaches), weather is not disruptive (not windy), and while there is daylight (time between sunrise and sunset).

Affinder's Expression Interface Layer was designed to structure a construction process that allows people to flexibly switch between breaking down a higher-level concept into a construction that gets closer to the detectable features (top-down) and in defining more general concepts that link detectable context-features to human concepts (bottom-up). To support this, Affinder uses a block-based programming environment that provides a single, visual workspace for authors to (1) declare concept variables, or intermediate concepts that serve as links between an abstract concept and the context features; (2) forage for context-features that match their concepts; and (3) compose representations using logical operators; see Figure 3.1.

From our design-based research process creating Affinder, we also uncovered a set of **bridging challenges**, or specific obstacles that arise when trying to express concepts of a situation that cannot be directly specified with any single, detectable context-feature. First, concept expressions may be too narrowly defined, or *underscoped*, when a designer retrieves one context feature for a concept (e.g., 'parks' for 'grassy fields') but misses other context features (e.g., 'football' fields) that also match the concept. Second, the

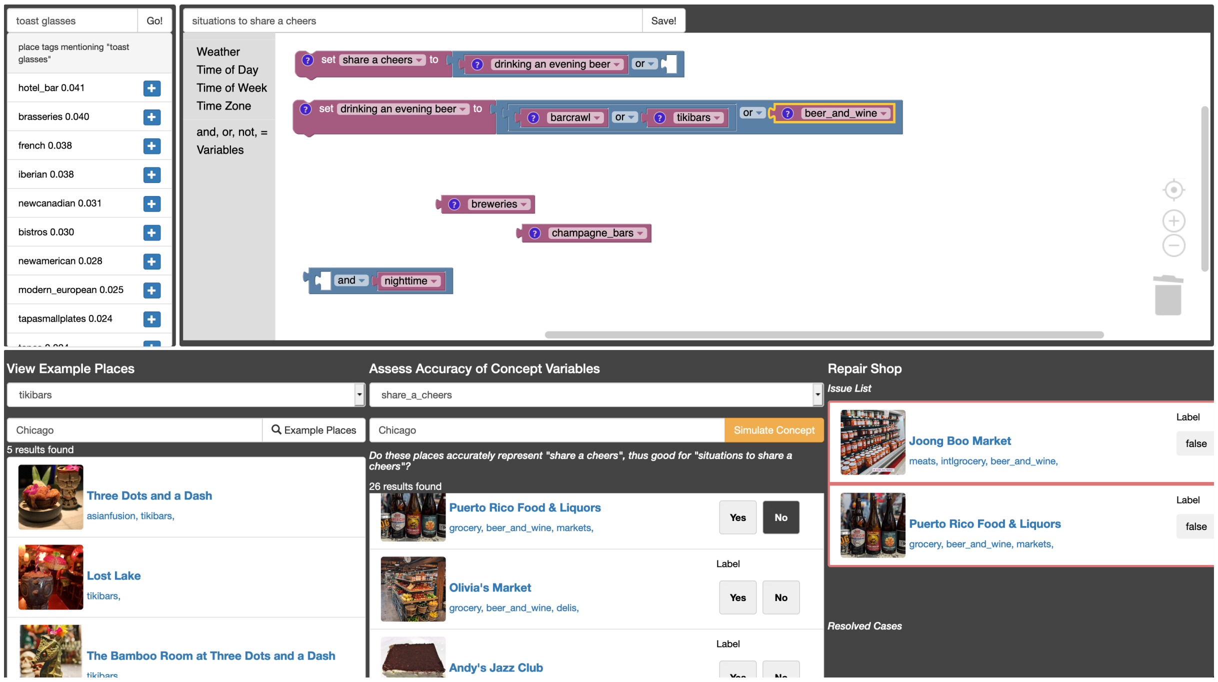


Figure 3.1. We built Affinder as an instantiation of a human-AI Expression Interface Layer. Affinder is a block-based programming environment for constructing concept expressions that effectively express a designer's concepts of a situation and the activities it affords (e.g., situations to share a cheers) to machines using available context-features. The visual workspace (top-right) supports declaring intermediate concepts that serve as links between an abstract concept and available context features; foraging for context features through browsing and searching hierarchies of features; and composing representations. As an Expression Interface Layer, Affinder implements three cognitive bridging tools: (1) an *unlimited vocabulary search* tool (top-left) helps designers discover available context-features relevant to their concepts; (2) *reflect and expand prompts* help designers generalize their notions of the concept they are trying to express and expand their foraging efforts; and (3) *simulation and repair tools* (bottom) help with identifying and resolving issues with the precision of concept expressions on real use-cases.

underscoping problem can also occur if a designer fixates on their early concepts of a situation (e.g., 'grassy fields' are good for frisbee tossing), which may result in a too narrow of an effort to forage for context-features that are relevant to the situation (e.g., in narrowly searching for context-features matching ‘grassy fields’, a designer can miss ‘beaches’ that are also good for frisbee tossing). Third, concept expressions may also include *detector inaccuracies* when a designer uses a context feature that does not evaluate the concept expression as expected (e.g., ‘parks’ may match all parks, including dog parks and skate parks that may be less desired as places for tossing a frisbee). To address these bridging challenges, Affinder contributes three core features that support designer cognition when expanding their concepts and translating them to machine features: (1) an *unlimited vocabulary search* for discovering context features they may have forgotten; (2) *reflect and expand prompts* that help designers generalize their notions of the concept they are trying to express and expand their efforts to forage for the context-features; and (3) *simulation and repair tools* for identifying and resolving issues with how machine detectors may operate on real use-cases differently than an author intends.

We conducted a between-subjects test comparing authors’ use of Affinder with its cognitive bridging tools vs. a baseline version of the block-based construction environment that only supported an opportunistic construction process. We find that the Expression Interface Layer’s cognitive bridging tools can help designers richly express a high-level situation by helping them define a more expansive set of concepts for ways to realize it (e.g., grassy fields and sandy areas are both open spaces that support throwing a frisbee). They expanded their own concepts while using the unlimited vocabulary search to forage for AI context-features, and while using the reflection and expand prompts to capture intermediate ideas about the more general reasons why a low-level construct they encountered was linked to their high-level idea. The cognitive bridging tools also helped designers to find relevant and accurate links between their intermediate concepts of a situation and the features that a context-aware AI system can understand and detect. Designers used simulation and repair tools to recognize and address issues with their concept expression operating differently on real-world cases than they expected.

In the rest of the chapter, we review related work to motivate challenges in using conventional interfaces for communicating high-level ideas of a situation to an intelligent, context-aware system. We describe the bridging challenges that arise during construction; and we detail the design of Affinder, a visual workspace equipped with cognitive bridging tools. We then report on the results of the lab study of Affinder, showing how the Expression Interface Layer's cognitive bridging tools aided designers in effectively bridging from their overarching concepts of situations to the available context-features. Finally, we end with a discussion on takeaways and future directions for developing Expression Interface Layers that support bridging between human concepts and machine representations.

3.2. Background

Context-aware systems are made up of two main components: (1) context providers, which use algorithms and inference techniques to extract attributes of a persons' context from sensors, which we refer to as machine detectable context-features; and (2) context-awareness services, which continuously reason about these attributes of context to perform useful actions on behalf of the user [**116**]. To develop use-cases for context-aware applications, authors must encode their concepts of a situation into a machine representation using context features made available by context providers. Since the created machine representations are composed of detectable context-features, context-aware services can then use these machine representations to act and facilitate interactions within desired situations.

Over the last two decades, research in context-aware computing and machine learning has significantly expanded the ability of context providers to infer aspects of human context across the dimensions of location, identity, activity, and time [**2**]. This work has led to numerous component detectors for various facets of context through better data, algorithms, and sensors (e.g., [**80**]). More recently, research systems have focused on using machine learning to model complex human situations within the domains of human activity recognition [**60, 93**], interruptibility and optimal work breaks [**88**], and mood-related mental health [**108**].

Today, many context features are widely available through mobile context providers (e.g., the Google Awareness API [**100**]) that implement a wide range of component detectors, including time, location, places, activity, and weather. Moreover, context-aware and trigger-action programming tools (e.g., iCAP [**38**], and IFTTT, or if-this-then-that [**127**]) have made it easier to program with context features. But while existing context provider APIs and programming tools provide application designers access to large sets of context features, they provide little support for expressing and encoding higher-level concepts of a situation that may be several levels of abstraction removed from these features. For example, while several trigger-action frameworks (e.g., multi-trigger variants of IFTTT) do support composing multiple context-features together into a single event, the situations expressed are assumed to be near the feature level, which allows rules to be built by accessing conceptual features directly [**127**].

When concepts of a situation are difficult for users to encode with the available features, programming by demonstration approaches have been successful in helping users map between a high-level situation and the available context features and sensors [**36**]. However, these approaches assume human teachers have an unchanged understanding about a concept and thus can easily provide positive and negative examples to a model to support. For our setting and task of expressing an author's concept of situations that support a desired activity, the application designer can suffer from design fixation [**9, 29**], which could result in the created machine representation being underscoped, or too narrowly defined, with respect to all the ways a situation could support a desired activity. Ultimately, unless we help the human expand their conception, whatever way they express it to machines (explicitly through construction, or implicitly through labels) will be limited to their current conception. For our proposed solution, Affinder provides specific cognitive scaffolds for the construction process, where an author declares their concepts and forages for context-features explicitly, which we argue naturally helps them flesh out and expand their own ideas in the process.

One particular challenge that arises in the construction process is identifying context features that support engaging in an experience or activity. Prior work by Dearman et al [**33**] attempted to do this by identifying potential activities supported at various locations by mining community-authored content (e.g., reviews). This approach extracts verb-noun pairs from community-authored content about a location in order to identify the potential activities supported by that location. While this work creates an extensive set of component detectors for potential activities supported by a location, it is limited to exact activities (e.g., drink soup) but can struggle to return results for higher-level situations of interest (e.g., places to enjoy warm food on a cold day). In other words, their approach may be particularly useful for finding context features matching specific (low-level) activities, but less useful for identifying features related to higher-level situations of interest. To address this problem, we introduce a more flexible unlimited vocabulary approach for finding features, and additionally introduce tools for representing and acting on relevant concepts across levels.

3.3. The Bridging Problem

In this section, we introduce the idea of the bridging problem: the difficulty in encoding human concepts of how a situation supports desired experiences into a machine representation using available context-features. For example, an author may want to design a situational trigger that identifies everyday opportunities for users to perform a playful activity like throwing a frisbee. Starting with their human concepts, an author wants to include other place contexts like 'parks' good for tossing a frisbee, and also recognizes that most 'open fields ' would be generally appropriate for this activity. Now, the author needs to figure out how to link between their concepts of how a situation enables tossing a frisbee and the detectable context-features.

A key technical challenge is figuring out an effective *construction* process for bridging from human (mental) representations of a conceptually-rich human situation to a machine representation built using available context features. In one direction, a top-down process can support a creator decomposing a situation into simpler concepts, but can frequently lead to scenarios where a creator finds that there are no matching context features for such concepts (e.g., no detector for 'open space'). In earlier systems such as iCAP [**38**], such concepts are simply ignored and left out of the construction. In another direction, bottom-up approaches allow for reusing and composing context features [**121, 127**], but developers can become stuck when they do not know a priori what context features may be useful for expressing an abstract idea that they have. To overcome these challenges, we recognized that authors need a more fluid construction process that allows them to choose a top-down or bottom-up strategy in order to find a link between their concepts of the situation and the available context-features.

In addition to enabling a more fluid construction process, we focused our design efforts on uncovering and addressing some of the specific challenges that arise in this construction process. We used a design-based research method in which we iteratively prototyped and tested with participants across two rounds of pilot tests (N = 7, N = 6) to understand how Affinder supports an author's process in constructing high-level situations from available context-features, and any remaining obstacles that arose. We tasked participants with expressing high-level situations that identify opportunities to have shared experiences in a context-aware social application use-case (e.g., situations to share a cheers; awesome situations to throw a paper airplane; situations to watch the sunset over water).

Through this phase of iterative prototyping and pilot testing, we identified three general bridging challenges authors face when there is a mismatch between their human (conceptual) representations and the machine's available representations. Across these challenges, we found that they can be caused by cognition difficulties in fleshing out concepts for how a situation supports potential activities, and in translating these concepts into available machine detectable features.

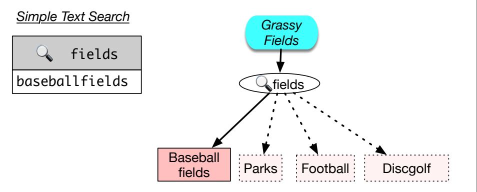
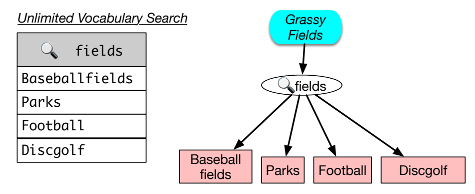
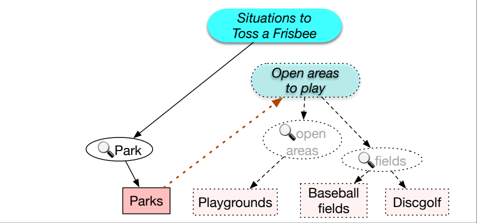
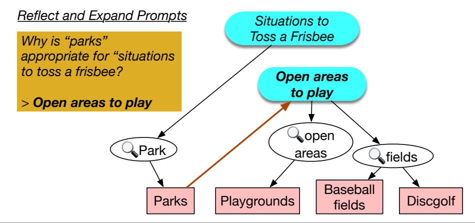
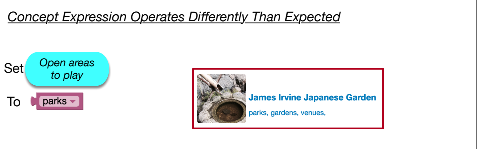
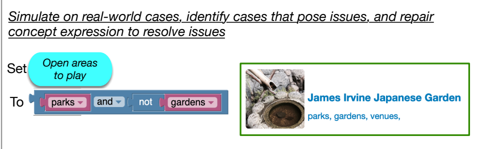
Construction Challenge	*Example of Construction Challenge*	*Affinder's Technical Solution*
Underscoped on Features Designer retrieves some context-features, but misses others that also match the concept	 Simple Text Search fields baseballfields Grassy Fields fields Baseball fields Parks Football Discgolf	 Unlimited Vocabulary Search fields Baseballfields Parks Football Discgolf Grassy Fields fields Baseball fields Parks Football Discgolf
Underscoped on Concepts Designer has narrow notions of the concept they are trying to express, resulting in a limited effort to forage for relevant context-features.	 Situations to Toss a Frisbee Open areas to play Park open areas fields Parks Playgrounds Baseball fields Discgolf	 Reflect and Expand Prompts Why is "parks" appropriate for "situations to toss a frisbee? > Open areas to play Situations to Toss a Frisbee Open areas to play Park open areas fields Parks Playgrounds Baseball fields Discgolf
Concept Expression Inaccuracies Designer may not realize that context-features will operate differently than they intended.	 Concept Expression Operates Differently Than Expected Set Open areas to play To parks James Irvine Japanese Garden parks, gardens, venues,	 Simulate on real-world cases, identify cases that pose issues, and repair concept expression to resolve issues Set Open areas to play To parks and not gardens James Irvine Japanese Garden parks, gardens, venues,

Figure 3.2. We identify three construction challenges, and implement their corresponding solutions in Affinder: (A) concept expressions can be underscoped if designers fail to retrieve relevant features for a concept; (B) concept expressions can be underscoped if *concept variables* are too narrowly defined which limits efforts to forage for features; and (C) concept expressions can have inaccuracies, when it executes not as a designer intends.

3.3.1. Underscoped on Features

Concept expression may be too narrowly defined, or *underscoped*, when a designer recalls one context feature for a concept ('parks' for open space) but forgets other context features (e.g., 'beaches') that also match the concept. The underscoping problem can occur when trying to translate from their concepts to the available features: when foraging for relevant context-features from a long list of features, designers may fail to find a comprehensive set of context-features matching their concept for a situation; see top-row of Figure 3.2.

We noticed in early piloting that the list of place category detectors provided by the Yelp Places API [**82**] was extensive and thousands of items long, making it impractical for designers to comprehensively scan through the hierarchy of items. Designers could use simple text search on this list, but they may not know all the names of the place categories that Yelp defines a priori to know which ones will be useful. For example, a pilot participant thought that “fields” would be a type of place listed on Yelp, and with simple text search, this query returned “baseballfields”. However, there are many other place categories (e.g., parks, football, stadiums, discgolf) which may contain the open grassy field for frisbee tossing they conceptualized. Evidently, different designers will have their own notions and vocabulary of a concept which makes it difficult to find relevant machine features [**52**]. Thus, our first round of iterative development and piloting aimed to support designers in finding the broader set of place context-features based on their notions of a high-level activity and situation.

3.3.2. Underscoped on Concepts

The underscoping problem can also occur when designers fixate on a narrow set of intermediate concepts to describe their overall concept for a situation. Studies of designer’s cognition and metacognition highlight that designers have a tendency to fixate their search for solutions, with a cognitive bias towards their earliest solution ideas [**9, 29**]. Much like the literature suggests, we saw this with participant designers in our pilots. For instance, the middle-row of Figure 3.2 illustrates how one designer from our pilots narrowly conceptualized the idea of “situations to throw a frisbee”. They started by recalling “parks” as a common place they associate with being able to perform the activity. However, after using this search term to find a few place context-features that afforded the activity (e.g., park, dogpark), the participant stopped their search for other potentially relevant place category context features.

What this participant didn't conceptualize was a more generalized notion of what makes a place good for tossing a frisbee (e.g., a place must have "open areas to play"). Had they had a broader notion, they could have continued to forage for context-features in other part of conceptual space (e.g., "open area", "fields"), and found other context-features relevant to detecting situations to toss a frisbee (e.g., playgrounds, baseball fields, discgolf). Thus, our second phase of pilot development focused on helping designers push past their earliest concepts for a high-level situation, and encourage them to expand their concepts and associated context-features for a situation.

3.3.3. Concept Expression Inaccuracies

Creating a concept expression that operates accurately in real-world scenarios requires iterative refinement. Concept expressions can operate inaccurately when a context-feature does not operate as a designer intends. For example, they may not precisely match a concept (e.g., public gardens are parks but are not good for frisbee tossing); see bottom-row of Figure 3.2.

In early prototyping, we observed that users' mental-model about a context-feature, based upon only its name, can often mismatch how a context-feature actually applies to real-world cases. This led to participants adding context-features which inaccurately matched the concept expression when applied to real-world place venues (e.g., 'recreation' refers to indoor recreation centers and gyms which are not appropriate for frisbee tossings).

As a low-fidelity solution for this challenge, we allowed authors to reference the Yelp website to view example locations that were listed for any place category they were uncertain about. Eventually, we incorporated this as an integrated interface in Affinder, where users could view several example location venues listed for a place category context-feature.

Additionally, a more subtle way concept expressions can inaccurately match was when several real-world cases challenged a designer's primary mental model of a context-feature. For example, a user was originally thinking of "parks" as a category for throwing paper airplanes, but recognized through foraging for other place category context-features that some "parks", such as "dog parks" would more likely lead to "a dog chewing up the airplane." Thus in our second round of pilot prototyping, we focused on developing tools integrated within Affinder's construction environment to help designers simulate their concept expressions on real-world cases so as to surface when concept expressions and their context-features operate differently than intended.

3.4. Affinder: a Human-AI Interface Layer for Expression

In this section, we introduce a human-AI interface layer for expressing high-level ideas using low-level constructs. An Expression Interface Layer reconfigures the user interface for communicating overarching intentions to AI systems by providing a visual workspace with cognitive bridging tools that helps people flesh out their high-level concepts, forage for relevant and precisely-operating AI low-level building blocks, and finally construct representations that an AI system can use to detect and act on.

We instantiate this Expression Interface Layer into Affinder, a programming environment for constructing *concept expressions* that effectively translates an author's human concept of a situation into a machine representation using context-features that can be acted upon computationally. Specifically, Affinder allows a designer to take a situation they wish to use in a location-based or context-aware application (e.g., where to go to share a warm meal on a cold day), and to translate it— through the process of construction— into a logical expression based on available context features that can be readily detected and used with an application (e.g., restaurants serving soup OR restaurants serving spicy food).

To support this, Affinder provides (1) a *block-based programming environment* that facilitates an opportunistic construction process designed to overcome challenges with strictly top-down or bottom-up strategies. To address the three bridging challenges that arise during the construction process, Affinder also provides cognitive bridging tools consisting of (2) an *unlimited vocabulary search* for discovering context features one may have forgotten; (3) *reflect and expand prompts* that help generalize one's concepts of the situation and expand one's efforts to forage for context-features; and (4) *simulation and repair tools* for identifying and resolving issues with how context features may operate on real use-cases differently than an author intends. We describe each of these functions below.

3.4.1. Block-Based Construction Environment

To overcome the shortcomings of top-down and bottom-up approaches, we draw on theories from opportunistic planning [**64**] to support an *opportunistic* construction process, in which a creator can follow both top-down or bottom-up processes at any time. Decisions and observations during construction may suggest new ideas or illuminate problems that cause the creator to shift their strategy. To support this construction process, Affinder

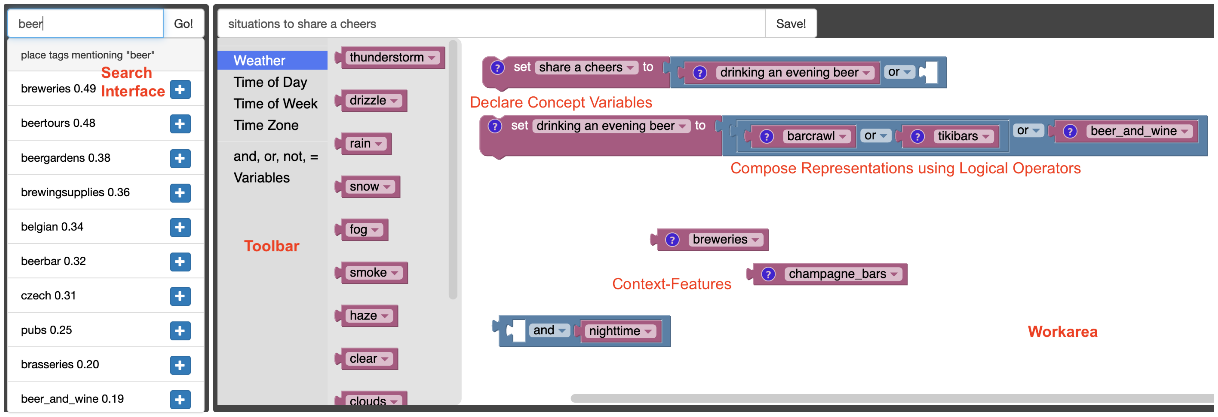


Figure 3.3. Affinder's *Toolbar* provides access to building blocks such as context features derived from weather, time, place in addition to logical operators like and, or, not, =. From the toolbar, users can drag and drop building blocks into the *Work Area* which is used to store relevant features and compose representations from building blocks. Affinder's *Search Interface* returns relevant place context-features. A button next to each 'adds' the feature to the work area.

employs a block-based programming environment that provides a single, visual workspace in which designers can (a) declare *concept variables* to represent intermediate concepts that serve as links between an abstract concept and available context features; (b) forage for context features by browsing and searching through categories of features; and (c) compose representations using logical operators; see Figure 3.3. We argue that this block-based approach can effectively support an opportunistic construction process by visually linking concept variables to context features; supporting recognition over recall; and reducing cognitive load by helping developers focus on concepts and how they are connected instead of on syntax and code.

Declaring concepts entails breaking down an idea about a situation into smaller concepts. These smaller concepts serve as a link between a conceptually-abstract situation and the available context features, and can be declared explicitly with *concept variables*.

A creator can declare concept variables before defining their contents; this provides a visual reminder to compose building blocks later for the concept. For example, for a construction expressing 'situations to share a cheers', a user can represent a smaller concept 'drinking an evening beer' by declaring a concept variable, defining its contents using context-features and logical operators, and using the defined concept variable in a top-level concept variable named 'share a cheers'.

Foraging for features involves both searching and browsing for building blocks. Creators can either go in a top-down fashion by using their declared concepts to guide the types of building blocks they might look for, or a bottom-up fashion where they browse through the available features to see which ones might be relevant. A creator can use the *Toolbar* to navigate to building blocks based on categories such as weather, time of day, time of week, and time zone (Figure 3.3, Left of Middle). Creators can also use the *Search Interface* (Figure 3.3, Far Left) to find context-features based on a thousand place categories on Yelp. For example, a user expressing 'having an evening beer' could browse for features based on time of day and find a 'nighttime' block to drag into the work area; they might query the search interface for 'beer' to find relevant place context-features to add to the construction.

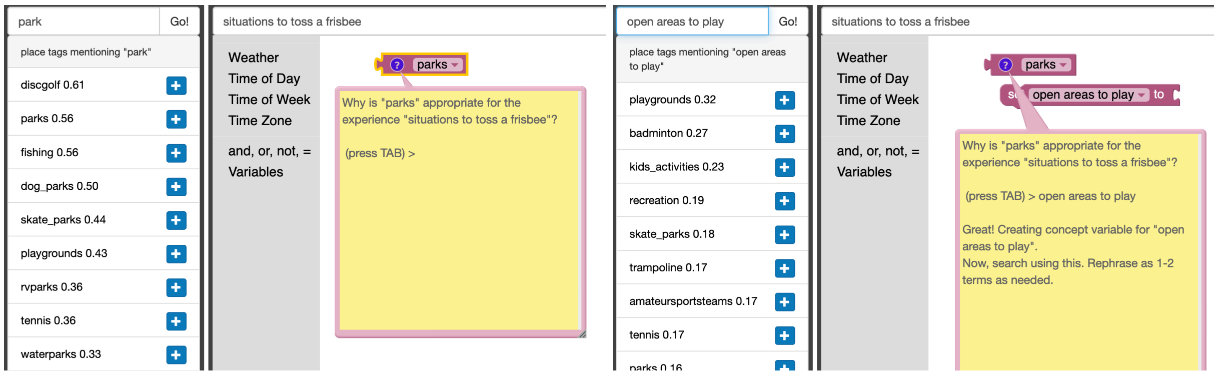
Composing Representations starts after creators find several context building-blocks which they combine together with logical operator blocks like *and, or, not.* Figure 3.3 shows an example of a logical composition used in the definition of 'drinking an evening beer' as various place contexts for having a beer ('barcrawl', 'tikibars', 'beer and wine').

3.4.2. Unlimited Vocabulary Search

Concept expressions can be underscoped in regards to the context-features a creator decides to include. This can occur when working with lengthy hierarchies of context-features (e.g., Yelp place categories), since designers can struggle to forage for features that can represent their intermediate concepts. To mitigate underscoping on detectable context-features, Affinder uses textual metadata from available APIs (e.g., reviews from Yelp) to create an unlimited vocabulary [**52**] of terms associated with the context features that users can query for based on their conception. This allows a creator to directly query for context features using aspects of a situation of interest, e.g., based on objects that afford actions, actions that can be taken, activities that people are engaged with, etc. This helps developers discover context features they may have forgotten, and to broadly shift their notions of the concept they are trying to express and how to express it throughout the process of construction. For example, using Affinder's unlimited vocabulary search, the query 'field' could match the feature 'parks' through a review that says "Other aspects of the park include a people park with slides and swings, soccer *fields*, baseball *fields*, and plenty of open space;" or 'fields' could match the feature 'discgolf' through a review that says "The majority of the course is quite open and flat, playing around some decent sized trees, grassy *fields* and pedestrian pathways that are considered out of bounds."

3.4.3. Reflect and Expand Prompts

To mitigate underscoped concept expressions caused by design fixation on intermediate experience concepts, Affinder provides prompts that encourage users to *reflect* on generalizable concepts about why a context-feature is appropriate for the situated experience



place tags mentioning "park"

discgolf 0.61
parks 0.56
fishing 0.56
dog_parks 0.50
skate_parks 0.44
playgrounds 0.43
rvparks 0.36
tennis 0.36
waterparks 0.33

Why is "parks" appropriate for the experience "situations to toss a frisbee"?

(press TAB) >

place tags mentioning "open areas to play"

playgrounds 0.32
badminton 0.27
kids_activities 0.23
recreation 0.19
skate_parks 0.18
trampoline 0.17
amateursportsteams 0.17
tennis 0.17
parks 0.16

Why is "parks" appropriate for the experience "situations to toss a frisbee"?

(press TAB) > open areas to play

Great! Creating concept variable for "open areas to play".
Now, search using this. Rephrase as 1-2 terms as needed.

Figure 3.4. A series of two screenshots illustrates how a user might use the reflect and expand prompts to create new generalizable concept variables, and expand their efforts to forage for features.

they are designing, and to *expand* their concept expression by foraging for context-features using this generalizable concept.

Each context-feature or concept variable can be used as the source of the reflection; reflection prompts can be activated for any of these by clicking on the corresponding blue question mark; see Figure 3.4. For example, on the left-side of Figure 3.4, a user has added the context-feature ‘parks’ in their work area, and chooses to reflect by pressing the ‘?’ button associated with the ‘parks’ context-feature. The reflection prompt asks them the question *“Why is ‘parks’ appropriate for the experience ‘situations for tossing a frisbee’?”*. On the right-side of Figure 3.4, the user has proceeded to answer by typing ‘open areas to play’. Upon pressing the tab key, a new concept variable is created that represents this generalized notion. They can subsequently use the search interface to find context-features matching the concept ‘open areas to play’. We designed the ‘?’ reflect button to be always visible and attached to the context-features in the workspace after observing during early testing that authors would sometimes forget that the reflect button was a feature they could use.

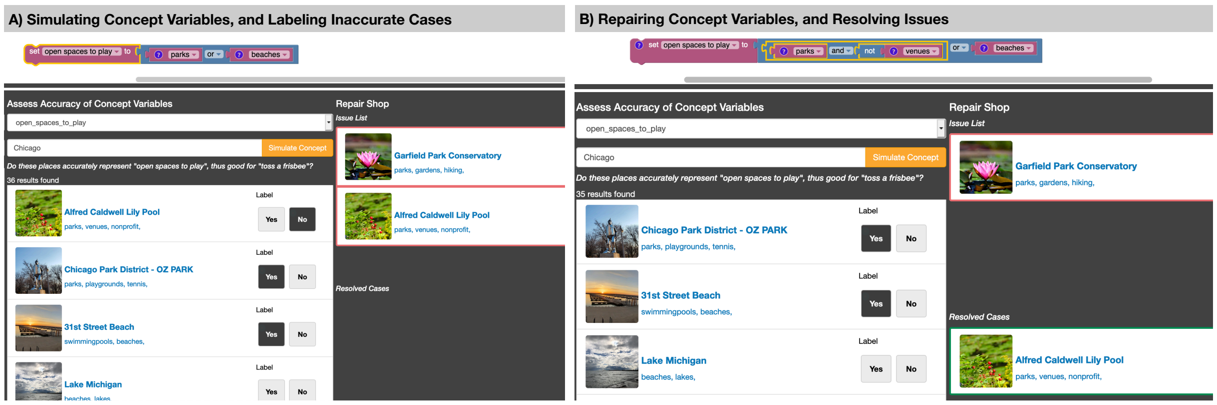

A) Simulating Concept Variables, and Labeling Inaccurate Cases
set open spaces to play to parks or beaches
Assess Accuracy of Concept Variables
open_spaces_to_play
Chicago
Simulate Concept
Do these places accurately represent "open spaces to play", thus good for "toss a frisbee"?
36 results found
Alfred Caldwell Lily Pool
parks, venues, nonprofit,
Label Yes No
Chicago Park District - OZ PARK
parks, playgrounds, tennis,
31st Street Beach
swimmingpools, beaches,
Lake Michigan
beaches, lakes,
Repair Shop
Issue List
Garfield Park Conservatory
parks, gardens, hiking,
Alfred Caldwell Lily Pool
parks, venues, nonprofit,
Resolved Cases
B) Repairing Concept Variables, and Resolving Issues
set open spaces to play to parks and not venues or beaches
Assess Accuracy of Concept Variables
open_spaces_to_play
Chicago
Simulate Concept
Do these places accurately represent "open spaces to play", thus good for "toss a frisbee"?
35 results found
Chicago Park District - OZ PARK
parks, playgrounds, tennis,
Label Yes No
31st Street Beach
swimmingpools, beaches,
Lake Michigan
beaches, lakes,
Repair Shop
Issue List
Garfield Park Conservatory
parks, gardens, hiking,
Resolved Cases
Alfred Caldwell Lily Pool
parks, venues, nonprofit,

Figure 3.5. Designers can simulate concept expressions composed of multiple context-features (e.g., 'parks' or 'beaches' are 'open spaces to play'). Simulation can reveal cases in which context-features may fail to accurately represent specific concept variables (e.g., some 'parks' include locations like a conservatory and a lily pool which are not 'open spaces to play'). After labeling these inaccurate cases, designers can repair concept expressions to make better use of existing context features (e.g., parks only if they are also not 'gardens' or 'venues'). Simulating the repaired concept moves the offending case to a list of resolved cases.

.

3.4.4. Simulation and Repair Tools

Given the challenge of knowing how a place context-feature applies to real-world cases based only its name, Affinder implements a feature for viewing example locations for a context-feature of interest (e.g., a list of the top 20 example locations tagged with the context-feature 'active' in Chicago), so designers can form a more accurate mental model of how a place category is used by the Yelp context-provider. This can help answer questions about context-features which are named in unexpected ways, such as what types of places the context-feature 'active' refers to.

Beyond viewing example locations for *individual* context-features, the full version of Affinder provides features for (A) simulating the execution of concept expressions composed of *multiple* context-features to help designers analyze real-world cases and label any that inaccurately represents the concept variable (as shown in Figure 3.5, Left); and (B) supporting designers to repair their concept expressions, so as to resolve outstanding issues in the execution of the concept variable (as shown in Figure 3.5, Right).

Affinder's Repair Tools supports two methods for repairing a concept expression so they operate more accurately: (1) using logical operators to exclude specific context-features; and (2) discarding features that are too inaccurate to be useful. First, context-features may not precisely match a concept (e.g., public gardens are parks but are not good for frisbee tossing). To resolve this problem, the first method of repair supports designers in excluding specific context-features (e.g., define 'open spaces to play' as 'parks' that are also not 'gardens'). By doing this repair, a designer can still effectively add a context-feature to their construction to increase coverage (e.g., most parks are good for frisbee tossing), while ensuring that this addition does not sacrifice precision (e.g., so that special parks that don't fit 'open grassy area for play' concept would not be identified for the situation or activity). To help designers quickly identify cases that would require this type of repair, the list of detected cases is ordered by cases that are tagged with multiple context-features (a conservatory tagged 'parks', 'gardens', and 'hiking') appear before cases tagged with a single context-feature (a beach tagged 'beaches').

Second, a context-feature may be too inaccurate to be useful (e.g., 'recreation' mostly refers to indoor recreation centers and gyms which are not appropriate for frisbee tossing). Thus, the second method of repair allows a designer to discard this context-feature from the construction. By recognizing when a context-feature is too inaccurate and making the choice to discard it entirely, designers are able to assess when a context-feature they've added to their construction would inaccurately represent the high-level concept they were trying to express, and ultimately degrade the precision of their expression.

3.5. Implementation

Affinder is a Meteor.js web application[1] that uses Google's Blockly library [**50**] for its block-based construction interface. Affinder uses Blockly to generate a concept expression's corresponding Javascript code (e.g., (parks || beaches) && !windy;) which is a logical predicate that can be used by context-aware services to check whether the user is in a situation that matches the concept expression, or to produce a list of situations and locations that would. Applications can integrate this generated code by requesting a user's current context-features (e.g., Is the Yelp place category 'park' currently detected for a user? Is the user's current detected weather 'windy'?) from a context-provider API and evaluating the predicate. Affinder's simulate and repair tool uses the generated code to simulate the concept expression on real-world place venues.

Affinder's unlimited vocabulary search engine was built by applying the term-frequency inverse-document frequency (TF-IDF) statistic to a corpus of 1241 documents corresponding to Yelp place category context-features. Each document comprised community-authored reviews for a single place category (e.g., 'parks') across all listed places in 8 major metropolitan areas [**81**]. The reviews associated with a place venue which is tagged with multiple place categories (e.g., a public park tagged as 'park', and 'playground') will be

[1]`https://github.com/NUDelta/affinder`

included in the text documents associated with all place categories. During iterative prototyping and testing, we observed that this also had the desirable side-effect of broadening the set of returned place category context-features. Document relevance for a query was based on the sum of TF-IDF values for all terms in a query [**115**], and the 25 top place categories are returned.

Affinder's feature for simulating and repairing concept expressions uses the Yelp Fusion Business API [**82**] to return a list of real-world places for a given city. Our implementation of simulating a concept expression creates a list of locations by taking the set union of the top 20 locations for each of the context-features in an expression (e.g., a concept expression open spaces = parks || beaches has two context-features and will return 40 real-world locations for a target city). Then, the set of positive predictions is obtained by applying the concept expression's corresponding Javascript code to the list of location venues.

3.6. User Study

We performed a comparison study to evaluate the extent to which Affinder supports creators in effectively encoding their human concepts of a situation into a machine representation using available context-features. Specifically, we conducted a between-subjects study that compared authors' use of the full version of Affinder with all the features for overcoming the bridging challenges vs. a baseline version of Affinder with a reduced set of features.

In this study we ask: **RQ1**: Does unlimited vocabulary search help designers find relevant context-features, to overcome the challenge of a concept expression being underscoped? **RQ2**: Do reflect and expand prompts help designers stretch their concepts and efforts to forage for context features? **RQ3**: Do simulation and repair tools help designer's recognize cases when a concept expression does not operate as intended on real-world cases, to overcome concept expression inaccuracies?

3.6.1. Method and Analysis

3.6.1.1. Experimental vs. Baseline Versions. For this between-subjects study, we provided participants two versions of Affinder: an experimental version with all the core features for overcoming the specific bridging challenges (unlimited vocabulary, reflect and expand prompts, simulation and repair tools), and a baseline version without these features. Both versions support an opportunistic construction process through its block-based environment where an author can follow a top-down or bottom-up process at any time; and both versions use the same set of base context-features including Yelp place categories, time, and weather features.

However, the baseline does not include the core features that address specific challenges arising within the bridging problem.

(1) Instead of the unlimited vocabulary search, authors forage for place category context-features using a simple text search (e.g., searching 'park' will match context-features 'parks' and 'parking lots', whereas searching 'frisbee' returns

no place categories). We hypothesize that without unlimited vocabulary, authors using the baseline will struggle to access relevant features based on their conceptions because they will not know the exact names of the place categories.

(2) Authors using the baseline do not have explicit prompts for reflecting and expanding concept expressions. Without these prompts, we hypothesize that users may fixate on their early concepts of a situation, and thus have conceptually-narrower searches for context-features.

(3) Authors using the baseline cannot simulate how their entire concept expression operates on real-world cases, nor keep an issue list to guide the repair of expressions. Instead, authors are only provided the tool that allows them to view examples of place venues for a single place category. This allows users to clarify the usage of any context-features they are uncertain about, such as ones that are named in unexpected ways (e.g., what kinds of locations would be tagged on Yelp as the place category "active"?). We hypothesize that without the full set of simulate and repair tools, authors will miss cases where their concept expression operates differently than they intended, which will result in concept expression inaccuracies on real-world cases.

3.6.1.2. Participants. We recruited 14 participants from several undergraduate HCI classes from a mid-sized university in the Midwestern US. Everyone had some prior background in designing computing technologies. While participants were not specifically selected to have a background in developing location-based or context-aware computing applications, we provided them with sufficient background material on the potential uses of our application and how our application would detect end-user contexts; we describe the details of our background and tutorial procedure in the next section. Each participant was compensated with a $25 gift card for their participation in the 1 hour long study. For this between-subjects study, 8 of the participants were assigned to use the version of Affinder with all the technical features (unlimited vocabulary search, reflect and expand prompts, simulation and repair tools), and the other 6 were assigned to use the baseline version without all the technical features.

3.6.1.3. Study Procedure. Participants were asked to watch a 5 minute background video prior to joining a video conference study session. In this video, they learned about our vision for a context-aware, social application use-case in which the app helps friends engage in a digitally-mediated shared experience when they are in similar situations at distance [**103**]. We described an example scenario of how a friend in Chicago having an evening beer and another friend in Taiwan drinking a morning tea could use the application to share digitally-mediated cheers with their beverages. Participants were told they would act as a designer of this context-aware application that uses mobile context-features (e.g., place, weather, and time) to detect aspects of a situation that would support engaging in these shared experiences. Specifically, they were told that using Affinder, they would figure out (1) what the possible situations are where people can engage in the shared experience (e.g., raise their beverages for a digitally-mediated cheers) and (2) how to express these situations using context-features like Yelp place categories.

They were then given a guided, hands-on tutorial creating a concept expression for “situations to share a cheers” using the version of Affinder that they were assigned to (15 min); as the experimental version had more features to teach, their tutorial usually took longer. After the tutorial, users constructed up to two concept expressions for situations to engage in shared experiences, as time allowed (30-35 min). These situations included 'awesome situations to toss a frisbee' and 'situations for grabbing food that is good for a cold day'.

With the full feature version, we expected users to expand their notion of the concept they were expressing; therefore, these participants naturally spent more time on the task, and often had time to complete only one concept expression. With the baseline, participants tended to naturally run out of ideas and complete their concept expression early; thus, participants often had time to complete two concept expressions in the time available. We observed and recorded the participant's construction process, and asked them to talk-aloud to explain their decisions while creating the situation detectors. Finally, they completed a post-study questionnaire and a semi-structured interview (15 min).

3.6.1.4. Measures and Analysis. Our answers to the research questions are triangulated amongst 3 sources of data: (1) qualitative descriptions and summary statics about the concept expression, captured as it evolved during the construction process and once an author has completed it; (2) qualitative observations of authors' behaviors and usage of Affinder's features; and (3) qualitative insights about participants thoughts and strategies during their construction process. We opted to use this qualitative approach because we can evaluate the core features of Affinder together in one interface while still gaining insight into how using each of the core features make a difference in how the concept expression evolves and how an author's thoughts and strategies change.

To summarize the created concept expression, we measured the breadth of context-features included in the concept expressions by counting the number of relevant place context-features included.

To answer RQ1, we noted search queries that were made and which relevant context-features were added to the concept expression while using the unlimited vocabulary vs. the simple text search. Additionally, we used talk-alouds and retrospective interviews to understand (1) how the context-features a participant saw in the search results influenced their thoughts, and (2) how these updated thoughts led to foraging and adding additional context-features.

To answer RQ2, we asked about moments when users activated the reflect and expand prompts. This included details about (1) the initial idea they chose to reflect on; (2) the general concept they articulated that made their initial idea appropriate for the situation; (3) how the general concept later influenced their thoughts as they talked-aloud; and (4) what actions they did shortly afterward, such as unlimited vocabulary searches that followed or how their concept expression evolved.

To answer RQ3, we looked for cases when users, after simulating their concept expressions, updated their concept expressions. Through talk-alouds and revisiting these cases in the interview, we gained a better understanding of (1) a user's prior notions for how the intended a concept expression to operate; (2) what real-world case they found that posed an issue or surfaced a misconception in how the concept expression operated; and (3) how they decided to repair the concept expression through removing or explicitly negating specific context-features.

3.7. Results

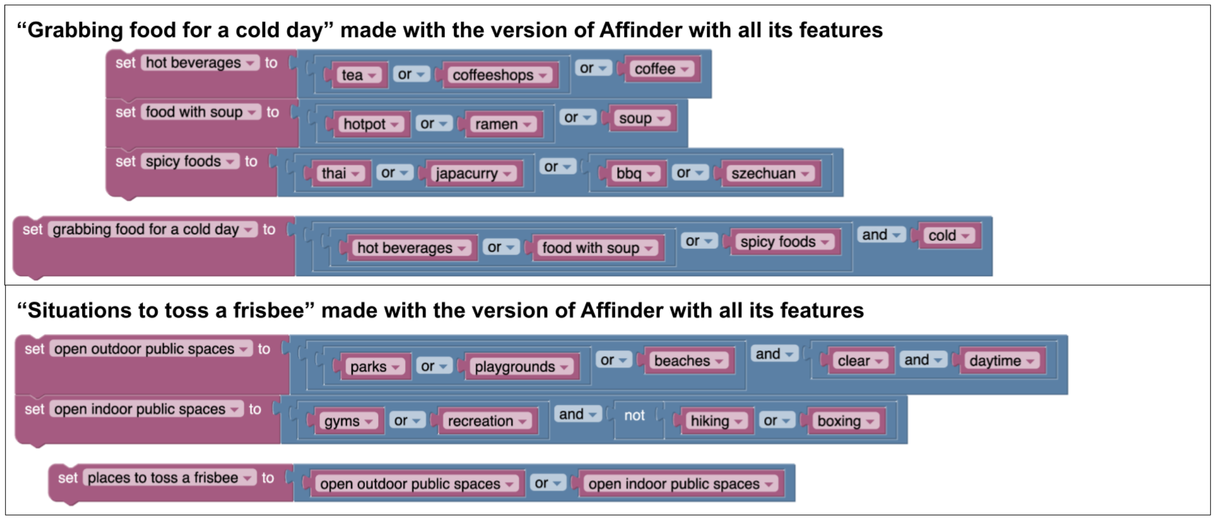
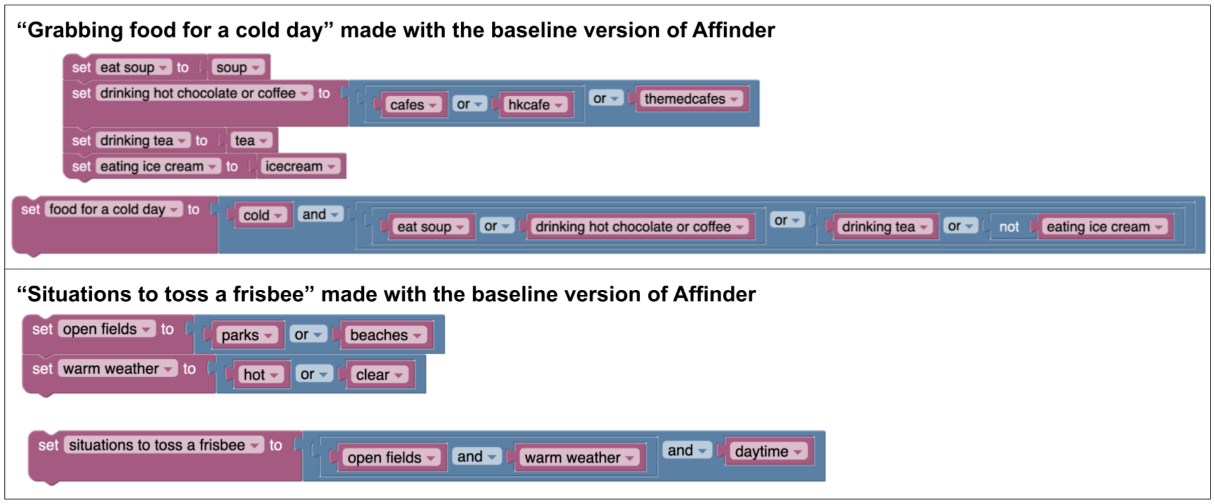

"Grabbing food for a cold day" made with the version of Affinder with all its features
set hot beverages to tea or coffeeshops or coffee
set food with soup to hotpot or ramen or soup
set spicy foods to thai or japacurry or bbq or szechuan
set grabbing food for a cold day to hot beverages or food with soup or spicy foods and cold
"Situations to toss a frisbee" made with the version of Affinder with all its features
set open outdoor public spaces to parks or playgrounds or beaches and clear and daytime
set open indoor public spaces to gyms or recreation and not hiking or boxing
set places to toss a frisbee to open outdoor public spaces or open indoor public spaces

Figure 3.6. Two concept expressions made with the full version of Affinder with all its features. The first construction expresses 'grabbing food for a cold day' as having hot beverages (tea, coffeeshops, or coffee), spicy foods (thai, japanese curry, bbq, or szechuan), or food with soup (hotpot, ramen, or soup) and while the weather is cold. The second construction expresses 'situations to toss a frisbee' as open outdoor public spaces (parks, beaches, or playgrounds, and when it is clear and daytime) and open indoor public spaces (most gyms or recreation, but excluding cases with hiking or boxing.)

Across both versions of Affinder, our 14 participants (8 for experimental, 6 for baseline) created a total of 20 concept expressions (10 for experimental, 10 for baseline). Figure 3.6 shows two constructions that were made using the version of Affinder with all of its core features. An example construction expresses 'grabbing food for a cold day' as any places serving hot beverages (tea, coffeeshops, coffee), food with soup (hotpot, ramen, soup), or spicy food (thai, japanese curry, bbq, szechuan) and where the weather is cold. Another example construction expresses 'places to toss a frisbee' as either open outdoor public spaces ('parks', 'beaches', or 'playgrounds') where the weather is clear and it is daytime, or open indoor public spaces (including most 'gyms' or 'recreation' but excluding cases



“Grabbing food for a cold day” made with the baseline version of Affinder

```
set eat soup to soup
set drinking hot chocolate or coffee to cafes or hkcafe or themedcafes
set drinking tea to tea
set eating ice cream to icecream
set food for a cold day to cold and eat soup or drinking hot chocolate or coffee or drinking tea or not eating ice cream
```

“Situations to toss a frisbee” made with the baseline version of Affinder

```
set open fields to parks or beaches
set warm weather to hot or clear
set situations to toss a frisbee to open fields and warm weather and daytime
```

Figure 3.7. Two concept expressions made with the baseline version of Affinder without all of its core features. The first construction expresses “grabbing food for a cold day” as when the weather is cold and a user is eating soup, drinking hot chocolate or coffee (cafes, hong kong cafe, themed cafe), drinking tea, or not having ice cream. The second construction expresses “situations to toss a frisbee” as open fields (parks, beaches) and when it is warm weather (clear and hot) and daytime.

that support outdoor activities, e.g., ‘hiking’; or activities associated with small spaces e.g., ‘boxing’). In both of these example cases, participants used Affinder to flesh out their concepts of the situation and bridge to a wide range of detectable context-features. Participants switched between top-down and bottom-up processes, allowing their concepts to evolve while foraging for the lower-level context-features. For example, a participant expressing “situations to throw a frisbee” started by foraging for the first places that came to mind such as parks and beaches; after declaring a concept variable for “outdoor public spaces” to unify these place context features, they realized that several outdoor and indoor places supporting athletic activities could also work, thereby expanding their efforts to forage for a wider range of place contexts.

In contrast, Figure 3.7 highlights two example constructions that were made using the baseline version of Affinder. One example construction expresses 'grabbing food for a cold day' as when the weather is cold and user is either eating soup (restaurants with 'soup'), drinking hot chocolate or coffee ('cafes', 'hong kong cafes', 'themed cafes'), drinking tea (places serving 'tea'), but not eating ice cream (places serving 'ice cream'). Another example construction expresses 'situations to toss a frisbee' as places with open fields (parks or beaches), where there is warm weather (clear or hot), and it is daytime. Similar to participants using the experimental version, participants using the baseline were able to create concept variables that linked their concepts of the situation to detectable context-features. In addition, participants were also able to move between top-down and bottom-up processes. For example, a participant started by foraging for place context-features that matched their initial concept of "drinking hot chocolate" and found many context-features related to cafes; then, they created more concept variables for other hot foods one might find at cafes like "tea" and "soup" and foraged for context-features matching these.

However, authors using the baseline declared concepts which were closer to the detectable context features (e.g., drinking tea) as opposed to more general concepts that would unify across context-features (e.g., hot beverages). As such, these concept variables were often linked to one or two place context-features (e.g., eating soup as 'soup'). For participants who did declare more general concept variables (e.g., open fields), they often struggled using the baseline to forage for a wider range of context-features matching these concepts.

We measured the concept expression for the breadth of context-features included in their constructions. For the concept expression ‘grabbing food for a cold day’, the median construction had 11 place context-features made with the full version of Affinder, while the median construction had 8 context-features made with the baseline version. For the concept expression ‘awesome situations to toss a frisbee’, the median construction had 4.5 place context-features when made with the full version of Affinder, while the median construction made with the baseline had 2 place context-features.

Having described an overview of the concept expressions created using both versions of Affinder, we now turn our attention to describing how each of the core technical features (unlimited vocabulary, reflect and expand prompts, and simulation and repair tools) supported designers’ construction processes.

3.7.1. Results for Unlimited Vocabulary Search

All 8 participants in the experimental condition used Unlimited Vocabulary Search to forage for context-features from the Yelp Places API that matched their initial concepts. For example, P7 tried to represent the concept of ‘snowy environments’ as a type of situation for ‘grabbing food on a cold day’. By searching for terms such as ‘ice rinks’ and ‘snow day’, they were able to find several place context-features matching their concept (skatingrinks, skiresorts, skischools). P1 said: *“I liked to see other examples of other [place contexts], I didn’t have to think about all these other [place contexts] on my own.”*

In contrast, participants using simple text search struggled to forage for context-features matching their conceptions. This happened because the concept vocabulary users formed as search queries (e.g., ‘spicy’, ‘grass’, ‘meadows’, ‘frisbee’) returned few or

search term	previous notion	feature that inspired updates	updated notion	behavior that followed
spicy	general notion of spicy food to warm the body	thai	curries in thai is a great example of spicy warm food for cold day	searched `curry' and added the feature `japacurry'
open space outdoors	physical outdoor locations with open space	active	places that support being physically active might be a good space for tossing a frisbee	searched `exercise' and adds the feature `bubblesoccer'
hot	hot chocolate and other hot food is good for a cold day	hotpot	asian soups are particularly good for cold days	searched `ramen' and added the feature `ramen'

Figure 3.8. Examples of queries made with the unlimited vocabulary search that helped designers discover context-features that inspired updates to their notions of their concepts. After a designer had shifted their notions, they often continued searching based on their updated notions and added new features.

no matching context-features. For instance, P7 expected various sport fields to show up after searching ‘fields’, but ‘baseballfields’ was the only result. In contrast with the experimental condition, P9’s search for a similar concept, ‘soccer field’, using the Unlimited Vocabulary Search helped them find ‘playgrounds’ and ‘college universities’ as relevant context-features.

Beyond finding detectable context-features matching their conceptions, participants using Unlimited Vocabulary Search were able to stretch and update their own concepts of the situation they were trying to express. This happened while browsing through the context-features in the search results that were associated but not what they were expecting to find. We present some examples from our user study where the search engine supported stretching concepts (see Figure 3.8). For example, when asked how the unlimited vocabulary search was helpful, P9 explained: *“What comes up in the [unlimited vocabulary search] suggestions already gives you ideas. For example, after seeing soup, I shifted my thinking to soup based queries, like pho and other soup noodle stuff... If I see something related, but in a different way, it helps me generate more [concepts] that I want to search.”*

Idea to Reflect On	Articulated Concept of Appropriateness	Searches that Followed	Relevant Concepts Added to Construction
hot chocolate	hot chocolate is a type of 'sweet' which is consumed during the cold, winter holidays	desserts, bakery	sweets to eat (desserts, bakeries)
coffee	coffee is an example of a 'hot beverage'	latte, mocha	hot beverages (tea, coffee, coffeeshops)
parks	some parks have vendors where you can buy and eat food outdoors; similarly, an 'outdoor festival' or farmers market with vendors selling food and drinks is perfect for eating outdoors	outdoor festivals, mead	outdoor festivals (cideries, wineries, fleamarkets)

Figure 3.9. Examples of how Reflect and Expand prompts helped users reflect on an idea part of their construction, articulate concepts about what makes that idea appropriate, and expand their foraging efforts by subsequently searching and adding new concepts and features to their construction.

In comparison, for participants using simple text search, the absence of related context-features led to stopping in their search for place context-features and expressing their concepts. For instance, when P6, a participant in the baseline condition, was asked how they decide when they have completed constructing, they said, *"When I run out of tags to search for... after finding 'parks' and 'playgrounds', grass and lawns wasn't showing me anything. Then I run out of ideas."*

3.7.2. Results for Reflect and Expand Prompts

Reflect and Expand prompts were used by 7 of the 8 experimental participants across 12 moments to articulate more generalized concepts and create new concept variables. 5 of the 7 participants used the articulated concepts to expand their efforts to forage for new context-features. We show in Figure 3.9 several examples of how reflection prompts led to expanded foraging efforts. 4 of these moments lead to successfully discovering new context-features that were added to their concept expression; while in 2 other moments, designers attempted to use the generalized concepts to search, but were ultimately unsuccessful at discovering new features.

As one example of how reflection prompts led to adding new context-features to a construction, P5 reflected on why their idea of hot chocolate was appropriate for 'grabbing food on a cold day'. Upon reflection, they realized that during winter holidays, "hot chocolate" is good with other sweet, hearty baked goods – implying that "hot" food isn't the only attribute, but any sweet foods that remind of winter would also be good to eat on a cold day. After P5 answered the reflection prompt, thereby creating a generalized concept variable named 'sweets to eat', they began to forage for place context-features matching this concept. By using searches such as 'desserts' and 'bakery', they were able to add 'desserts' and 'bakeries' as additional place context-features to the construction.

However, for 2 participants, foraging for context-features using the generalized concept variable did not find relevant context-features. For example, after P12 articulated how 'parks' are appropriate 'situations awesome to toss a frisbee' because they are 'outdoor public spaces', they could not find relevant features when searching for 'open public spaces' (which returned contexts like 'sharedofficespaces', 'libraries', 'publicservicesgovt', 'galleries', 'communitycenters'). They tried to revise their query by searching 'outdoors', but did not find any other context-features they did not already have in their construction. This suggests that the generalized concept variables created through reflection may not always be useful when used verbatim as search queries, which may require redesigning the prompts to guide users to not only articulate the general reasons for appropriateness, but also to form useful queries from these concepts that are aligned with the underlying textual metadata (e.g., how others describe Yelp reviews). Additionally, users of the unlimited vocabulary may need support in understanding why their query is not returning results they are expecting, and how they might revise their query to more effectively bridge to the relevant context-features.

Some users used the prompts to reflect and create generalized concept variables, but did not aim to expand their foraging efforts using these generalized concepts. Rather, 2 of the 7 participants who activated the reflect and expand prompts created a generalized concept variable that unified context-features that they had already added to construction workspace. For example, P9 reflected on why 'szechuan' was appropriate for 'situations to grab food on a cold day', then created the concept variable 'spicy foods', and finally used a logical 'or' operator to unify existing features like 'szechuan', 'japacurry', and 'thai'.

We sought to understand when and why users activate the reflection prompt during their construction process. Some participants used the reflection prompts during moments when they themselves felt stuck. Participants often felt stuck when they could not find additional place context features after trying several search queries. For example, as P7 was trying to express the situation "food for a cold day", they felt stuck and fixated: "*my mind keeps going to snow days, and from the standpoint of that situation, I think I got [all the context-features] out of that. So I think I should use a reflection prompt.*" Others participants understood the reflection prompts as a primary way to create concept variables, and used the reflect button to create concepts that unified existing context-features: "Once I equated the blue question mark as a way to create [a concept] that linked [context-features] together with a logical expression, I developed that association... and used it when I wanted to create a general category for the features I had added" (P13). Overall, authors initiated the prompts for reflecting and expanding their concepts

Construction Before	Prior Notions	Misconceptions or Issues found in Real-World Cases	Construction Afterwards
open_space_with_grass = (campgrounds or baseballfields)	baseballfields in a park where the grassy outfield is open to use campgrounds will have open grassy spaces to throw a frisbee	a place tagged with baseballfields and stadiumsarenas will be more restrictive and less appropriate to use the outfield if a campgrounds support activities like rafting, the physical environment will likely not have open grassy areas	open_space_with_grass = (campgrounds **and (not rafting)**) or (baseballfields **and (not stadiumsarenas)**)
big_open_space = (parks or beaches)	open space at a public park	there are some parks, like conservatories or lily pools that are tagged as gardens or venues, that are too fancy and inappropriate as places to toss a frisbee	big_open_space = (parks or beaches) **and (not (gardens or venues))**
hot_drinks_and_pastries = (bakeries or coffee or coffeeshops or coffeeroasteries or cafes or tea)	bakery cafes are where you can typically sit to have a coffee/hot chocolate with the pastry	a bakery in a grocery store does not support the same leisurely eat and drink experience	hot_drinks_and_pastries = (bakeries or coffee or coffeeshops or coffeeroasteries or cafes or tea) **and (not intlgrocery)**

Figure 3.10. Examples of Affinder's Simulate and Repair Tools helped users simulate their concept expression, identify real-world cases that highlight misconceptions and raise issues in precision, and repair their concept expressions.

during transition points in their process when they felt they had stopped on their current task to forage for context-features. In this way, its usage aligns with how Affinders was designed to allow authors to move flexibly between foraging for context-feature matching concepts (top-down) and fleshing out concepts that could link features (bottom-up) when issues or decisions arise during the construction process.

3.7.3. Results for Simulate and Repair Tools

Affinder's simulate and repair capabilities helped participants identify issues with how their concept expressions operated in real-world locations, and refine concept expressions to be more accurate. We present some examples from our user study where participants used simulation to find issues in real-world cases and refine concept expression to resolve issues; see Figure 3.10. Participants updated their concepts when they saw items from simulation different from their original mental image, helping enrich their understanding of the nuances of how context-features actually apply to real-world cases. As an example of this process (see Row 1 in Figure 3.10), P14 was originally thinking of ‘baseball fields’ as a ‘place for throwing a frisbee’, but recognized through simulation that some baseball fields are categorized as ‘stadiums and arenas’, which would make it harder to access the grassy outfield. Next, P14 used the repair shop to update their concept expression for ‘open space with grass’ to include baseball fields that are not also stadiums and arenas.

In another example (see Row 2 in Figure 3.10), P3’s original construction expressed ‘big open space’ as parks or beaches. Through simulation, they realized that some places, like a Conservatory or Lily Pool, are meant for formal events and activities and thus would make for an inappropriate location to throw a frisbee. Next, P3 used the Repair Shop’s Issue List to notice that the Conservatory was tagged as a ‘garden’ and ‘park’, while the Lily Pool was tagged as a ‘venue’ and ‘park’. To resolve these issues, the designer updated their construction to express ‘big open space’ as parks or beaches that are not also gardens or venues.

We were interested in how usage differed between simulate and repair capabilities in the full version of Affinder versus the feature for viewing example locations for a single place context-feature. Many participants simulated concept expressions once they had completed their representation for a concept variable. For example, P12 used the simulate feature composing together context-features for a ‘warm foods enjoyed indoors’ concept; they said: *“when I simulate my concept I am trying to find [location] options that don’t fit. But I’m looking and there’s really nothing that stands out.”* When designers did find location options that did not match their concept, they would label these issue locations and proceed to the repair shop to refine the concept expressions to resolve the issues. In this way, the tools for simulating an entire representation helped users to complete a representation of a concept variable, simulate to see if any issues arose for any of the multiple context-features, and resolve any of these issues as they arose.

In contrast, a majority of participants used view example locations to understand a specific context-feature they were uncertain about. For instance, P11 thought the context-feature ‘active’ might be relevant for tossing a frisbee, but did not know what this context-feature meant based on its name; viewing example places helped them understand that ‘active‘ contained many indoor gym and activity centers that were irrelevant for the ‘spacious and nature’ concept they were trying to express. In another case, P14 chose to view example places of ‘bbq’ because they thought that it would apply to locations where one can use a grill outside to barbecue, which is inappropriate for ‘good situations to grab food on a cold day’. Instead, the examples were barbeque restaurants, which they felt matched the situation they were expressing. In general, viewing example locations supported a designer’s while they foraged for features, helping them understand whether any individual feature they felt uncertain should be included in the expression.

However, view example locations can only identify misconceptions for context-features a participant chooses to inspect. For example, P11 said: *“I didn’t need to view example places for ‘campgrounds’ and ‘baseballfields’ because I already know what those types of places are.”* This suggests that if participants feel certain about the context-features, they may not choose to view example places for those features. However, this same participant later simulated the entire concept expression, and found nuances in how a context-feature belonging to the expression applies, such as how ‘campgrounds’ that support ‘rafting’ would not apply to their concept because such locations would have more rivers instead of open grassy areas required for tossing a frisbee. This suggests that simulating and refining a concept expression after they have completed it serves a role in surfacing inaccuracies in how a context-feature detects real-world places, without designers having to explicitly look for them.

3.8. Discussion

Having demonstrated how an Expression Interface Layer can help authors of context-aware experiences express their concepts of a situation to machines, we first revisit the core ideas behind an Expression Interface Layer and discuss how it support people's ability to bridge the semantic gap between their overarching idea and a machine's lower-level constructs. Then, we envision the role of Expression Interface Layers for developing intelligent, context-aware applications that can facilitate human activities and experiences.

3.8.1. Enable the Expression Interface Layer between Human Concepts and Machine Representations

In this work, we uncovered a set of bridging challenges that arise when designers of context-aware applications express conceptually rich situations that are several layers of abstraction removed from the underlying machine features. Accounting for this, we developed the Expression Interface Layer to include a set of cognitive bridging tools that support people's cognition in the process of translating from their high-level concepts to the machine's available constructs. Our approach is distinguished from conventional interfaces for programming an intelligent, context-aware system, as an Expression Interface

Layer explicitly supports human cognition when interfacing with context-features. The Expression Interface Layer implemented into Affinder supports two interconnected cognitive processes. First, Affinder helps designers define a more expansive set of concepts for how a situation affords engaging in a human activity or experience. Second, Affinder helps designers find relevant and accurate links between their intermediate concepts of a situation and the features that a context-aware AI system can understand and detect.

3.8.1.1. Fleshing out and Expanding Human Concepts. We found that expanding one's conceptual notions was important for helping designers explore the conceptual space and overcome challenges with underscoped concept expressions. Designers who used the reflect and expand prompts stretched their concepts of a situation by reflecting on why a context-feature they added was appropriate for the situation they were trying to express; this process of using the prompts inspired new searches, ultimately helping with overcoming underscoping of concepts. Additionally, designers who used the unlimited vocabulary search tool encountered context-features that shifted their notions of the situation they were trying to express; these updates led to expanding their search queries for additional context-features.

The core ideas behind these two techniques for conceptual stretch have parallels to existing research tools for conceptual ideation and overcoming design fixation. First, the cognitive bridging tool for reflecting and expanding on concepts overcomes fixation on early ideas by scaffolding designers to re-represent their initial ideas at a higher-level of abstraction, to help designers remember other contexts that might also apply. This core idea adapts methods for re-representing general linguistic terms to increase the chances that people recall additional ideas from other parts of the conceptual space [**98**]. This literature uses these methods to help product designers remember useful analogs, i.e., solutions that can be adapted from other domains to solve their current design problem. In contrast, we use it to support context-aware application designers to re-represent their human concept of a situation at different levels of abstraction, so that they can find multiple ways of expressing it using context-features.

Second, the Expression Interface Layer's tools for searching for relevant context-features led to conceptual stretching in several cases by helping with the discovery of place contexts that were conceptually different from their initial concepts. Taking this finding more broadly, designers encountered machine constructs while foraging for an initial intermediate concept that expanded their concepts of the overarching goal they were expressing. Within HCI, several computational ideation tools have also supported the search and discovery of example ideas for the purposes of inspiring and influencing a designer's concepts [**124, 125, 133**]. Many of these tools for supporting conceptual stretch have been isolated to the concept-ideation phases of the design process, where designers are sketching a description or image of a product design idea. With Expression Interface Layers, searching for conceptually-inspirational example items is interconnected with other stages of the design process like finding implementable forms for concepts—as the unlimited vocabulary search tool can serve the dual-purposes of stretching a designer's concepts, and helping them find matching context-features for an existing concept.

Supporting cognitive processes like conceptual stretching will be important for other design or creative processes in which ideation and implementation are interconnected processes. One such creative process is in the field of human-AI co-creation with generative models, where users must express their creative concepts through a collaboration with an

AI capable of generating content. Within the domain of music co-creation, composers must flesh out their concepts for how to achieve a creative goal (e.g,. expressing the emotions of sad and stuck in the music), while also strategizing on how to implement their concepts through using the available controls to steer the AI to generate music that expresses their intent [**102**].

Studies of human-AI music co-creation have observed cases where a human creator will change their own concepts of how to express a human emotion through music, based on interacting with music alternatives generated by the AI [**101**]. This finding represents a case where the exploration of implementable solutions can inspire refinement of human concepts. In addition to example-driven conceptual shifts, such tools can encourage reflection and expansion of concepts through re-representing concepts at different levels of abstraction.

3.8.1.2. Linking Human Concepts and Implementable Machine Representations. In addition to coming up with concepts for solutions, an Expression Interface Layer also supports finding an implementable form–in that it helps designers translate their concepts into machine representations using detectable context-features. The final output is a piece of code that uses detectable context-features as input, and can be used by an intelligent, context-aware agent to identify the situation across distributed contexts. The problem an Expression Interface Layer solves makes it distinguished from conceptual ideation tools for this reason. The Expression Interface Layer effectively supports this translation process by providing (1) a rich vocabulary in which authors can query the available set of machine features, and (2) tools for checking how a machine's features may or may not operate on realistic cases as a designer intended.

The Expression Interface Layer's core ideas have parallels with interactive machine learning paradigms, such as active learning and machine teaching. In this setup, a human evaluates and refines a machine representation, such as machine classification model, by testing it on new and existing cases and changing the machine's representations—either indirectly via providing labels to a machine learner [**5**], or more directly by specifying or removing features in the representation [**20**]—to improve the machine's ability to accurately identify the desired concept. Most of this work typically relies on humans to provide labels of a concept so that a learning algorithm can infer which machine features might be relevant to include or to put greater weight on. The Expression Interface Layer takes a different approach, allowing a human's concepts, externalized through natural language, to be used as queries for discovering which machine features could be useful. We argue that this is an effective and complementary approach by letting humans find useful machine representations through using a human's richer notions of the concept. This approach should be useful when the machine features are too numerous to browse through, but do have semantically-meaningful metadata that can be used for querying.

Bridging problems can manifest in other domains where humans must express their rich concepts into implementable machine-representations. As we have discussed, bridging problems are a marriage of the problems of conceptual ideation and finding implementable machine representations for concepts. As such, tools like Affinder which overcome bridging challenges need to embrace ideas from both literatures on design concept ideation and interactive machine teaching. We argue that the next generation of expression tools will closely support the processes of fleshing out concepts and linking to machine representations. There is a double loop between humans refining their mental concepts, and refining the machine representations too. In this way, construction processes, like the one Affinder aims to support, help to create expressions that are more human, while also more detectable.

3.8.2. Imbue machines with an understanding of human situations and the experiences and activities they afford

Two decades ago, Abowd and Mynatt envisioned a near future in which computing technologies would augment and benefit our everyday lives. Everyday computing would support a mode of continuous interaction where computing technologies were no longer just a localized tool, but a constant companion that runs in the background, and could act opportunistically to promote the informal and unstructured activities of our everyday lives, from orchestrating tasks, to communicating with family and friends [**2**]. At the same time, researchers began prototyping the types of context-aware applications to step towards this vision, largely enabled by technical advances in sensors and algorithms for inferring aspects of human context, as well as frameworks and toolkits that made it easier to write applications using sensors and component detectors [**35**]. Yet amidst the technical opportunities afforded by such advances, it became apparent that the importance of context-aware computing extended beyond the contextual factors that machines can detect (e.g., spatial location, user identity, proximity of people and devices). Equally important was considering how these contextual factors contribute to the meaningfulness of humans acting and relating in situations [**42**].

In some ways, these challenges still exist within the current landscape of technologies. The technologies we have today for inferring context—including the variety of commercially-available physical and virtual sensors (e.g., cameras, smart home devices, location-based metadata, social media and application usage) and the machine learning and algorithms for processing this sensor data—are starting to give applications a richer understanding of people's everyday worlds. Yet, to really leverage these component detectors to create applications that have an awareness of our human ways of acting and relating in situations, application designers ultimately need to bridge between their human concepts of a situation and the machine's available detectors. Thus, it becomes ever more important to develop an Expression Interface Layer that helps individuals in representing overarching human ideas with the available component detectors.

In the near future, as emerging technologies like augmented reality (AR) and virtual reality (VR) will likely take an increasing role in mediating our social interactions and personal activities within everyday contexts [**67**], a richer understanding of our human situations in these environments will be important for ensuring these technologies can be aware of and facilitate the experiences users want to have. Devices for AR, such as AR glasses and mixed reality headsets, will have access to the familiar set of context-features based on location, activity, and time that current mobile context-aware platforms provide—as well as additional context-features such as object-classes recognized through computer vision [**86**]. Despite the increased number of fine-grained context-features that AI systems can infer, designers must figure out how to best articulate one's concepts for situations with the detectable features. Thus, an Expression Interface Layers can empower designers to imbue these applications with a deeper understanding of these human situations and the activities afforded by them.

We anticipate that tools for expressing the human ways of relating and acting in situations will be important for facilitating user social experiences and activities occurring in VR as well. Collaborative virtual environments should have a sense of place, where there are social norms and understandings of what experiences or activities are appropriate in these environments [**61, 43**]; thus, systems that facilitate interactions in these virtual environments should have an understanding of human experiences and activities that can take place in these situations too. While some virtual places may borrow the social norms from the real-world physical places they are modeled after (e.g., a VR bar or cafe), it will be important that application designers have the tools to express their concepts of how virtual situations may differ from their physical situation counterparts, and what experiences and activities in virtual reality are afforded and appropriate in them.

3.9. Limitations and Areas for Future Work

3.9.1. Extending Beyond Location-based Context Features

The Expression Interface Layer built into Affinder supported designers in encoding concepts of situations in terms of *location-based contexts*, such as place venues that one might encounter while being mobile across one's day. Since designers must express their ideas of situations in terms of place categories, their expressions are more useful for identifying situations that could occur in public venues (e.g., parks, types of restaurants) but currently less useful for detecting situational contexts that occur within the home or office as detected by indoor sensors. To expand the set of detectable contexts, future developments of Affinder could include additional base context-features beyond Yelp place categories. When extending this context-set, it would be important to consider whether it is sufficient to create another toolbar list that a user can browse through, or whether techniques like the unlimited vocabulary search would be needed to help users discover context-features. For example, for context-features describing different areas of a home (e.g., dining room, kitchen, bathroom, garage, backyard), it may be sufficient to list them in Affinder's toolbar for users to browse through.

3.9.1.1. From Designer-driven to End-user-driven Authoring of Context-Aware Experiences. We developed Human-AI Interface Layers to aid the communication of ideas for context-aware experiences to AI systems, and focused on helping application designers who were fostering contextual activities among diverse end-users. Nonetheless, issues can arise when an application designer's concept of situations is not commonly shared with the end-user who will encounter these situations, perhaps due to differences in personal preferences or cultural understanding. For example, end-users may have their own personal interpretation of a concept for a situation that differs from the application designer's concept of it (e.g., an end user might prefer soupy foods over spicy foods for "food good for a cold day"). If a context-aware application uses a concept expression that does not align with end-users' concepts of the situation, the app will try to facilitate the activity in situations that some end-users will not agree with.

Therefore, a promising direction for future work is to develop Expression Interface Layers that involve multiple authors in the construction and customization of how concept expressions operate, in order to be accountable to these diverse end-users' concepts of a situation. Pursuing such a direction would build upon foundational work on intelligibility, accountability, and control for end-user interaction with context-aware applications [**12, 37**]. Affinder's concept expressions—logical predicates which are composed of semantically-meaningful intermediate concepts and context-features—are in an intelligible representation that makes them good candidates for supporting this desired control and customization by other authors.

An end-user driven authoring paradigm prompts the question: if end-users start authoring context-aware experiences tailored for themselves, how would this impact how they express their overarching ideas of experiences into machine representations that AI agents can computationally act upon? Our answer to that question is as follows: Whether designers or end-users drive the creation of context-aware experiences, the bridging problem is what the human will have to face. Similar tools or techniques would be necessary to flesh out concepts, gather pertinent contextual features, and pinpoint misconceptions about how machine constructs operate.

One point of difference is the extent to which the person must enumerate the various ways an overarching idea of a situation could be realized. Whereas designers strive to create experiences that are adaptable across various contexts and cater to a diverse user base, end-users can author experiences designed with specific friends in mind, often situated within a limited geographic scope. For instance, an end-user describing the experience of 'warm comfort foods on a cold day' might only need to include warm comfort foods that align with regional tastes, personal tastes, or those of their friends.

Furthermore, the advent of end-user creation could open the possibilities to a paradigm of collaborative authoring among a community of end-users. Each individual end-user would have the opportunity to express their personal interpretation of the shared experience, such as "warm food on a cold day." This approach could lead to the reuse and adaptation of concept expressions conceived by other community members for the same experience. Leveraging this collective effort, cognitive bridging tools could play an integral role in facilitating such collaboration. For instance, additional techniques inspired by analogical design principles, which involve re-representing concepts to overcome cognitive fixation, could be applied to counteract potential fixation issues when extending collaborators' concept expressions during the process of individual authoring.

Thus, while the extent and specific techniques might differ, the fundamental cognitive processes underlying the translation of a high-level idea for experiences to the detectable context features represent a unifying thread. To effectively formulate human ideas and translate them into the machine's constructs, both designers and end-users will need authoring environments that include Expression Interface Layers and the cognitive bridging tools they offer.

3.10. Conclusion

In this chapter, we examined a kind of challenge when the user's overarching idea they are communicating to an AI system is several levels of abstraction removed from the AI system's operable constructs. Our investigation was based in the domain of programming context-aware applications: we aimed to empower designers of a context-aware experience to more easily express their ideas of a conceptually-rich human situation and the interactions they afford to machines, so that the applications can be aware and responsive to such situations across distributed contexts.

To address this, we contribute an Expression Interface Layer that provides a designer with cognitive bridging tools that address challenges that arise when forming one's high-level ideas and finding relevant and precisely-matching machine constructs for one's concepts. We built an Expression Interface Layer into Affinder, a visual programming environment for expressing concepts of situations using location context detectors that intelligent, context-aware agents can computationally act upon. Affinder's technical contribution includes 3 core features designed to overcome challenges when constructing concept expressions: (1) an *unlimited vocabulary search* for discovering features they may have forgotten; (2) *prompts for reflecting and expanding* their concepts used for organizing and foraging for features; and (3) *simulation and repair tools* for identifying and resolving issues with the precision of concept expressions on real use-cases. In our studies, we show that these features can (1) mitigate underscoped expressions by helping designers discover context-features relevant to their concepts, (2) stretch people's concepts for what aspects they consider to be important for enabling interactions in these situations, and (3) uncover and resolve mismatches in how a creator expected their concept expression to operates vs. how it actually executes across real-world, distributed contexts. Our work with Affinder and Expression Interface Layers represents an exciting direction for the development of intelligent and context-aware applications. Complementing the predominant paradigm of developing a richer set of component detectors, our research explicitly focused on advancing the capabilities of humans to bridge between their high-level ideas and the lower-level machine representations.

Our design of the Expression Interface Layer in Affinder provides numerous techniques to support effective cognition processes as designers bridge between their concepts and the machine representations. In future work, we envision that AI techniques like semantic embeddings and knowledge graphs could play a larger role in supporting designers during the construction process. As one example, using language embeddings [**107**] could enable ways to recommend concepts and vocabulary that would be better aligned with what humans actually want (e.g., if the term ‘spacious’ is returning undesirable place contexts that are indoors, we might add the term ‘outdoors’ to the embedding to direct the search). In another direction, commonsense knowledge graphs [**14**] could help users traverse a graph of related concepts to explore and discover new concepts.

CHAPTER 4

An Interface Layer for Recognizing and Stating Implicit Expectations for Execution

Even when an individual has expressed instructions to an AI agent to carry out, agents can take action and make decisions that disappoint the individual when actually carried out. This occurs because a person may forget to communicate their implicit expectations and an automated system can *fail to reveal constraints* it might face that would have otherwise prompted individuals to clarify all that they expect when delegating personal tasks to an AI agent.

To resolve this, I introduce in this chapter an *Execution Interface Layer* for recognizing execution constraints and stating implicit expectations for how to adjust to these constraints. Using the domain of opportunistic interdependent experiences as an example, I advance three techniques illustrating how an Execution Interface Layer can help designers communicate to intelligent context-aware agents their expectations for how to mitigate breakdowns in social norms when facilitating socially interdependent experiences. In one technique, this Execution Interface Layer can use computational models to simulate execution to help an individual recognize potential breakdowns before they occur. Made aware of the potential breakdowns due to constraints, the individual is supported in reformulating their instructions, which would mitigate these breakdowns. In the other two techniques, the Execution Interface Layer empowers individuals to state their expectations via a new mechanism, allowing AI systems to adapt decisions when facing constraints to satisfy an individual's expectations.

All together, the development and evaluation of these three techniques demonstrate how an Execution Interface Layer can mitigate disappointing breakdowns by enabling individuals to communicate about execution constraints and implicit expectations, before handoff to an automated system.

4.1. Introduction

Despite individuals providing explicit instructions to an AI agent, its actions and decisions can sometimes disappoint due uncommunicated constraints and expectations. For example, an automated system can automatically plan an itinerary for a user, but the itinerary it outputs has packed too many activities in one day which a person may not have the energy to do [**137**]. A navigation route planner can suggest directions to walk and take a bus, but the suggested route isn't workable because the user walks slower than the "normal" speed needed to catch the bus on time [**6**]. These disappointments stem from individuals possibly overlooking sharing their expectations, and automated systems not disclosing potential constraints that would have prompted individuals to clarify.

As a focal example, consider a designer of a digitally-mediated social experience, called an Opportunistic Collective Experience [**103**], for connecting friends and colleagues who live across geographic contexts. A designer delegates to an intelligent context-aware agent the job of identifying moments that arise coincidentally when physically distant people are in a similar situation or doing a similar activity and structuring shared activities or experiences in these moments, that without such intelligent agents, would have otherwise been missed. Despite designers providing instructions for how many people and in what situations to coordinate a social interaction, *intelligent agents can still fail to successfully realize these experiences due to uncommunicated implicit expectations.* Experiences can break down if one's friends and family are not there to reciprocate participation (e.g., taking half of a photo that is never completed by others in their family, or raising a glass with no friend to return the cheer for some time), and incomplete activities and artifacts can disappoint (e.g., building a photo mosaic that never reveals its shape after fewer pictures are submitted than expected). The core problem is that context-aware agents can face constraints when executing these experiences—including the uncertain and limited availability of people in specific situations to participate. When designers forget to state how an AI agent should handle cases when there are fewer people available than expected to complete an experience, these systems will lack the awareness and sensitivity about user expectations, i.e., mitigating breakdowns in social interactional norms that degrade the intended social benefits of an opportunistic experience.

To handle execution constraints and implicit expectations, I propose in this chapter an *Execution Interface Layer* for recognizing implicit human constraints and expectations, and communicating them to automated systems so they can adjust their behaviors to accommodate them. Augmenting the use of an existing context-aware system for coordinating these experiences, it provides (1) constructs for users to state their expectations regarding interaction norms; (2) tools and mechanisms for reasoning about how implicit expectations for preserving interactional norms can be affected by the constraints of people's availability to participate; (3) tools and mechanisms that promote flexible strategies for reformulating the desired experiences and adapting execution behaviors to accommodate those implicit constraints and expectations.

This Human-AI Interface Layer supports the communication of implicit constraints and expectations in two directions. From the direction of AI to humans, additional AI models can help people recognize how their implicit expectations could be impacted by how they've currently articulated their desired goals, which can then help them decide how to reformulate their original goals in light of these implicit constraints and expectations. For instance, an awareness of whether participants are likely to encounter certain situations can help a designer reformulate a social experience that won't work in some parts of the world by adapting and broadening the set of situations one needs to be in to be able to participate (e.g., broadening a 'ramen sharing' experience that won't work in places where ramen is less common to a 'noodle sharing' experience). From the direction of humans to AI systems, people can use intuitive and actionable constructs, powered by new AI system components, to state their implicit expectations. The new AI system components are capable of reasoning about changes in implicit constraints—ultimately allowing the combined AI system to make the decision that best balances explicit goals and implicit considerations. For example, an awareness of when starting an experience that requires timely completion across participants is likely to garner sufficient participation can allow an automated system to decide whether to initiate an opportunistic social experience that requires timely reciprocity (e.g., whether to start a shared lunchtime experience that can fail if no one is available to reciprocate earlier participation within a desired experience time window).

This Execution Interface Layer addresses problems in respecting implicit interaction norms for three corresponding types of interdependent, opportunistic experiences:

- Consider a type of opportunistic experience that fosters connection through people participating in similar situational contexts, or what we call *situational interdependence*. In defining an experience's situational requirements, designers can fail to realize that other implicit expectations such as ensuring that people across geographic contexts can easily access the situations for connecting in this experience. This Execution Interface Layer reconfigures the tools for defining situational requirements, by providing designers models of people's availability to participate to help surface the impact on the implicit norm of inclusive access to opportunities and scaffolds for reformulating their explicit goals for the experience to better respect these implicit norms.
- An opportunistic experience that fosters connection through two people participating together within a shared time frame, or what we call *temporal interdependence* [**17**], can break interactional norms if the experience is started but others are not available to join to reciprocate. To prevent this from occurring, this Execution Interface Layer adds a new decision-theoretic mechanism that evaluates when to start an experience based on the expected value of completing it versus the potential costs (e.g., how long it will take to complete it, if at all); designers can convey their implicit expectations through a construct for specifying the time frame in which social reciprocity is valued.
- Finally, the Execution Interface Layer can help designers state their implicit ideas for how social experiences can still provide social value from people's unique

contributions towards a shared goal together (e.g., pictures forming a timelapse of a sunset) when there are fewer than expected participants available. We introduce a *programming model to encode the value of partial results* which enables designers to state their social value of different partial results of the collective goal as the experience progresses which the AI agent can then use to prioritize which roles it coordinates available participants, allowing it to maintain *role interdependence.*

Through three evaluation studies, we show how the Execution Interface Layer can empower designers and intelligent agents to communicate about execution constraints and implicit expectations for three corresponding types of interdependent, opportunistic experiences. First, we find that the Execution Interface Layer, instantiated in a tool for defining situational requirements, established an iterative loop between a designer formulating their definition and envisioning the intelligent agent's execution under simulated constraints, which helped designers recognize and address breakdowns in experiences being inaccessible for a geographically-diverse target population. Second, we show that an Execution Interface Layer can be used to impart automated systems with a sensitivity to interactional norms of reciprocity—deciding to only initiate experiences when they are likely to complete in a timely manner. Third, the Execution Interface Layer can be used to convey implicit ideas on the value of desirable partial results for a social experience, so that automated systems can generate artifacts from a limited set of contributions that still reveal the desired shape of a collective artifact a group was co-creating.

Taken together, these results highlight that Execution Interface Layers can mitigate disappointing breakdowns by communicating about execution constraints and implicit expectations, before handoff to an automated system.

4.2. Background

The conceptual foundations of this work draw on ideas in HCI that focus on the problem of users forgetting to consider implicit constraints when configuring and using automated systems. Relatedly, this work aims to address practical problems in recognizing and accommodating the implicit constraint of people's availability to engage in computer-mediated social interactions. As such, it also builds on work from the areas of social computing systems, interruptibility, and crowdsourcing.

4.2.1. Implicit Constraints and Expectations when using Automated Systems

The issue that many expectations or constraints are assumed or missed and therefore not explicitly stated by the user has been studied in the field of human-computer interaction. A crowd powered system can automatically plan an itinerary for a user, but the itinerary it outputs has packed too many activities in one day. A generative AI assistant can help draft an introductory message, but the draft is too verbose for a receiver to read over a text message. A navigation route planner can suggest directions to walk and take a bus, but the route isn't workable because the user walks slower than the "normal" speed needed to catch the bus on time. In such systems, a reasonable solution is to have the "person in charge" (e.g., requester, user, novice creator) evaluate the systems outputs as they are being created and clarify their expectations accordingly (e.g. add their missing constraints to the planning mission; communicate their additional expectation for the generative AI to revise). Contrary to these examples, this chapter focuses on cases of user handoff to automated systems, where such systems are taking proactive action without user intervention or oversight. In such cases, it would be too late for users to recognize their implicit expectations after the system has already been deployed to make decisions or take actions. To evaluate the potential actions of these systems, people need to use simulated scenarios and data to verify and test potential behaviors in order to surface failures caused by implicit constraints or expectations. Specifically, we study how designers use automated systems—such as context-aware systems to facilitate social interactions—and consider the issues with improving the likelihood that these automated systems can accommodate user's implicit expectations and constraints when performing tasks on their behalf. Like previous work has done, we simulate how an automated system would operate based on data of people's availability to visit specific categories of location across multiple cities to highlight any geographic/regional bias in the activities a designer wants to promote.

In addition to recognizing unstated constraints and expectations, adapting an automated system's operation to accommodate these implicit expectations is equally important. Some systems make it easy to accommodate constraints once they have been recognized by a user. For example, a system for itinerary planning can conceivably take into account additional constraints, and a generative system can adapt once a user has revised their request. In our work, however, accommodating implicit constraints can be less straightforward because the presence of constraints can imply making tradeoffs with the original goals.

To enable this general approach, we take inspiration from flexible strategies for achieving desired computational outcomes despite uncertain and limited computational resources in computer systems operations, such as graceful degradation [**105, 65**]; flexible computation [**71, 69, 72**]; and anytime algorithms [**139**]. The key idea behind such strategies is considering the expected costs and benefits of various ways of realizing a computational goal given resource limitations and uncertainty. For our setting, we achieve our desired goals for opportunistic interdependent experiences by creating **models of people's uncertain availability to participate opportunistically**, which allows us to develop designer-facing and automated systems that support reasoning about various strategies for achieving the intended interdependence given uncertain but anticipated availability.

4.2.2. Considerations for People's Availability in Personal, Social, and Crowd Computing

Practically, we aim to support technology-mediated social interactions given the implicit constraint that people may not always be available to participate. However, many social technologies have not not needed to worry about the negative impacts that participant constraints, such as people's availability to participate, can have on the intended social interactions. Existing work in HCI and social computing has largely side-stepped this challenge by either (1) making simpler experiences without interdependence; (2) planning interactions that forgo opportunistic participation; or (3) deploying in large communities where viral participation is common. First, while removing interdependence between contributions can reduce issues with coordination [**22, 11**], this limits the quality of interactions since positive interdependence in shared activities is important for promoting social closeness and the feeling of being part of a group [**85, 34**]. Second, while requiring commitments on participation can make it easier for the system to coordinate interactions (e.g., only activating group events after a critical mass has committed to attending [**23**]), the higher effort required to commit dedicated time means that people don't always find opportunities for interacting as frequently as they might otherwise. Finally, targeting larger online communities can open the possibility to viral participation, such as what is seen on sites like TikTok [**129, 97**]. While clever, virality is not a useful method for coordinating engagement among smaller groups of people, such as friends and families. Ultimately, by side stepping the challenges coordinating opportunistic interactions when participation is uncertain, HCI research has largely moved away from supporting this interaction modality altogether.

Our work largely thinks about how to improve the ways computers understand and engage people in interactions, despite uncertainty over people's availability for interaction. These aims are largely shared by research on models of human interruptibility [**49**] too, where previous efforts focused on getting computers to appropriately engage a user given uncertainty about whether they were available to interact (e.g., give attention to an incoming computer notification or communication message). Decision-theoretic approaches have been used to effectively reason about engaging a single individual through considering the value of engaging a user and the cost of the interruption [**70**]. Much like that, our work takes a decision-theory approach, modeling the uncertainty in availability for an interaction, the value of satisfying the interdependence we aim to promote, and the costs of violating interactional norms of the social interaction (e.g., the second person not being available to reciprocate in a shared experience). Extending beyond modeling whether an individual is available for an interaction, our work to coordinate social interactions must model the social interdependence in how multiple people might successfully interact (e.g., via sharing a similar experience or constructing a collective artifact together). In this way, we move beyond a focus on an individual's interruptibility, to consider the relational values and costs associated with engaging with others and how the uncertainty of further engagement with others may affect the value of the interaction as a whole.

Our work also focuses on effectively designing the ways people contribute to shared activities, and coordinating when and for which contributions people make. In this way, it resembles in part crowdsourcing research on optimizing how to structure and coordinate contributions to collective efforts, albeit towards the goal of getting tasks completed. While a large body of early work assumed participants could be recruited on demand, later work considered how contributions could be sourced opportunistically, or on-the-go, while people are going about their existing routines and routes [**120, 90**]. In on-the-go crowdsourcing, researchers have also, similar to how we do it, advanced techniques that model the uncertainty in participation, such as whether people will be available in specific situations (e.g., passing by a task location), and reason about when and who to engage based on this model [**89**]. However, a key difference between tasking and social interactions is that the values are not over the completion of tasks but over the quality of the interaction. This has implications for needing to reason about how interactional norms would be violated by a lack of reciprocation, or by disappointing partial results, which is critical in our setting but largely out of mind in the crowdsourcing setting. Moreover, this focus on quality of interaction rather than exactly on which contributions participants must fulfill provides opportunities to apply flexible strategies to adapt what roles or contributions that we can fulfill with the interactional resources that are available.

4.2.3. Opportunistic Collective Experiences

We study how creators communicate with intelligent agents about execution constraints and implicit expectations in the area of facilitating opportunistic, interdependent social experiences at distance with intelligent context-aware agents. Our previous research on Opportunistic Collective Experiences (OCEs) [**103**] demonstrates the potential benefits of using intelligent context-aware agents to facilitate people's shared interactions during coincidental moments that arise in their daily lives, thereby giving friends and family who are no longer colocated convenient opportunities to connect that would have otherwise been missed. In contrast to social media feeds which often promote passive consumption which is not associated with social connection outcomes [**22**], OCEs promote social interdependence at a distance through surfacing to people coincidences in their similar situations to foster a sense of shared experience [**15**] (e.g., grabbing a warm meal on a cold day together; jumping into the water at the beach), or inviting people to construct a digital artifact together or to achieve collective goals by making unique contributions through local situations (e.g., contribute snapshots of the sun setting to enable the creation of a Sunset Timelapse). Additionally, the opportunistic nature of interaction reduces the burdens of planning or initiating interactions typically required for actively engaging via a call or direct message. In these ways, a core benefit of opportunistic collective experiences is that they allow for rich, interdependent social interactions that are possible with planned interactions but which find ways to work during convenient moments in users' busy lives.

Designers of OCEs can structure a specific kind of shared experience or activity by using an domain-specific API to describe the contributions needed for an activity and the situational requirements for contributing (e.g., to recreate the feeling of a shared meal time, the opportunistic experience needs two people to contribute photos when they are grabbing a meal in their respective cities). The context-aware AI systems can then facilitate opportunities to have these experiences for people separated by time and distance, by monitoring the everyday situation of group members and coordinating participation in these shared experiences and interdependent activities in appropriate situations.

Despite this promise, deployment studies of OCEs have also highlighted the difficulty of executing them in practice. This execution challenge is rooted in the opportunistic nature of people becoming available in situations to fulfill an experience's interactional needs, which implies that participation in experiences is uncertain. An experience's situational interdependence can break down if others are not in a shared context to reciprocate participation. For example, [**73**] designed and deployed OCEs to pairs of acquaintances, but found that some pairs did not visit similar contexts to reciprocate their participation while the experience was still fresh. Additionally, [**103**] found in their deployment studies with college alumni friends that incomplete artifacts and collective activities can disappoint (e.g., taking half of a photo that is never completed by others, or a collaborative storybook that does not get past its first few pages). Based on these challenges observed during deployments, OCEs, therefore, serve as an apt scenario for studying the interdependence breakdowns in opportunistic participation.

4.3. Interdependence Breakdowns in Opportunistic Participation

In this section, we will discuss breakdowns of interdependent interactions when participation occurs opportunistically. Using OCEs [**103**] as an illustrative scenario, we show

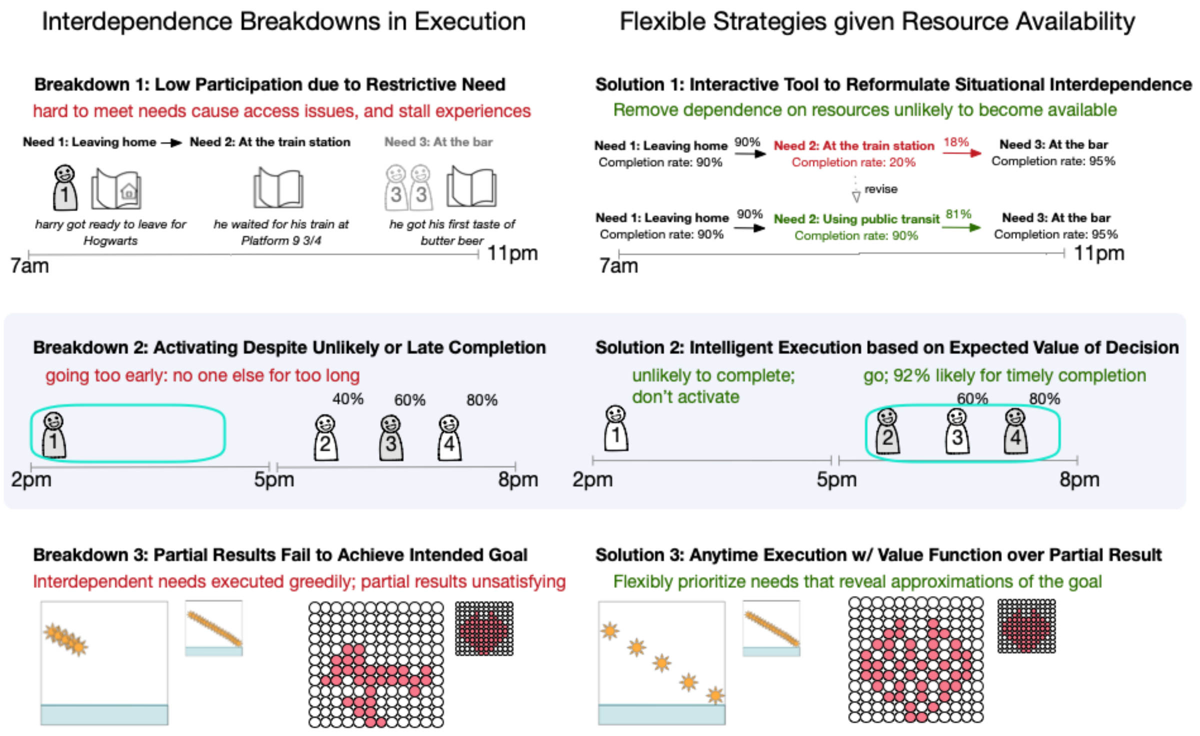

Interdependence Breakdowns in Execution
Flexible Strategies given Resource Availability
Breakdown 1: Low Participation due to Restrictive Need
hard to meet needs cause access issues, and stall experiences
Need 1: Leaving home
Need 2: At the train station
Need 3: At the bar
harry got ready to leave for Hogwarts
he waited for his train at Platform 9 3/4
he got his first taste of butter beer
7am
11pm
Solution 1: Interactive Tool to Reformulate Situational Interdependence
Remove dependence on resources unlikely to become available
Need 1: Leaving home
Completion rate: 90%
90%
Need 2: At the train station
Completion rate: 20%
18%
Need 3: At the bar
Completion rate: 95%
revise
Need 1: Leaving home
Completion rate: 90%
90%
Need 2: Using public transit
Completion rate: 90%
81%
Need 3: At the bar
Completion rate: 95%
7am
11pm
Breakdown 2: Activating Despite Unlikely or Late Completion
going too early: no one else for too long
40%
60%
80%
2pm
5pm
8pm
Solution 2: Intelligent Execution based on Expected Value of Decision
unlikely to complete; don't activate
go; 92% likely for timely completion
60%
80%
2pm
5pm
8pm
Breakdown 3: Partial Results Fail to Achieve Intended Goal
Interdependent needs executed greedily; partial results unsatisfying
Solution 3: Anytime Execution w/ Value Function over Partial Result
Flexibly prioritize needs that reveal approximations of the goal

Figure 4.1. Illustrations of the breakdowns for situational, temporal, and role interdependence; and solutions for each which flexibly reason about alternative strategies for achieving the intended interdependence given anticipated availability of interactional resources.

how breakdowns can occur for three forms of interdependence in opportunistic interactions: (1) *situational interdependence* where the social value comes from people being in shared contexts; (2) *temporal interdependence* where their social value comes from people participating around the same time when the experience is fresh (e.g., meal time within a couple hours); and (3) *role interdependence*, where the social value comes from people making unique contributions to achieve a shared goal together (e.g., pictures forming a timelapse of a sunset).

We have observed that in every interdependence breakdown, failure to consider the limited and uncertain availability of *interactional resources* (individuals capable of fulfilling interactional needs) during the structuring and execution of an opportunistic interaction is likely to result in missed opportunities and incomplete experiences that ultimately harm social connections and violate interaction norms. In what follows we show three kinds of interdependence breakdowns that result from a lack of consideration of interactional resource availability.

4.3.1. Breakdowns in Interactions with Situational Interdependence

One way to structure socially engaging opportunistic experiences is through *situational interdependence*, where people across distance participate from shared contexts to promote a sense of shared experience (e.g., slurping noodles at a ramen shop; stepping into the waves at the beach). However, one type of breakdown occurs when designers forget to consider whether people can access the required contexts for participating. As a representative example, Figure 4.1 (top left) shows the design of an experience for illustrating a story together that requires users to take a photo at a train station, even though no one in the group frequents train stations. This stalls the experience from progressing to later interactional needs that group members can complete. In previous research studies deploying OCEs [**103, 27, 73**], some deployed experiences were hard to access because their situational needs were narrowly-defined (i.e., a rare contextual overlap such as a specific kind of food like "bubble tea"), or because of regional differences in access for various members of a target population (i.e., there are many train stations in Chicago, few in Los Angeles and Phoenix).

Designers can unintentionally forget about these implicit constraints and norms because the user interfaces for designing context-aware experiences [**104**] lack support for evaluating how the likely availability of participants across regions impact inclusiveness and ease of accessing the opportunity. Ultimately, without better tool interfaces for recognizing the impact that additional constraints can have on their implicit norms, designers will be challenged to create experiences that make the best tradeoff between providing access to an experience across geographic regions and their desired vision for connecting across distance via similar situational contexts.

4.3.2. Breakdowns in Interactions with Temporal Interdependence

Another way interdependence can be structured in an OCE is through a *temporal interdependence* where their social value relies on reciprocated participation coming within a meaningful time-window (e.g., a meal time encounter within a couple of hours; sharing about weekend adventures occurring over the same weekend). However, automated systems can initiate experiences with a first participant without considering if future participants are available to complete the interaction in a timely manner, resulting in experiences going incomplete or suffering from long delays. Figure 4.1 (middle left) illustrates an example of an opportunistic experience that aims to identify an opportunity for Alice and one of her friends to virtually share a meal together across their respective cities. However, Alice may have a late lunch while the rest of her friends will have their next meals much later in the day when the experience is no longer fresh in Alice's mind. This would result in breakdowns of interaction norms around social reciprocity (i.e., where Alice is left hanging), similar to how delays in responsiveness in semi-synchronous chats are negatively interpreted [**8**]. From previous studies that have deployed opportunistic experiences [**103, 73**], users have said that “being left hanging” via an incomplete experience left them feeling disappointed and less connected.

Such breakdowns in reciprocity occur partly because designers lack the means to state their implicit expectations for the timeliness of completion to these automated systems. Moreover, the automated systems for launching an experience do not currently consider constraints on the interactional resources available to complete, nor have the ability to adapt their execution decisions to better satisfy the expectations for timely completions. A further challenge in accommodating the additional expectations for social reciprocity is that fixed decision policies set by a designer—such as always activating to eagerly promote a dyadic experience, or never activating for fear of a lack of reciprocity—will always lead to suboptimal social benefits *some of the time.* For instance, there might be more people available for a shared mealtime experience during the peak hours of lunch and dinner but fewer in the afternoon between these two times; a policy that always starts experiences will sometimes launch when there are fewer people to reciprocate participation. In summary, current automated systems for launching experiences lack effective interfaces for conveying implicit expectations for the social value of timely completions, and mechanisms for reasoning about changing interactional resource constraints and adapting their decisions to satisfy implicit expectations.

4.3.3. Breakdowns in Interactions with Interdependent Roles or Contributions

The third way interdependence can be structured is through *role interdependence*, in which individuals’ contributions must come together in a particular configuration to construct a digital artifact or achieve a collective goal. However, automated systems can fail to account for execution scenarios when there are fewer participants available to fulfill the expected roles of the collective activity. For example, consider an OCE that promotes interdependent activities through unique contributions to a collective goal, such as creating a visual artifact constructed from pieces contributed by different members in a group (i.e., a Sunset Timelapse made of snapshots at every minute of the hour before sunset time; a Heart-shaped photo mosaic). Figure 4.1 (bottom left) shows the resulting patterns from an execution strategy where needs are met whenever they can be first met, which might complete a partial set of needs, but still *fail to reveal the shape of the collective artifact that the friend group was building towards.*

The problem with this approach is that while the system has a notion of what it means to fully complete an experience, it has no notion of what a good partial result might be. Without considering ways to achieve a similar interdependent activity with fewer roles (e.g., how to adapt a story acting activity, originally written for 5 character roles, to work when only 2 characters are available), automated systems will likely fail to coordinate people's opportunistic participation to achieve satisfying artifacts or collective goals.

4.3.4. Summary of Challenges

In this section, we have illustrated three kinds of interdependence breakdowns: situational, temporal, and role. With each breakdown, we observe that structuring and executing the opportunistic interaction without accounting for or directly reasoning about the limited and uncertain *interactional resources* available is bound to lead to incomplete experiences and missed opportunities that harm social connections and violate interactional norms. In order to realize opportunistic, interdependent activities, then, we will need tools and computational mechanisms for reasoning about interactional resource availability explicitly, based on which to make decisions about when, if, and how to execute an opportunistic experience so as to preserve interdependence. This can then enable us to provide designers with tools to reformulate interaction needs to be more accessible and automated systems with computational mechanisms to strategize about and achieve timely completions and satisfying partial results during execution. In the next sections, we illustrate how we use this approach to recover interdependence for each of the breakdowns we discussed here.

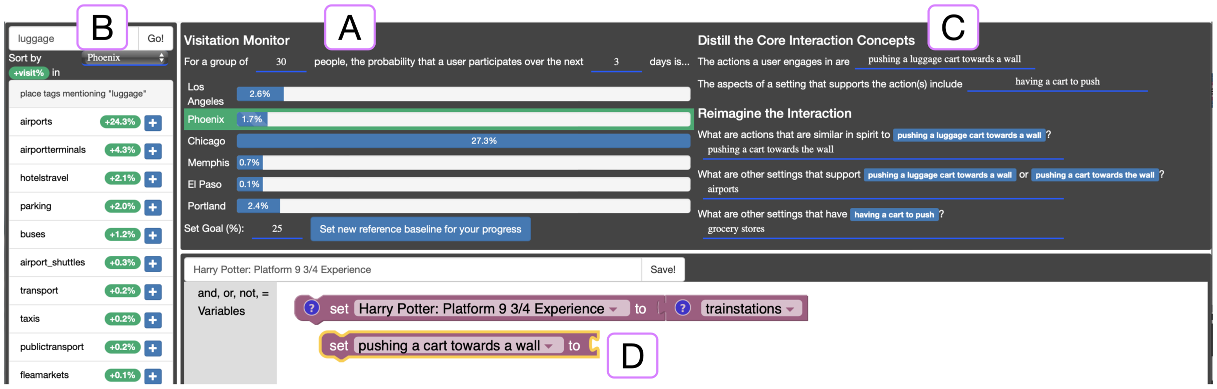
4.4. Execution Interface Layers

To address these challenges, we advance different Execution Interface Layers that augment the existing interfaces for instructing automated systems with new interfaces for reasoning about potential constraints faced in execution, recognizing and stating expectations not originally communicated, plus new computational mechanisms so systems can adapt their behaviors to account for them. In the domain of coordinating OCEs, the Execution Interface Layer supports considering the constraints of people's uncertain or limited availability to participate, recognizing and stating which types of breakdowns in interdependence should be avoided, so that opportunistic coordination engines promote the intended social benefits when executing these experiences. For example, the Execution Interface Layer can augment authoring tools for designers, so they can recognize how an experience might be inaccessible across geographic regions and reformulate their experience definition to mitigate breakdowns in situated interdependence. Furthermore, the Execution Interface Layer provides designers with constructs to convey the importance of an experience completing in a timely manner, and automated systems with decision mechanisms for adapting whether they start experiences based on whether it's likely to lead to a breakdown in timely reciprocity. This approach for adapting execution takes inspiration from flexible strategies used in computer systems operation (e.g., graceful degradation; flexible computation; anytime algorithms): in cases when there are limited resources to achieve a desired computation, it is important to reason about alternative computation strategies for achieving a goal or maintaining minimal operation, given the resources that are available. Analogously, we argue that in cases where it is clear that there are insufficient interactional resources (people in specific situations) to achieve the desired social interactions, it is important to consider alternative ways of implementing or executing opportunistic interactions to best achieve the interdependence given the interactional resources that are available.

We demonstrate this approach through tools and computational mechanisms that can help to address breakdowns in situational, temporal, and role interdependence. They include (1) a tool for reformulating hard-to-meet needs; (2) a decision-theoretic mechanism for launching experiences based on the likelihood of a timely completion; and (3) an anytime execution strategy for coordinating satisfying partial results.

4.4.1. Reformulation of Hard-to-Meet Needs

Satisfying situational interdependence in an opportunistic experience demands that designers consider the potential constraints and breakdowns in executing the experience's situational requirements across geographic regions. However, doing so is difficult with



luggage

Sort by Phoenix

+visit% in

place tags mentioning "luggage"

airports	+24.3%
airportterminals	+4.3%
hotelstravel	+2.1%
parking	+2.0%
buses	+1.2%
airport_shuttles	+0.3%
transport	+0.2%
taxis	+0.2%
publictransport	+0.2%
fleamarkets	+0.1%

Visitation Monitor

For a group of 30 people, the probability that a user participates over the next 3 days is...

Los Angeles	2.6%
Phoenix	1.7%
Chicago	27.3%
Memphis	0.7%
El Paso	0.1%
Portland	2.4%

Set Goal (%): 25

Set new reference baseline for your progress

Distill the Core Interaction Concepts

The actions a user engages in are pushing a luggage cart towards a wall

The aspects of a setting that supports the action(s) include having a cart to push

Reimagine the Interaction

What are actions that are similar in spirit to pushing a luggage cart towards a wall?

pushing a cart towards the wall

What are other settings that support pushing a luggage cart towards a wall or pushing a cart towards the wall?

airports

What are other settings that have having a cart to push?

grocery stores

Harry Potter: Platform 9 3/4 Experience

and, or, not, =

Variables

set Harry Potter: Platform 9 3/4 Experience to trainstations

set pushing a cart towards a wall to

Figure 4.2. An interactive tool so designers can reformulate an experience's situational interdependence to remove reliance on hard-to-meet interactional needs. Its key features include a monitoring tool for checking how a model of people's availability changes based on changes to the definition of the need (A); a resource-aware search tool for finding relevant context-features that would impact the visitation likelihood (B); a set of reflection prompts to help a designer distill the core interaction concepts for the experience (C); and the visual programming workspace for constructing the concept expression of the need (D).

current tools for defining situational requirements because they provide no means of simulating realistic execution across regions and identifying these breakdowns beforehand. Thus, we develop an Execution Interface Layer that reconfigures the tools for defining situational requirements, by providing designers computational models of people's availability to participate to help surface how breakdowns in access might occur across regions; the Execution Interface Layer provides additional scaffolds to help a designer in the process of reformulating their requirements for the experience to mitigate breakdowns. This approach can help designers identify access issues early in the design process, and use designer ingenuity to find alternative conceptions that will allow for more inclusive experiences that are more readily realizable in practice, based on predictions of what interactional resources will actually be available during execution. For example, Figure 4.1

(top right) shows that there is only a 20% chance that a user in a group will be at a train station, suggesting that it is likely that the experience will stall and not progress further within the specified timeframe. Realizing this issue, the designer can revise the restrictive need by creating the interaction need “using public transit,” which matches situations that are more likely to be met.

To instantiate this Human-AI Interface Layer for recognizing and stating implicit expectations for execution, we develop an interactive tool for reformulating the situational requirements for an opportunistic experience. The tool provides designers with an awareness of the anticipated availability of resources during the design process; see Figure 4.2. In this setting, our tool integrates computational models of interactional resource availability by modeling a target popluation’s availability for accessing defined situations in their daily routine based on their visitation likelihood. Using these models, we support designers actively monitoring for the likelihood of an experience completing given its current concept expression definition (Figure 4.2(A)). When the likelihood is low, the tool helps designers reformulate the concept expression using a *resource-aware search* that accounts for incremental changes in interactional resources when adding a context-feature (Figure 4.2(B)). This helps designers reflect on the accessibility of their opportunistic experience, and find ways to expand the situational requirements to make it more feasible. The reformulation tool also presents *reflection prompts* that ask designers to reflect on the core interactional requisites for the experience’s situational interdependence, so that they may consider removing reliance on surface features that hinder accessibility and finding alternative ways of realizing the core actions and qualities of the experience (Figure 4.2(C)). Similar to how, in the domain of design fixation, finding a functional schema can help to find other relevant mechanisms for achieving the function or purpose [**98**], we hypothesize that focusing on aspects of the situation that affords certain actions should help a designer for opportunistic interactions find useful alternative conceptions for interactional needs that can still promote the social value of situational interdependence.

4.4.1.1. Modeling. The Execution Interface Layer improves upon the status quo tools for defining situational requirements by integrating computational models of what interactional resources will likely be available for the current definition of an interaction need. The model represents interactional resource availability as the probability that a target population in a region could access the required situations within some number of days (e.g., the probability that someone from a group of 30 people from Phoenix visits "train stations" in the next 3 days). To calculate this probability, we can use historical visitation frequencies to understand how often a target group accesses the context in a defined interactional need. Since OCEs heavily rely on location context to define the situational interdependence, our model of interactional resources is estimated by the frequencies of visiting different location categories, trained on a dataset of global checkins from the Foursquare platform [**134**]. For the resource-aware monitoring interface, models of resource availability are computed for the current concept expression across 6 different geographic regions. For the resource-aware search interface, the incremental change in resources is computed by comparing the resource-availability increase that would result from adding the location context-feature into the concept expression.

4.4.2. Intelligent Activation of a Time-dependent Activity

Mitigating breakdowns of temporal interdependence in opportunistic interactions requires considering potential constraints in execution—specifically, the uncertainty in people's future availability—and the impact on completing an experience. We built an Execution Interface Layer to reconfigure the automated system's default behavior of always launching experiences without future regard to availability to complete them. With this Execution Interface Layer, designers can communicate their expectations (e.g., experience should be completed in the next three hours) which automated systems use to adapt their in-the -moment decisions to satisfy those expectations. We focus on dyadic interactions where, for example, this strategy could decide to pass on an opportunity to engage the first user if no one is likely to participate in the near future (leading to long delays or experiences failing to complete altogether), and later engage another user in the early evening when reciprocal participation is more likely; see Figure 4.1 (middle right).

Specifically, we develop computational models of an experience's potential execution scenarios by modeling the likelihood of another user completing it after (hypothetically) starting it. These models are used in a *decision-theoretic mechanism* that controls the launching of experiences by reasoning about the expected value (and cost) of starting an experience at the current moment. The decision-theoretic mechanism trades off the potential positive value of launching and completing the experience in a timely manner with the negative value of interactional resources not being available soon enough to complete it. When the expected value is negative, the temporal interdependence is likely to be violated in execution, and thus the decision-theoretic mechanism will choose to not launch the experience.

Our implementation of the decision-theoretic mechanism decides to activate an experience at a given hour by calculating an *expected value of activation*, which is the weighted average value of experience completion at each hour (or a large negative cost if not completed) weighted by the probability of someone completing the experience in that hour. Assuming independence between arrivals across time periods and that the value of not activating an experience is zero, we can formulate the expected value for activating an experience as follows:

$$\mathbb{E}[\text{value of activating}] = \sum_{i=1}^{N} V(C_i)P(C_i)\prod_{j=1}^{i-1} P(\overline{C_j}) \tag{4.1}$$

where $V(C_i)$ is the value (and cost) of completing within i hours after the first participant; $P(C_i)$ is the probability that someone completes after i hours; $\prod_{j=1}^{i-1} P(\overline{C_j})$ is the joint probability that no one completed in each of the hours before i.

4.4.2.1. Modeling. Designers can convey their expectations for the system, deciding to launch experiences through a value function that represents the social benefits from completing an experience in a timely manner and the costs or breakdowns if delayed. In our implementation, designers can use an intuitive parameter ω to define the desired time-window in which an experience should complete, which should correspond to their vision for the experience (e.g., $\omega = 3$ for a three hour time window for a meal time experience). This parameter is used to define a $\tanh(\omega - t)$ value function modeling the value of experience completion by hour; see Figure 4.3. This value function represents the number of hours the first user can wait and still receive the benefit of social reciprocity,

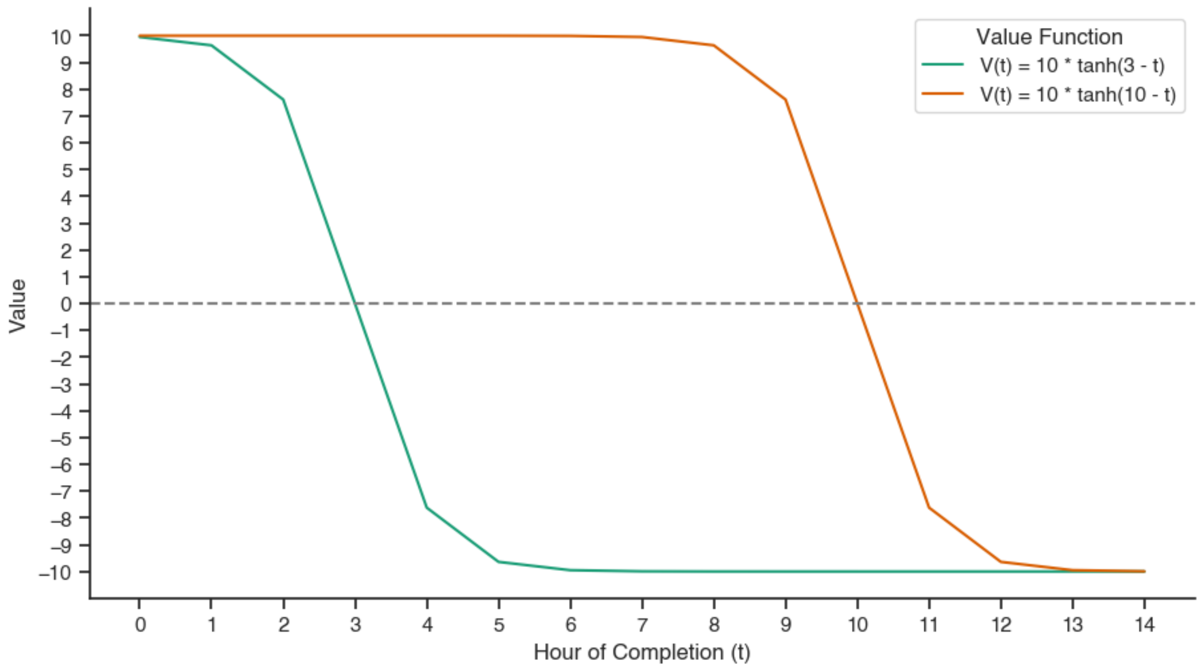


Figure 4.3. The desired temporal interdependence of different experiences can be modeled by the family of value functions $V(C_t) = tanh(\omega - t)$. For example, $\omega = 3$ defines a same-mealtime experience with completions within a 3-hour time-window while $\omega = 10$ defines a same-day experience.

after which the delay becomes too long to feel satisfying. This is used by the automated system through its decision-theoretic mechanism.

To model the likelihood of experience completion by hour, we must model people's availability to visit specific situations and participate in experiences. For example, in the case of opportunistic social experiences grounded in physical contexts, people's likelihood to visit a situation could be estimated by hourly visitation frequencies for a location category (e.g., visitation frequencies for restaurants across the hours of a day and days of the week). People's likelihood to participate when notified in the situation could be estimated through a simple probability, or through more complex models that are conditioned on different factors of personal interruptibility.

4.4.3. Anytime Execution Strategy by Assigning a Value to Partial Results

In this third technique, we focus on the problem of there being insufficient people during execution who can fulfill the roles of a collective activity. Satisfying role interdependence demands that designers communicate their ideas for how to achieve a similar interdependent activity with fewer roles, so automated systems can prioritize which roles to assign people, so as to achieve a visible approximation of the goal, or reveal the shape of a collective artifact.

Building on top of the existing opportunistic coordination systems, I developed an Execution Interface Layer that allows designers to state their ideas for how a social value through people's unique contributions can still be achieved when there are limited participants available. The Execution Interface Layer adds an additional programming interface for creators to encode value-functions that describe which interaction needs should be prioritized over others and what partial results are preferred. With this encoding, an opportunistic execution engine can coordinate people to the highest-value interaction needs that would most effectively show visible progress towards the collective goal, or reveal the shape of the collective artifacts a group is building towards. For example, Figure 4.1 (bottom right) shows the results from applying an *anytime* execution strategy that has a value function over partial results that prefers uniform coverage across the shape of the artifact (e.g., a sequential interpolation for a Sunset timelapse; filling the key edges of the heart mosaic). This ensures that the few contributed photos, or partial results, still resemble the shared artifact that the group is aiming to create.

We demonstrate this anytime execution strategy by modeling a value function in which uniform coverage across the shape of the artifact is preferred, as is the case for the Sunset

Anytime Sequential Interpolation

```
beginning = 120 // 120 min before sunset
end = 0 // 0 min before sunset
shot = 2 // snapshots every 2 min
anytimeSunsetExperience = {
 ``needs'': createSunsetNeeds(beg, end, shot)
 ``anytimeSequential'': {
 ``startingBuckets'': 3
 }
}
```

Figure 4.4. In a few lines of code, designers can specify in the OCE API definition [**103**] that the interactional needs should have a value towards sequential interpolation, and decides that the starting interpolation should divide the range of interactional needs into 3 buckets.

Timelapse experience which prefers a sequential interpolation over the range of needs. A designer of the opportunistic experience can simply define the set of needed contributions (e.g., photos of the sunset taken at 2-minute intervals for the 2 hours before the sun sets), and use this pre-defined value function for sequential interpolation across a set of ordered needs; see Figure 4.4. Based on this specification, the execution engine will treat the list of needs as sequential, and divide them into B buckets (this parameter can be adjusted, based on the expected demand); using a node-counting strategy, the anytime strategy assigns users to buckets that still have not yet received a contribution. Only when all buckets have received a contribution (i.e., satisfying the anytime interpolation) will the strategy further divide the sequential needs into 2*B sequential buckets, to then source additional contributions to create a higher fidelity sequential interpolation.

4.5. Evaluation

In summary, we argue that Execution Interface Layers can result in more desirable execution outcomes by helping people (1) consider potential constraints when executing (2) recognize and state their expectations for how they'd like automated systems to act and facilitate experiences on their behalf. In this section, we test these claims through evaluating the efficacy of the Execution Interface Layers in helping designers to realize their intended opportunistic, interdependent interactions despite automated systems operating in scenarios with uncertainty in participation. We present three evaluations for the presented Execution Interface Layers for addressing the breakdowns in situational, temporal, and role interdependence.

4.5.1. Study with Designers using Reformulation Tools

For this first evaluation, we seek to understand how our approach can resolve access issues in execution while maintaining situational interdependence of the experience. Since many access issues can be attributed to decisions during authoring, our claim is that having an interactional resource-aware authoring process will enable designers to more successfully formulate opportunistic, interdependent experiences. Thus, we conducted an formative evaluation with designers of opportunistic experiences to understand how the interactive tools for reformulation help them recognize how current interaction needs are restrictive, and reformulate the experiences to be more inclusive of what interactional resources are available.

4.5.1.1. Method. We recruited 5 designers who had prior background in either designing opportunistic collective experiences (N=3) or familiarity with using authoring tools [**104**] to define criteria for an OCE interaction opportunity using context-features. During each 60 minute study session, we first briefed designers on the problem of restrictive interactions across different cities, and their goals of redesigning experiences to broaden participation while retaining the core aspects of the interactions.

After going through a tutorial, we asked participants to author two OCEs with situational interdependence, while being mindful of how their formulations could affect access to the experience across different cities. The two OCEs were randomly selected from a pool of four OCEs which were known to be narrowly-defined (e.g., required people to be at a specific location category like a ramen restaurant) or affected by access issues due to regional differences (e.g., required people to be at a train station even though there are few in Los Angeles and Phoenix). The version of the interactive tool allowed designers to monitor anticipated availability of people across different cities in the United States: Los Angeles, Phoenix, Chicago, Memphis, El Paso, Portland. We included these cities because they reflect variations in geography, population size, and urban-ness, all of which had the potential to surface various access issues.

In the study, we aimed to observe whether designers could resolve access issues by reformulating opportunistic experiences to be more inclusive, while also retaining qualities that promote interdependence through shared situations or activities. We hypothesized that the resource-aware tools helped designers (a) recognize access issues; (b) reflect upon core aspects of the situational interdependence they wanted to maintain for the experience; and (c) expand sets of contexts that are more inclusive to the target population. Users were instructed to think-aloud as they used the authoring tool. Following the study, users completed a post-study survey that asked them to describe why the interaction was restrictive, how they reformulated the experience, and list any location categories they added which they were not confident or clear how the interaction could happen in that context. In an interview, we also asked about the parts of their process they found most effective, how the tools were helpful in their process, and what if anything was challenging when doing reformulations.

4.5.1.2. Results. Using the tools that promote resource-awareness during the construction process, designers could reformulate interactional needs that were more accessible across regions. An example reformulation for "dipping toes in the ocean (at the beach)" became stepping into something barefoot, such as water (found at beaches, lakes, marinas, or waterparks), sand (found at parks or playgrounds), or muddy nature (found at farms, hiking, and campgrounds). Through reformulating this interactional need, P2 increased participation likelihood in cities like Chicago from 2.3% to 12.5%, and expanded access to landlocked cities like Phoenix from 0.1% to 6.3%. An example reformulation for "traveling with luggage on platform 9 and 3/4s (at train stations)" became traveling with luggage (which happens at airports, buses, taxis), boarding a train (at train stations or trains), or moving around with a cart (grocery). By designing this reformulated experience, P1 addressed access issues for cities like El Paso where the original experience of train stations was unlikely to occur (from 0.7% to 36%). Had the original experiences been deployed to target populations across these cities, fewer people would have been able to engage in these experiences with situational interdependence. Through identifying these access issues early in the design process, designers can mitigate potential access issues and breakdowns by defining alternative formulations that are more accessible to the target population.

Designers used the models of execution constraints to recognize potential breakdowns, which led them to adjust the situational requirements to account for these problems. Specifically, when designers recognized that the social experience's situational requirements were hard to meet, they began searching for contexts to add to their definition for which the predicted availability of resources would increase. For example, for the ramen slurping experience that was narrowly-defined to only "ramen" restaurants, P5 used the resource-aware search to find other location contexts that served the same food of "ramen" and afforded the same action of "slurp." This helped them discover additional contexts such as places tagged as "noodles", "soup", "asian fusion", and "chinese". P5 used resource-aware search to determine which location contexts would most increase access to the opportunistic experience across the various cities. When trying to find location categories with noodles, P5 noticed that some location categories like "korean" restaurants would add negligible participation in cities like Phoenix, whereas adding "chinese" restaurants would help to increase participation probability across all cities. In this way, awareness of resource distribution across subpopulations, provided by our models, aided designers in finding alternative formulations for addressing regional differences in interactional availability.

The reflection prompts and process of construction helped designers to consider alternative formulations to achieve the situational interdependence. While some designers found the reflection questions for distilling the core interaction concepts helpful in their process (e.g., P5 said that these questions *"helps me be able to come up with more search terms, and see how relevant each would be given how I was imagining the interaction"*), others were able to brainstorm relevant interaction concepts throughout the process of construction without the help of the prompts (e.g., P3 expanded their idea of slurping by naturally thinking of related actions and other kinds of foods that involve slurping).

Additionally, the reflection prompts aided designers to come up with ingenious interaction concepts as alternative ways to promote situational interdependence. Designers felt that the process of distilling and reimagining the interaction questions helped them generate their alternative interaction concepts. P1 commented that *"the questions that asked about actions, settings that support them, and then thinking of related actions and settings created this back and forth in which you can kind of uncover relationships that you didn't really think of before."* P2 used the reimagine the interaction questions to reformulate "dipping toes into the ocean" into an experience more about "taking that first step into a new environment or experience"; for the same experience, P4 came up with a concept for an experience which united people who were "metaphorically getting their feet wet, or trying out something new", which they associated with ideas like trying an extreme sport. In either case, designer knowledge of the interaction requirements, such as its core actions and qualities of the situation, helps to uncover ingenious ways to expand the interaction need but that appropriately maintains the intended situational interdependence.

From there, the resource-aware search tool was then helpful for designers to test which of their interaction concepts would reduce the dependence on unavailable interactional resources, thereby increasing predicted access and participation. For example, P1 was trying to express an idea for stepping into sand, and *"originally thought desert was a legitimate place that would expand for people in Arizona."* But after seeing in the search tool that *"parks may be more common,"* P1 decided that they *"should really press on with* *this sandbox idea."* P1 said, *"the [resource-aware] search tool helped me think about the experiences that would realistically happen more often, and what I should actually spend most of my time making experiences for."* In this way, integrating models of interactional resource availability into the tools helped designers reason about which of their alternative formulations for promoting similar situational interdependence was more realizable in practice.

4.5.2. Simulation Study of Interface Layer to Preserve Temporal Interdependence in Execution

To show how we can maintain temporal interdependence when running social experiences, we conduct simulation experiments to compare our "intelligent activation" strategy to an "always activate" baseline strategy that starts opportunistic experiences without regard to the future availability of interactional resources to complete them. Our experiments show the benefit of approaches that reason about and strategize around the uncertain availability of interactional resources when initiating an interaction, and also provide intuition about how our intelligent activation strategy makes decisions under changing levels of interactional resource availability and the costs of delayed completions.

4.5.2.1. Methods and Models. We conduct our experiments in the context of facilitating meal time experiences between two people, for which a designer wants to promote temporal interdependence by having the experience complete around a meal time. We consider two desired experiences: one in which the experience should complete within the same mealtime, and one in which the experience can be completed anytime during the same day.

To model the likelihood of people being available to complete an experience (i.e., the availability of interactional resources), we model the joint probability of being at a restaurant and the probability of actually participating when in the location. For likelihood of visiting, we extract hourly visitation data for restaurants from Google Maps [**44**] that correspond to the two distributions we use for the experiment: (1) restaurants open for lunch and early dinner to model a *peak during midday* distribution; and (2) restaurants popular for lunch and dinner to model a *two meal time peaks* distribution. We then vary the likelihood of actually participating experimentally, as a way of simulating varying levels of interactional resources.

We model the temporal interdependence that should be achieved through a value function describing the positive value (benefits) of completing experiences in a timely manner, and the negative value (costs) of delays or incompletions. We specify a fixed positive value for completing the experience (fixed at 10), which decays over time and negates if the experience is not reciprocated within a specified time window. We use the same function $\tanh(\omega - t)$ to model this decay for two parameterizations of ω: $\omega = 3$ to represent meal-time experiences that should finish in the same meal (in a 3 hour window) and $\omega = 8$ for a meal completing anytime across the same day (within 8 hours); see Figure 4.3. The decision-theoretic mechanism then uses this value function in its computation of expected value, where a negative expected value indicates temporal interdependence would likely be violated, and a positive expected value means we can maintain it, and thus should launch the experience.

Using these models, we conduct three simulation experiments:

(1) In experiment 1, we study the benefit of Intelligent Activation over the Always Activate baseline at varying levels of interactional resource availability by varying the likelihood that a person participates in a situation from 10% to 100%. We model a uniform distribution of arrivals over all hours of the day for simplicity, and a value function that encodes a same meal-time experience (a 3 hour acceptable window).

(2) In experiment 2, we show how experience activation decisions are made given changes in interactional resource availability throughout a day. We use the *peak during midday* distribution of people's availability, fix the likelihood that users participate at 30%, and use a value function that encodes a same-day meal experience (a 8 hour acceptable window).

(3) In experiment 3, we show the effects of the changing costs of delays (i.e., time-window of acceptable completions ω) on experience activation decisions by comparing decisions made with respect to the two value functions we consider (same meal vs. same day) while using the *two meal time peaks* distribution of availability.

4.5.2.2. Measures and Analysis. Value across our three experiments measures whether temporal interdependence was achieved in execution. A positive value means that experiences completed within the relevant time-window, and negative value represents the cost of not completing in a timely manner which is associated with breakdowns in interactional norms like social reciprocity. To show the benefit of the intelligent activation strategy over the baseline when the availability of interactional resources changes (experiment 1), we measure and compare the value achieved for each strategy averaged over many decisions.

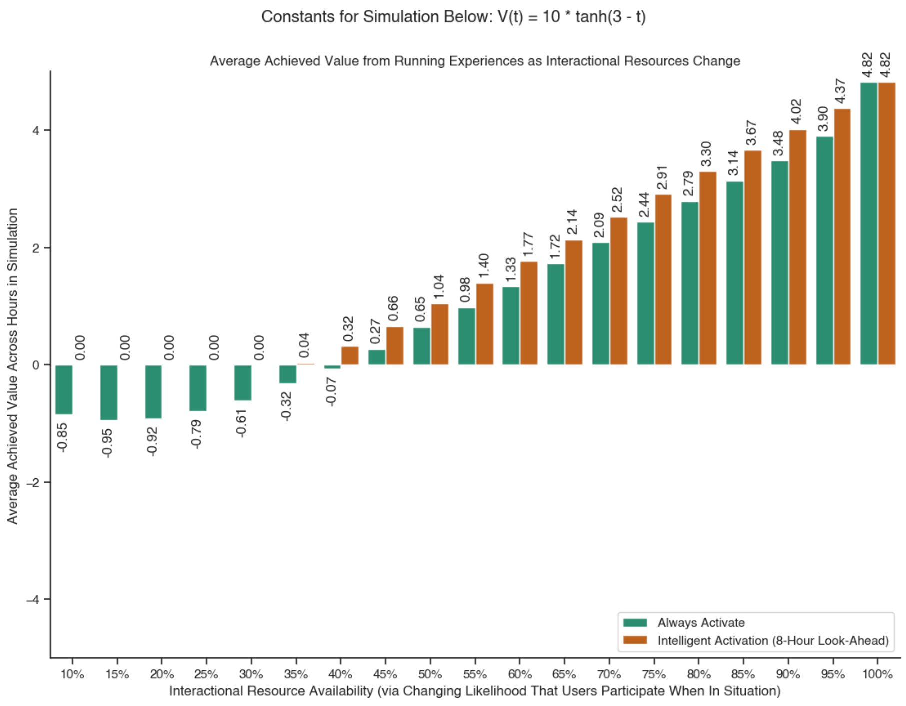


Figure 4.5. As resources become less available, the baseline strategy that starts experiences indiscriminately without considering the future rapidly drops in achieved value. At 65% and below, the times it makes costly decisions outweigh all the times that activating results in a (lucky) completion. In contrast, the intelligent strategy can be judicious about decisions as resources become less available, where all the times it decides to activate lead to positive value. Only at 30% and below does the intelligent achieve zero-value by deciding to not activate.

To provide intuition about how decisions are made as availability of resources changes throughout a day (experiment 2), we measure the expected value of activating which is used by the intelligent strategy, and compare the achieved value for decisions for both policies. To provide intuition about how decisions are made when costs of delays change (experiment 3), we investigate a decision at a single moment, plot the value for a potential completion across the hours of the day, and show how the integration of potential value across is simply the expected value for activating at this moment.

4.5.2.3. Results. When initiating experiences, the Intelligent Activation strategy outperformed the Always Activate strategy regardless of how constrained interactional resources were; see Figure 4.5. At lower levels of interactional resource, Intelligent Activation achieves zero-value by never activating experiences since they are unlikely to complete. Meanwhile, the baseline strategy incurs many costly delays or incompletions which result in negative value. At the highest level of availability of interactional resources (100% of people participating), both strategies achieve the same positive value since there is an abundance of people who visit situations and are available to complete experiences. In short, systems that can reason about when to start social experiences based on the availability of interactional resources can better maintain the experiences' temporal interdependence.

As interactional resources vary throughout the day, the Intelligent Activation strategy only begins experiences when there is enough upcoming resource availability. Figure 4.6 (Row 1) shows as we move into hours past the peak during midday, the model estimates that the expected value of activating would result in negative value achieved (Row 2) due to scenarios where costly delays or incompletions are likely (i.e., failing to have temporal interdependence). As a result, Intelligent Activation does not activate in those hours whereas the Always Activate strategy does, incurring negative value since someone is not there to reciprocate the participation in time; see Figure 4.6 (Row 3). In particular, Intelligent Activation initiated experiences in the morning (10 AM) when there were fewer interactional resources to start experiences, but more *upcoming* available in the near future to complete experiences. In contrast the strategy stops initiating during late lunch (2 PM) even though there is more availability for experiences at that moment because there

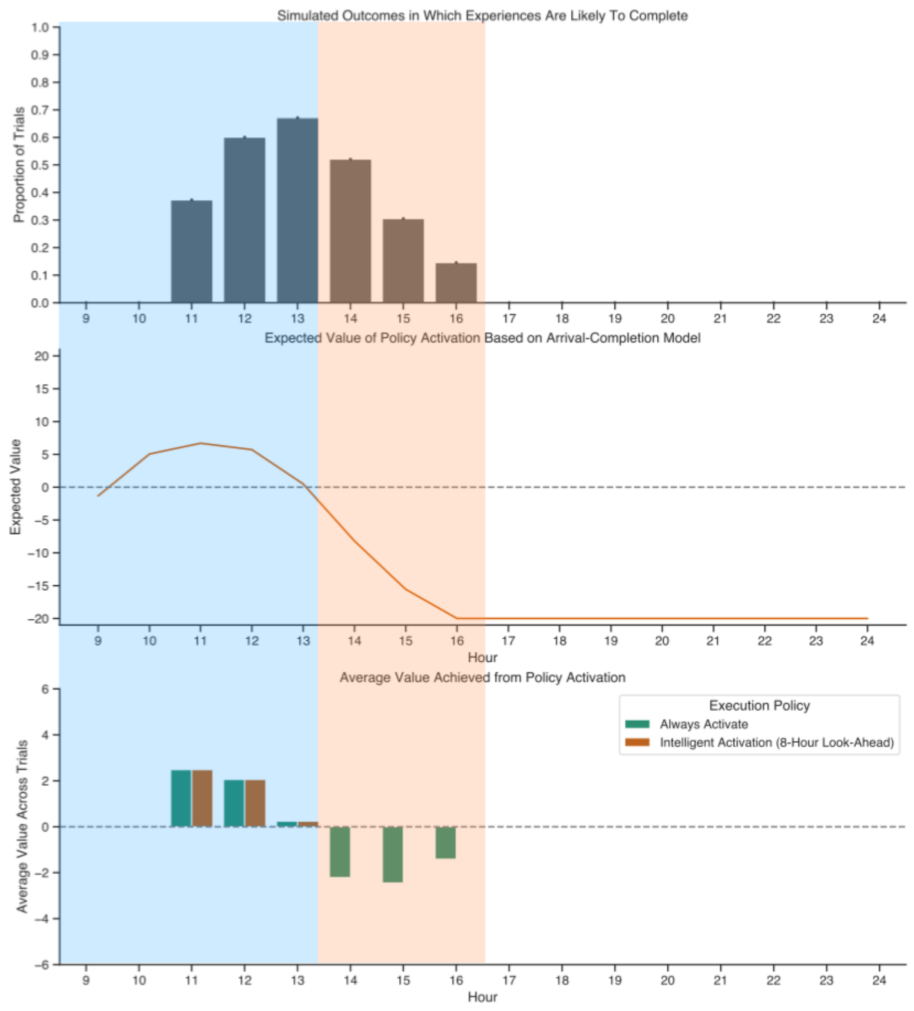


Figure 4.6. The Intelligent Activation strategy can maintain temporal interdependence by adapting its decisions as the future availability of people changes throughout the day. When the time is before or at the start of the midday rush (blue region), the Intelligent Activation reasons that expected value for activating is positive (achieving temporal interdependence), and thus starts experiences only when they are likely to complete. However, since there is limited future availability after midday (red region), the intelligent strategy decides not to activate since the expected value will be negative, or that activating will more likely lead to costly delays or incompletions.

are few resources in the future to complete them. This indicates that reasoning about *future* availability–instead of current availability–is what allows the Intelligent Activation strategy to effectively decide when to activate experiences that are likely to have timely completions.

When the time window for desired temporal interdependence is tightened ($\omega = 3$ for meal-time experience), more resources in the smaller window are needed for Intelligent Activation to begin experiences; see Figure 4.7, left. This matches our intuition about the decision moment in focus: at the end of lunch time, there are few people who will likely become available in the acceptable time frame, warranting a decision to not activate an experience to avoid potentially breaking temporal interdependence. Taken together, these results show how systems can be designed to preserve a designer's intended temporal interdependence by determining when to launch experiences based on the future availability of interactional resources.

4.5.3. Deployment Study of Anytime Execution Strategy

For our third evaluation, we study how automated systems can coordinate to achieve role interdependence despite uncertainty in the availability of opportunistic participation to complete a full experience. We deployed two versions of a Sunset Timelapse experience, one using an *anytime execution strategy* that values sequentially interpolated results, and one using a *baseline execution strategy* that is agnostic to the value of partial results and will immediately meet any unmet need that can be fulfilled as soon as someone may be available to fulfill it. We hypothesize that despite insufficient interactional resources to complete the full experience, the anytime execution strategy will coordinate opportunistic

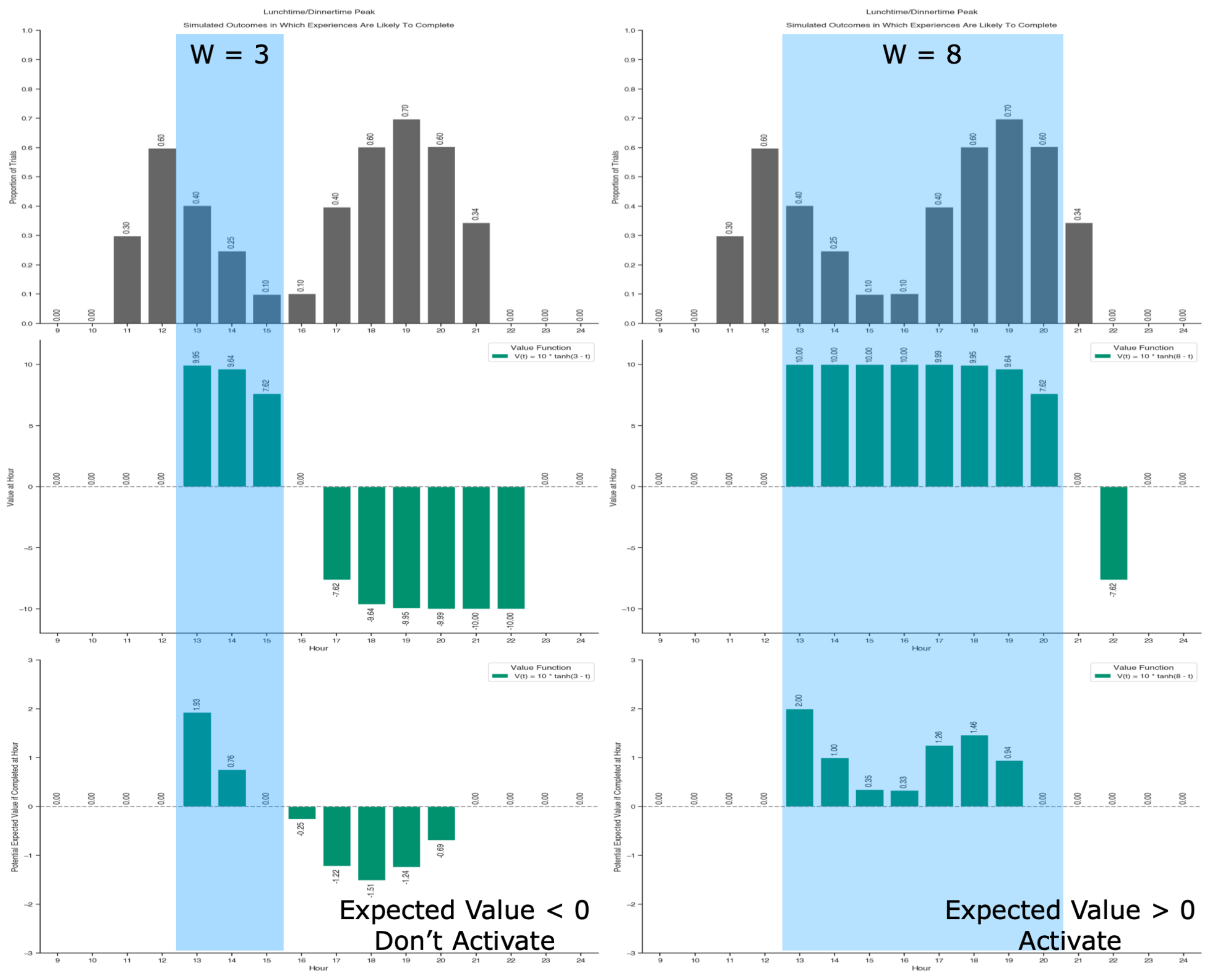


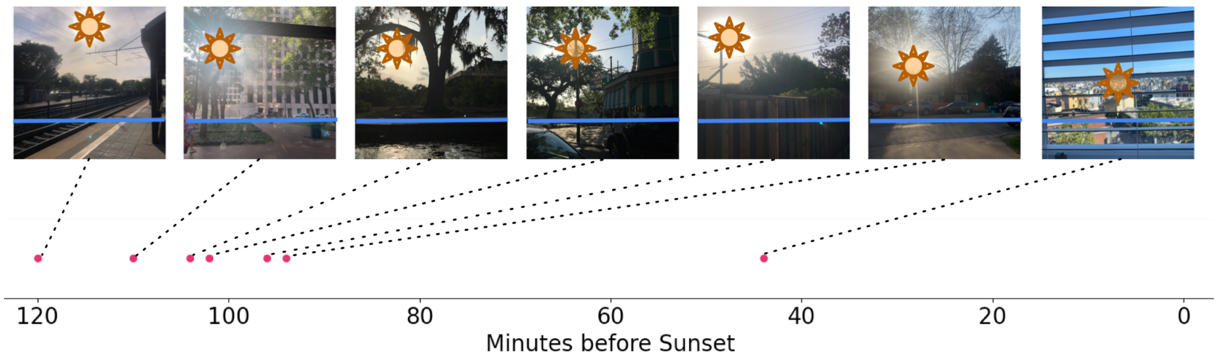
Figure 4.7. We consider an intelligent strategy that is making a decision at hour 13 (after the lunch peak). In the case where costs of delays become negative after 3 hours (same meal experience), the strategy does not activate since there are less interactional resources available within this 3 hour window to complete in time. In the case where costs of delays become negative after 8 hours (same day experience), the strategy activates since throughout this 8 hour window there are more interactional resources available, including people who will become available during dinner time.

participation to form partial results that have the shape of the intended, shared artifact, thereby promoting the social benefits from achieving role interdependence.

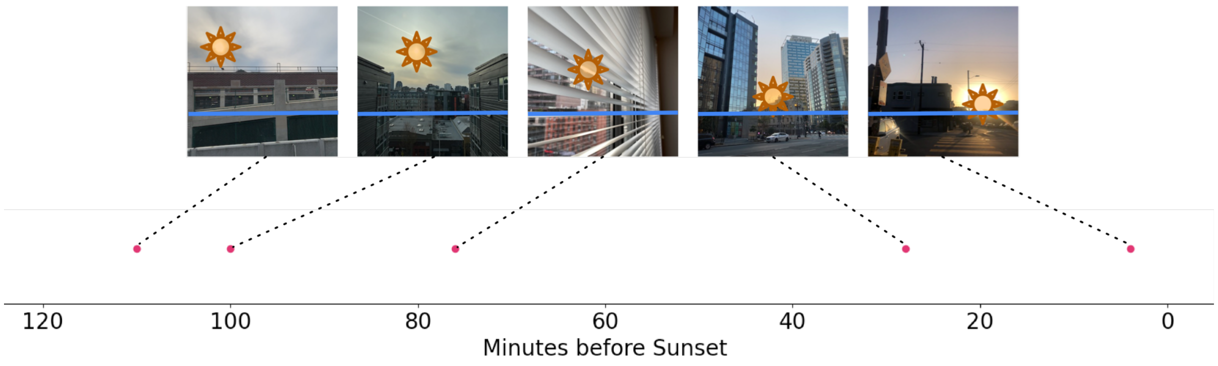
4.5.3.1. Deployment Prototype. Both versions of the Sunset Timelapse experience define 60 interactional needs, which represent snapshots to be taken at 2-minute intervals between 120 minutes before sunset until 2 minutes after the sun sets. The anytime version implemented an execution strategy which placed a priority on first fulfilling a 3-frame interpolation across the duration (i.e., 3 buckets), after which it attempts to fulfill a 6-frame sequential interpolation, and so-on. The baseline version used an execution strategy that was indifferent to ordering, by sourcing contributions for whichever remaining need that a participant first becomes available for.

4.5.3.2. Study Setup and Measures. We deployed this sunset experience to participants who were alumni members of a college organization and who were interested in connecting with one another. Similar to previous deployment studies of opportunistic experiences [**103**], we chose this college organization population because they are an instance of a larger challenge facing friends and family who are typically living physically distant but who don't always find opportunities to actively engage with one another.

We recruited 8 participants in total, which meant that our deployment conditions represented the limited interactional resource problem since there were a limited number of people available to fulfill all 60 interactional needs. Participants were distributed across three different timezones in the United States. We used a within-subjects design, where the group of participants were given the anytime version of the experience for one week, and the baseline version the next week. By testing with the same group of participants



(a) In this Sunset Timelapse Artifact sourced using the Baseline Strategy, the result has the sun barely setting, with many snapshots taken during the beginning buckets.



(b) In this Sunset Timelapse Artifact sourced using the Anytime Strategy, the results capture the whole process of the sun setting, where contributions are made more uniformly

Figure 4.8

and assuming that their availability to contribute on a given week stays the same, we try to control for the availability of interactional resources across deployment conditions.

We triangulate our results across two sources of data. First, we look at the final artifact to understand whether it shows the sunsetting, which is evidence that a partial set of interdependent contributions achieve the shape of the desired artifact. Second, we visualize how needs were fulfilled for each execution strategy across the timelapse range, to understand if the anytime strategy can more effectively source partial results with uniform coverage across the timelapse.

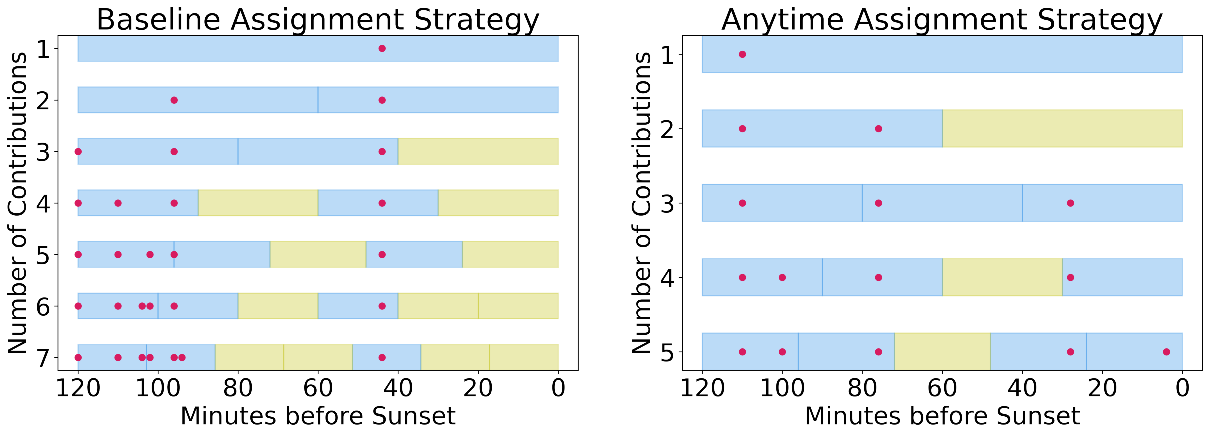


Figure 4.9. Across different stages of partial contributions, we show which needs were contributed to for the anytime strategy that assigns higher value to a sequential interpolation (right) versus the baseline strategy that is indifferent to utility (left). Additionally, the colored rectangles visualize the intervals in which coverage was successfully achieved (blue) vs. not achieved (yellow).

4.5.3.3. Results. The anytime execution strategy better captured a timelapse of the sun fully setting as compared to baseline version; see Figure 4.8b vs. Figure 4.8a. The baseline strategy had 7 interactional resources available (users who contributed snapshots of the sunset), while the anytime strategy had 5 available. Despite having more interactional resources available, the baseline strategy resulted in a view that has the sun barely setting. A majority of the contributions were made at a similar time point in the beginning when the sun was high above the horizon, with no contributions that show the sun setting close to the horizon. In contrast, for the anytime strategy, the produced artifact shows the whole process of sunsetting, from a high point above through to the finish with the sun close to the horizon. These results indicate that by being sensitive to the value of partial results, the anytime strategy can prioritize roles which interdependently reveal the shape of the shared goal, even when lacking interactional resources to achieve the full set of needs.

The anytime strategy can coordinate the limited interactional resources that are available to approximate the shape of the sunset, because it prioritizes partial results with uniform coverage over the sequential range of needs. Figure 4.9 visualizes the distribution of contributions across the sunset needs timeline after each contribution. We see that the baseline strategy results in clumps of contributions at the early stages of the timeline; this is because the baseline strategy will assign users to interactional needs that they are immediately available for (for instance, someone available at all times would still have been delivered the earliest photo need not yet fulfilled). In contrast, the anytime strategy results in more uniform coverage because it waits to assign users to contribute to sunset needs in buckets that are not yet full, resulting in a smooth interpolation after three photos before it expands to collect other photos to fulfill unmet needs across a larger number of buckets. This demonstrates that by having a notion of what partial results are better while the activity has yet to complete, the anytime strategy helped the people who engaged with it have the satisfaction of seeing their digital artifact actually reveal its shape (versus not at all in the baseline), and at any stage of participation.

4.5.4. Study Takeaways

4.5.4.1. Preserving Situational Interdependence through Reformulating Hard-to-Meet Needs. Providing social experience designers with computational models of people's availability allowed them to recognize and correct potential issues with users being unable to access location contexts used for their experiences. Of note is how designers were able to exercise their ingenuity to reformulate experiences in ways that automated systems could not on their own. For instance, designers were able to distill the core interactions of the experience and consider alternate, more accessible situations that could support those interactions using a mixture of reflection prompts and tools that showed the change in participation likelihood. This shows that designers could use the Execution Interface Layer's scaffolds to form their tacit ideas for what they really intended in the design of the shared experience, allowing them refine the experience and requirements to preserve these ideas while accommodating the need for broader access to the opportunity across geographic contexts.

4.5.4.2. Meeting Temporal Interdependence through Intelligent Activation.

With the Execution Interface Layer, the opportunistic coordination system gained new abilities to reason about the anticipated availability of interactional resources and choose when to initiate experiences based on how likely they are to complete. Regardless of how constrained interactional resources were, we saw that our Intelligent Activation policy would only ever initiate experiences when there was enough future availability of interactional resources. Moreover, a key benefit of the Execution Interface Layer was allowing designers to specify the the temporal interdependence of their experiences (e.g., within the same meal, or all day) that prior systems without the Execution Interface Layer's computational mechanism would have no way to maintain. In other words, an Execution Interface Layer that augments execution engines with models of future interactional resource availability allows them to reduce potential incompletions of experiences and also maintain aspects of temporal interdependence that a designer may want for a social experience.

4.5.4.3. Satisfying Role Interdependence by Communicating the Value of Partial Results. Finally, our findings showed that Execution Interface Layers can allow people to impart systems with an understanding of desired partial results of a social experience. These systems, in turn, can generate artifacts from people's contributions that still resemble the shape and feel of the broader collective goal despite limited interactional resources. The baseline execution engines, which were unaware of what these partial results can look like, would solicit contributions without regard to how those contributions progress the social experience towards the collective goal, resulting in many pictures of the sun early in the sunset but little to none later on. In contrast, the anytime execution engine—augmented by a type of Execution Interface Layer—was able to produce an artifact that still resembled a timelapse of a setting sun, even though it had fewer interactional resources available. In practice, such resource constraints are likely to arise even when designers have considered ways to make social experiences more accessible. Thus, these findings show the importance of also developing ways to convey what partial results of a collective goal may look like to systems so that they can preserve role interdependence when executing experiences.

4.6. Discussion

Having conducted studies on three types of Execution Interface Layers, we revisit the key conceptual ideas of how an Execution Interface Layer can augment the usage and communication with an pre-existing automated system, and discuss how they can generally support individuals in recognizing and stating implicit expectations to automated systems.

4.6.1. Consider How Execution Constraints Impact Implicit Expectations and Norms

In this chapter, we illustrated three practical breakdowns in interactional norms when executing opportunistic, interdependent experiences. Far from being three separate problems, they could all be understood by the mistake of not considering execution constraints—such as people's availability to be in situations to participate in these experiences—and not stating to automated systems how people's implicit expectations and norms can be accommodated. Thus, addressing the practical breakdowns in interdependence required developing tools and mechanisms provide an awareness of how uncertainty or limitations in people's availability to participate—can impact interactional values, which then enabled designers and automated systems to find alternatives to realizing the experience based on where and when resources were likely to be available, or that worked with limitations in resource availability. These tools and mechanisms defined an Execution Interface Layer built on top of existing AI capabilities for coordinating opportunistic experiences, enabling designers and automated systems to communicate about the impact of execution constraints and the strategies to respect implicit expectations in light of them.

Our techniques employed two ways of recognizing and reasoning about the implicit human constraint of end-users being available to participate in these opportunistic experiences. The first way explicitly modeled the uncertainty of interactional resources needed to achieve the intended interdependence. A model of uncertain availability allowed designers to evaluate whether their contextual trigger was too restrictive across geographic regions. This model also supported them in finding how to adjust their original goals in light of these implicit constraints via a resource-aware search tool that helped them find additional context-detectors that would increase participation likelihood. This gave them ways to reformulate their experience's contextual requirements to be more accessible. A model of uncertain availability across hours was used by automated systems to compute an expected value for activating, allowing them to choose to only activate when expectations for timely completion would be satisfied. In this way, the computational models of implicit constraints powered a decision-theoretic mechanism that reconfigured the ways automated systems could adapt their behaviors moment-to-moment to accommodate designer's stated expectations. Rather than modeling the implicit constraints, the second way assumes that these constraints prevent the full result from being achieved as originally intended and proposes anytime execution strategies for achieving satisfying partial results given these constraints. This Execution Interface Layer provided a simple construct in the programming API for designers to state their implicit ideas for what configurations of partial results were more preferred to promote the intended role interdependence. Already, this allowed the automated systems to adjust which roles people are assigned to, thereby coordinating partial results that worked with constraints of limited resources while still satisfying the designers expectations for role interdependence.

4.6.2. Responsibility of Humans and Automated Systems for Successful Execution

To promote the opportunistic experiences a designer has envisioned, they must rely on automated systems that recognize when people are available for interactional needs and decide how to execute to coordinate their participation in experiences. However, as we illustrate through the breakdowns in interactional norms of these opportunistic, interdependent experiences, accounting for implicit constraints and conveying unstated expectations is critical when handing off to automated systems The Execution Interface Layer is meant to improve the interface between people's expectations for social interactions and the AI system's execution behaviors when faced with the constraints of end-users' uncertain availability to be in situations to participate.

Amongst the different techniques offered in this Execution Interface Layer, the decision to accommodate implicit expectations falls under two categories of responsibility: the decision can be made by AI systems at execution time, or the decision can be made by designers themselves. When AI systems are entrusted to make decisions for adapting aspects of the experience or execution, this requires the designer to more explicitly encode their understanding of interactional value and norms into a representation that a computational mechanism can use to reason over and act upon. Doing so empowers designers to configure how their experiences run when deployed given variations in the constraints, while ultimately giving systems a greater level of autonomy to decide how to satisfy human expectations. For example, designers could describe their implicit expectations for timely completion through through parameter ω that determined when the value of completion is no longer positive, which the decision-theoretic mechanism then uses to decide when experiences should be launched based on whether that expectation is likely to be satisfied, as determined by the expected value function. This shows how an Execution Interface Layer can reconfigure an existing automated system to interpret a human's stated expectations and reason about the variations in uncertain constraints, to allow it to make decisions across different moments that will best satisfy human expectations.

The second category of responsibility shows when designers must ultimately use their implicit understanding of expectations to decide what goals or behaviors should be adjusted in response to the execution constraints. The tool for reformulating the situational requirements, and the anytime execution engine for achieving satisfying partial results are both examples of how designers can convey, prior to deployment, exactly how they want execution to be changed to achieve a balance between the originally stated experience and the possibly limited availability of people to participate. For example, the reformulation tool addresses challenges designers face in knowing how to accommodate these constraints. It was important to note that systems could not automatically make these accommodations: while the system, aided by models of resource availability, could suggest how the stated situational requirements could be changed to be more inclusive, some of their suggested reformulations could expand the situation so it no longer recreates a feeling of shared experience through similar situations. More generally, while machines can suggest ways to adjust to the constraints, people need to be in the loop to reason about their implicit expectations and goals—like how to promote the essence of their envisioned experience—that machine's don't have an explicit understanding of.

4.7. Conclusion

In conclusion, entrusting AI systems to proactively facilitate human experiences hinges on recognizing and accommodating implicit human constraints and expectations. Neglecting these factors, as illustrated by context-aware AI systems coordinating opportunistic social interactions, can lead to disappointing outcomes and undermine social benefits. The proposed Execution Interface Layer offers a solution by empowering users to articulate interaction norms explicitly and by equipping AI systems to adapt behaviors based on implicit expectations. This Human-AI Interface Layer facilitates two-way communication. It empowers AI systems to assist users in recognizing how their goals might be affected by implicit constraints, promoting thoughtful reformulation of experiences. Conversely, it equips users with intuitive constructs to articulate their implicit expectations, enabling the AI system to incorporate these considerations into decision-making processes. Ultimately, the success of AI systems in fostering meaningful interactions and experiences hinges on their ability to adapt and respond to implicit human constraints and expectations. As we move forward, embracing this approach will pave the way for AI systems to fulfill their promise as facilitators of enriched human pursuits, enhancing end-users' lives through thoughtful and accommodating actions.

CHAPTER 5

Discussion

In this chapter, we revisit the core contributions of Human-AI Interfaces and their significance in addressing challenges in communicating human intentions to AI collaborators and AI agents; discuss the design principles of Human-AI Interface Layers; and provide advice on how to conduct research in this area. Furthermore, we discuss the importance of generalizing and extending Human-AI Interface Layers with the emergence of new AI capabilities like Large Language Models. Finally, we highlight how Human-AI Interface Layers are a promising approach to promote human engagement in meaningful pursuits where craftsmanship, self-expression, and mastery should be preserved.

5.1. Human-AI Interface Layers Revisited

AI capabilities are being infused into systems and user applications that touch many aspects of human life. The uses of AI and automated systems have expanded, from common and repetitive tasks to more personalized and custom needs. As such, AI systems have aimed to be more responsive and adaptable to a user's or creator's inputs and needs. In support of these personalized uses, many AI and automated systems now provide an interface with some means for users to provide inputs to influence the AI operation or output—such as the content it produces, the situations it recognizes, or the contributions it coordinates. Examples from our work include a musical AI generating different music depending on the starting notes provided, to the context-aware AI agent deciding when and for whom to send digital experiences based on the situation the agent has been instructed to trigger upon.

More and more, tool and application builders are exploring how to use these more interactive AI capabilities to assist with meaningful human endeavors, across the domains of human creativity and social connection. These domains require awareness and sensitivity to a human's subjective and personal knowledge which is derived from the user's emotions, cultural understanding, and personal experiences. However, many available AI capabilities are trained upon predefined labels and built on pre-programmed knowledge, operating in a general-purpose and more objective manner. Critically, they have a limited understanding of a user's personal and subjective ideas.

These AI systems need to be imparted with human ideas about emotions, social situations, or social norms–so they can assist and facilitate a creator's intent appropriately. However, a core problem is that some existing AIs, which are supposed to be interactive and configurable for creators, are instead still challenging to interface with. First, the AI's available constructs—the current "language" the AI can receive as input—do not match the human ideas a creator needs to communicate. Second, these AI interfaces assume individuals can comprehensively specify their inputs upfront, thus placing the entire burden on the user to form, evaluate, and clarify their ideas and communication strategies. When developers build AI-powered tools and applications with user interfaces that mirror these limited AI interfaces, these problems for communicating ideas and intentions persist.

In this dissertation, I introduce a crucial layer within the stack of an AI-powered application, called the human-AI interfacing layer. The primary objective is to enhance the communication interface between individuals and existing AI capabilities. A Human-AI Interface Layer can expose intuitive and actionable constructs for adjusting how pre-existing AIs work to support the more straightforward articulation of ideas. Furthemore, a Human-AI Interface Layer can augment the AI's interface to support incremental forming and refining of meaning, avoiding the failure of expecting comprehensive communication upfront. Together, Human-AI Interface Layers empower creators and users to make the most of an AI's inherent capabilities despite its limited interfaces for understanding human ideas, thereby assisting individuals in realizing their personal vision for an artifact or experience.

I have created human-AI interfacing layers for two domains where an understanding and sensitivity to a human's personal and subjective emotions, ideas, and knowledge is crucial. The first domain is music composition with generative AI tools. A novice composer's interest lies beyond using AI to generate coherent-sounding songs; most importantly, they desire to create a song that captures their creative ideas about how different musical elements can evoke their envisioned emotions. A Human-AI Interface Layer supports their communication of personal ideas to exercise creative control when using generative AI to co-create a song. The second domain is creating convenient opportunities to have meaningful social interactions at a distance, enabled by proactive context-aware AI systems. For example, to effectively foster coincidental moments to share a similar experience between people across different cities, context-aware systems need knowledge about situations that would recreate the specific vision of a shared experience a designer has in mind.

I have demonstrated how interfacing layers can address three types of interfacing challenges that arose due to limitations in the default user interfaces for using an AI's capability. First, when an AI tool's user interface only affords indirect influence over coarse-grained objects, such as fully-completed generated artifacts, people can struggle to impart their fine-grained and opinionated ideas into an AI's outputs. As such, I have developed a *Steering Interface Layer* that allows people to partition an AI's outputs into semantic chunks and constrain what the AI generates based on several semantically-meaningful dimensions. My studies have found that partitioning the AI's outputs allowed people to work with manageable chunks at a time, allowing them to evaluate and intervene in the AI's generation. Using the mid-level constructs for requesting how the AI should generate outputs, creators could impart their ideas for how the generated artifact should take shape.

Second, when a person's overarching goals for their AI-enabled creation are conceptually-rich and sometimes vague, they can struggle to translate these high-level concepts to the available constructs for configuring an AI system. To address this, I have developed an Expression Interface Layer, consisting of a visual workspace with cognitive support tools, that helps people with the process of translating their high-level concepts into a representation composed of the AI's constructs. My studies have found that cognitive support tools helped designers form an expansive set of concepts that spanned different facts of their high-level ideas. Moreover, the cognitive support helped in finding AI constructs that they may have missed and in verifying when constructs were not operating as intended. Together, designers used the Expression Interface Layer to encode a richer and more precise expression of their goal into a representation an AI system can effectively use.

Third, when an individual provides instructions to an automated system to fulfill, the automated system can fail to reveal constraints it might face that could cause breakdowns in the individual's tacit expectations. As such, a person may forget to communicate their implicit expectations, leading to AI systems that do not know how to appropriately adjust to these constraints. To resolve this, I developed an *Execution Interface Layer* for recognizing execution constraints and stating implicit expectations for how to adjust to these constraints. This Execution Interface Layer provides models to simulate execution to help an individual recognize potential breakdowns before they occur. Made aware of the potential breakdowns due to constraints, the person can use scaffolds for deciding reformulations of their instructions, which would mitigate these breakdowns; and allow them to state their expectations via a new mechanism, allowing AI systems to adapt decisions when facing constraints to satisfy an individual's expectations. My studies of these techniques have demonstrated that Execution Interface Layers can mitigate disappointing breakdowns by communicating about execution constraints and implicit expectations, before handoff to an automated system.

5.2. Design Principles of Human-AI Interface Layers

The Human-AI Interface Layer advances two key principles for enabling people to convey and impart their ideas to an AI system. People need a means to communicate their ideas, which are both intuitive for them and actionable by the AI system. And people need to engage in a back-and-forth process to form and clarify the full extent of ideas for the artifact or experience they are trying to create with the AI system. In what follows, we reflect on the advantages of creating an interfacing layer that embodies each of these principles.

5.2.1. Constructs that are Intuitive for People and Actionable by an AI system

Any person using an AI capability to realize their goal for an artifact or experience needs the means to convey their ideas and intentions. While many AIs provide in their interface some programming or control constructs, current constructs can be an insufficient modality for individuals to communicate their ideas for creative artifacts or envisioned experiences. To address this, we built Human-AI Interface Layers, sitting on top of existing AIs, to reconfigure the control or programming interface. This interface layer introduces intuitive and actionable constructs for communicating human ideas and intentions to AI systems. These constructs bridge the gap between user understanding and AI's abilities and execution.

Taking a Human-AI Interface Layer approach to this problem allowed us, as the developers of the AI-powered creation tools and applications, to use effective design and need-finding methods to understand what constructs would align with users' thinking. With a technical understanding of the AI system, we developed other computational components that complemented the existing AI capabilities. The combined AI system was built to understand and adjust its outputs and actions based on how the user has described their ideas through the construct.

The Steering and Execution Interface Layers especially exemplified the development of new user interface constructs with accompanying computational system components.

Mapping sliders to semantic dimensions of happy/sad, conventional/surprising, and similar/different supported novices' intuitive control over the qualities of the music. To make an existing AI model controllable based on these constructs, we developed a soft priors technique for adjusting the sampling distribution of the model. In the Execution Interface Layer, we developed a decision-theoretic mechanism to make the right decision about when to launch experiences given whether the decision would likely satisfy a designer's stated value for timely completion. Designers could encode the social value for various timely completion scenarios through a simple parameter. In summary, these Human-AI Interface Layers added new constructs in the front-end user interfaces and computational components that translate and act on them in the backend.

In a different light, Expression Interface Layers highlight the potential of front-end workspaces to assist users in creating intermediate semantics that bridge their overarching ideas and the AI's existing constructs. Moreover, users can leverage computational tools, such as the unlimited vocabulary search algorithm, to streamline the mapping between human concepts and detectable constructs. In contrast to the other Human-AI Interface Layers, the Expression Interface Layer shows that creators can bridge the semantic gap between their ideas and the AI's constructs when given interactive cognitive support tools. Rather than relying on tool developers to build new semantics and mechanisms to do this mapping for them, effective front-end workspaces and computational tools can help them do this bridging themselves.

5.2.2. Create Dialogues for People to Form and Articulate their Ideas, Progressively

Human-AI Interfaces facilitated the incremental formation and refinement of meaning, avoiding the failure of expecting comprehensive communication upfront. By promoting incremental processes with pauses for evaluation, adjustment, and reflection, a dialogue was formed that enabled creators to refine different pieces of their ideas and their strategies for communicating with AI systems.

The Steering Interface Layer enabled creators to incrementally communicate pieces of their intent by working on chunks of generated content. Working bit by bit enabled them to explore semantic dimensions and generate alternatives before clarifying their intended direction. The Expression Interface Layer provided a workspace for opportunistic construction, where observations and reflections spark new ideas and cause shifts in communication strategy. Cognitive support tools provided the necessary scaffolds for reflecting and expanding their ideas, foraging and discovering relevant strategies to articulate them, and recognizing and resolving misconceptions. One kind of Execution Interface Layer, which was instantiated through an interactive reformulation tool, established an iterative loop between formulating instructions and envisioning an AI system's execution under potential constraints. This loop helped them to recognize implicit expectations of theirs, and refine their instructions to best satisfy these expectations under constraints. Altogether, through reconfiguring the user interfaces for communicating with an existing AI system, Human-AI Interface Layers enable creators to progressively form, articulate, and evaluate smaller portions of their ideas. Ultimately, this enables them to more effectively impart personal and tacit knowledge to AI systems and to create aligned outputs and actions according to a creator's goals.

5.3. How to Conduct Research in Human-AI Interface Layers

If you are a researcher interested in developing Human-AI Interface Layers, you might be wondering how you might discover the goals and intentions of users, and the shortcomings in how people can use existing AI capabilities. In this section, I'll give advice about how to systematically approach the discovery of people's aspirations for using AI in their personal pursuits, as well as their challenges with effectively communicating their ideas and intentions to AI Assistants. Then, I'll reflect on my own design research methods for gaining answers to these questions.

5.3.1. Framework for Understanding In What Respect One Should Enhance Communication with AI Assistants

As a researcher of AI-powered applications for personal pursuits, one can think about three important facets of the problem: (a) people's motivations for using AI assistance, (b) the types of ideas, intentions, and expectations they needed to impart to AI collaborators and agents, and (c) the shortfalls in communicating these intentions with existing AI interfaces.

When considering how AI can assist in personal pursuits, we have to understand what personal knowledge or specific ideas users have for these pursuits. What intention and expectation does a user have for their bespoke usage, to achieve their custom goals? By the same token, what does the AI need to know to appropriately assist the user in this endeavor that is personally meaningful, and uniquely conceptualized by them?

Remember, we are trying to enhance a user's ability to communicate their bespoke goals and intentions to the AI system. Rather than designing interfaces or constructs which directly match an individual's specific goals, my approach seeks to understand the types of intentions that users have for this type of task, and consider what is challenging about communicating these types of intentions to existing AI systems. In this dissertation, I characterized three example types of intentions people can have: (1) ideas to evoke in their co-created artifact that require fine-grained and incremental control; (2) overarching ideas that are described by a collection of subconcepts and machine constructs; (3) implicit expectations, such as interactional norms that should be upheld, that they've forgotten to state regarding AI execution.

5.3.2. Reflections on my own methods and perspective

I used a variety of design-research methods to understand (a) people's motivations for using AI assistance, (b) the types of ideas, intentions, and expectations they needed to impart to AI collaborators and agents, and (c) the shortfalls in communicating these intentions with existing AI interfaces. In what follows, I'll recount the types of methods I used to understand these facets when developing Human-AI Interface Layers for the domains of music co-creation and opportunistic social connection.

Music Co-Creation: I conducted *interview studies* to understand how novice composers wanted to use generative AI to assist them in the pursuit of music composition. This laid the motivation for users wanting to engage in music making despite lacking music theory expertise, and to create a song that embodied their detailed and nuanced creative vision. By conducting *formative tests* with novice composers using an existing generative AI music tool, I was able to understand the types of fine-grained elements composers wanted to control when shaping a song to match their creative vision, and what was challenging about using the conventional AI interface to generating the completed artifact at once.

Opportunistic Social Connection: Collaborating with other HCI systems researchers and design researchers to invent new social technologies for connecting at distance helped me develop a clear design concept for the high-level situations that afford activities which context-aware AI agents would need to identify and facilitate. I came to deeply understand these requirements for designing and implementing contextually-triggered, opportunistic experiences—by working closely with research collaborators who were designing and prototyping these experiences [**131**], and becoming one of those designers myself [**103**].

To understand the shortfalls in communicating high-level ideas of experiences to context-aware AI systems, I conducted *pilot lab studies* of an early version of the visual programming environment tool for defining situational triggers. Studying the expressions made by participants, and doing detailed think-aloud studies to understand designers' processes, helped me clarify the bridging problem, and the three bridging challenges faced when translating a high-level idea of a situation into lower-level context-features.

Through conducting *deployment tests* of an AI agent for coordinating opportunistic social experience [**103**], I discovered that certain human expectations regarding interdependent social interactions were often violated, such as dyadic interactions not completing in a timely or satisfying manner, or situations for shared experiences being inaccessible. I came to understand that such breakdowns occurred because designers did not have a means to communicate such expectations, or could fail to recognize how their expectations might be impacted, due to variations in the anticipated availability of people across geographic regions during execution.

How might I advise an aspiring researcher in this area in light of these research reflections? A design-research approach—which can entail collaborating with design-researchers in the domain of interest, or becoming one yourself—-was a useful way to understand users' aspirations for AI assistance and the kinds of ideas or intentions they need to communicate to AI to realize their aspirations and goals. Additionally, I found that employing various fidelities of needfinding and testing helped to reveal where problems could arise when creating with AI collaborators, expressing ideas in terms that AI agents understand, and through the real-world execution of such AI agents. You too can employ such methods to discover people's motivations for employing AI assistance, the types of goals, ideas, and intentions they desire to realize with AI's help, and what challenges occur when trying to interface with existing AI systems.

5.4. Generalizing and Extending Human-AI Interface Layers

We examined three example classes of problems for communicating an individual's ideas and expectations to existing AI interfaces. The Human-AI Interface Layers developed in this dissertation addresses instantiations of these problems in the domains of music co-creation with generative AI, and intelligent, context-aware agents for facilitating opportunistic social experiences. In this section, we revisit the three Human-AI Interface Layers and discuss themes regarding their generalizability and extendability.

The first theme I will discuss is the generalizability of Human-AI Interface Layers with respect to the three classes of problems they address. We maintain that these Human-AI Interface Layers do offer a general approach which future researchers or application builders can extend beyond the instantiations of the problems we considered. We'll discuss other practical problems, and how one might consider adapting the Human-AI Interface Layer accordingly. The second theme I will discuss is how the arrival of more powerful AI capabilities, like Large Language Models (LLMs), impacts the obstacles to communicating intentions through an AI's interface, and the conceptual and technical approaches used for designing a Human-AI Interface Layer.

5.4.1. Generalizing and Extending Steering Interface Layers

Steering Interface Layers were developed to reconfigure the conventional interface of an AI which generated all the content at once, and provided few ways to control how the AI generated content. Steering Interface Layers provides several key benefits. By constraining how the AI model generated content, people could more easily find content that more closely aligned with their intentions. By giving ways to partition the generated outputs, people could assign different parts different musical qualities; build the composition incrementally, bit-by-bit; comprehend the different components that were generated; and intervene and edit the generated outputs. In this subsection, I'll discuss these benefits and discuss how current and future interfaces can promote these benefits.

5.4.1.1. Conditioning Generation According to Semantic Qualities. In our studies, the Steering Interface Layer helped composers have more control by allowing them to constrain how the generative AI model should produce content based on semantically-meaningful dimensions like happy/sad, and similar/different. Since the existing generative models for infilling music compositions did not support conditional generation, we developed a soft priors technique that adjusted the sampling distribution of this AI model without retraining.

Since then, the ML field has developed a range of techniques for adding controllability of semantically-meaningful attributes to an existing generative AI model (e.g., Plug N Play models [**30**]; Prefix Tuned Transformers [**95, 136**]). The most fundamental advance for controllability in generative AI has been the development of models that can generate content based on text-based prompts and instructions [**114, 110, 21**]. The flexibility of describing requests through language has enabled users to have vast ways to control how these models generate content. Such advances have found their way to AI models for generating other types of content, such as music and sound [**3, 28**].

However, there are new challenges in communicating intentions to AI models with the advances in conditional control via natural language prompts. Recent work has revealed that crafting prompts—the sole means to align generated outputs to one's intent—can be a challenging task. In response, HCI Researchers have developed tools, on top of text-to-image generative models, to support the iterative exploration of prompt ideas. Opal is a system for generating images for a news article that uses a structured exploration of visual concepts (e.g., tone, emotion, subject) extracted from the news article [**99**]. Promptify is an interactive system to facilitate iterative prompt exploration and refinement for a widely-used text-to-image generative model, with a unique prompt suggestion engine powered by a chat-based LLM that supports iteration of prompts through natural language instruction [**18**]. These innovative developments highlight that conditional-generation based on text prompts poses new challenges for steering of generative AI models towards one's intentions; and that the formulation and articulation of one's intentions remains important for realizing creative goals with powerful text-to-content models.

5.4.1.2. Assigning Different Parts Different Qualities. Beyond the benefits of conditional control, Steering Interface Layers also were important in allowing creators to partition generated outputs into semantically meaningful chunks. In our studies, the Steering Interface Layers were beneficial by allowing creators to control how different chunks take on different musical or emotional qualities.

Some text-to-image and text-to-music offer similar abilities to reference different parts in an artifact to control their generation (e.g., paragraphs in an essay [**110**]; different objects or characters in an image; different parts and voices in a song [**3, 28**]). However, an important difference is that end-to-end models still generate all of those parts together, rather than one at a time. We found that generating all parts at once can cause AI-induced information overload and make it difficult for users to evaluate the generated outputs. The difficulty evaluating the outputs was especially apparent in the case of music, where multiple voices playing at once may be hard to disentangle from a novice's ear. Furthermore, our studies found that generating a fully completed artifact can restrict people's engagement in defining fine-grained elements of their music composition. Other research in co-creation with generative AI corroborates our findings that when creators want to craft how different elements work together, larger models that generate entire artifacts end-to-end were less preferred; and “song writing teams resorted to breaking down their musical goals into smaller components, leveraging a wide combination of smaller generative models and recombining them in complex ways to achieve their creative goals” [**77**]. Given that powerful, text-to-content models continue to use the conventional interface of generating content end-to-end, there is a growing need to make these same models decomposable, steerable, and interpretable to support fine-grained and iterative engagement.

5.4.1.3. Building an Artifact Incrementally, or Bit-By-Bit. The other kind of interface feature that supported fine-grained control in the Steering Interface Layer was allowing users to partake in an *incremental steering* process. This process allows creators to intervene and rectify potential errors before proceeding with further generations, enabling them to progressively construct their artifacts chunk by chunk. This incremental approach served a twofold purpose: not only did it facilitate a deeper comprehension of the artifact under construction, but it also assisted individuals in identifying and addressing issues arising from the intricate interactions between different components. Moreover, this method enhanced the sense of ownership and connection that creators felt toward their creations. In the present landscape, various other generative AI tools are embracing and enhancing the concept of incremental steering. These tools constrain the AI model’s output, enabling it to generate single components or layers at a time. Generative tools that support applying different prompts to different canvas areas [**24**], or building out an image through an outfilling interaction [**119**] play a pivotal role in this process. For instance, an AI could be guided to create an image of a snowy background scene and subsequently populate it with specific characters or animals in the foreground. Such interfaces provide a means to specify which pieces or layers to operate on and how they should be generated, allowing creators to work incrementally, step-by-step, chunk-by-chunk, layer-by-layer, etc.

5.4.1.4. Tracking the progression via Editing and Repairs. Incremental steering is beneficial because it facilitates the gradual construction of outputs, a process that entails generating segments in a sequential manner. This method not only aligns with the concept of partitioning outputs and generating bit by bit, but it also introduces another core advantage: the ability to employ the current state of the artifact as a point of reference for future directions. For instance, the steering interface empowered composers to construct their creations brick by brick, enabling them to reference previous segments with the assurance that these portions remain unchanged, while only the additions introduce novelty.

Comparatively, in the realm of text-to-image models, a similar concept is emerging – text-driven image editing. For instance, Prompt-to-Prompt image editing [**66**] is a technique in which a user generates an image based on a prompt; then, they can tweak the prompt to create a new image, making only minimal changes to match the updated prompt. An initial prompt of "a cake with decorations" might be refined to "a cake with jelly bean decorations" and the initial image would be updated accordingly. This approach holds two key advantages. Firstly, it deepens the user's understanding of how the artifact develops, enabling a clearer grasp of its progression. Secondly, it empowers users to fine-tune and repair existing elements [**4**]. For my studies on the Steering Interface Layers, these benefits had downstream effects for people's self-efficacy, learning, engagement and agency in the process, and ownership over the final artifact. In summary, by building upon what's already there, editing and steering are similar in that they enhance a user's engagement through progressively updating and clarifying their intentions and instructions.

5.4.2. Generalizing and Extending Expression Interface Layers

5.4.2.1. Using AI collaborator to Formulate Concepts and Articulate Them with Machine Features. The motivation behind Expression Interface Layers was to make up for the deficiency in a context-aware AI agent not having constructs that directly match a designer's concepts for the situations and experiences they wanted to facilitate. In framing the problem as helping designers express their high-level concepts with the AI agent's available constructs, I discovered that humans have cognitive deficiencies in their process of expressing ideas to machines. The purpose of the tools in the Expression Interface Layer is to address these cognitive challenges that humans face when they must translate their overarching human ideas into lower-level machine representations.

Recent developments in Large Language Models (LLMs) have ushered in an era where AI collaborators can provide suggestions in tasks related to creative design ideation and turning natural language ideas into computer code. The capability of an LLM to work with available programming APIs is outstanding. By training on the documentation of public programming APIs, LLMs are capable of using programming APIs to fulfill user's natural-language instructions. The capability of an LLM to do conceptual design work is also considerable since they have learned a range of common sense and domain-specific knowledge for a variety of tasks. Such developments have impacted the specific areas of designing and programming context-aware experiences, which is a domain this dissertation specifically covers. At the time of writing, the chat-based LLMs like ChatGPT have been trained on internet data containing information about Yelp's Public API for place categories and thus understands the semantics of those categories and how they are used. ChatGPT is also capable of reasoning about knowledge about human activities, and the types of places that support such activities.

While the tools implemented in this dissertation augmented a creator's manual process of fleshing out subconcepts and linking to relevant features, the future area of authoring context-aware experiences, and Expression Interface Layers more generally, will be impacted by these LLM advances. As future tools are developed for expressing concepts to context-aware agents, it should not be surprising to see tools incorporating AI collaborators who are capable of producing good first drafts of situational requirements, as captured through logical expressions. For example, an AI collaborator like ChatGPT could be instructed to (1) brainstorm types of places to do an activity and (2) identify the Yelp categories that can detect these types of places. At the time of writing, I tested this for the situation "places for enjoying a relaxing stroll"; I found that ChatGPT produced several concepts for types of places (City Park, Botanic Garden, Beach Boardwalk, Historic District, Zen Garden, Urban Waterfront, Art and Sculpture Garden) and linked these types of places to the closest Yelp categories when available (parks, gardens, beaches, landmarks, art galleries).

Nonetheless, the structure of the problem that the Expression Interface Layer solves remains the same. Whether a person has access to more capable AI collaborators like LLMs to aid them, the bridging problem is what the designer must still face when expressing a human's overarching concept in terms of lower-level machine constructs. Whichever specific techniques will be used, they will need to be wary of challenges such as underscoping of concepts, underscoping of machine features, and addressing inaccuracies when machine constructs operate differently than intended.

The techniques I implemented for the Expression Interface Layer were simple and straightforward but were nonetheless effective for overcoming the bridging challenges as compared to the version without any cognitive bridging tools. The introduction of AI capabilities like LLMs can aid in the development of more sophisticated techniques to further tackle these same bridging challenges. For example, since LLMs generations to the question can be used to brainstorm many types of subconcepts or examples, this type of computational support can overcome narrowly defined expressions by helping designers recognize more concepts they might have not thought of, and other relevant context features.

5.4.2.2. Addressing Bridging Problems in Other Domains. The Expression Interface Layer tackled a general problem, where a person is representing their overarching idea in terms of a rich constellation of subconcepts and fine-grained attributes that can be detected by an AI system. Although our work focused on this problem in the domain of context-aware computing, we anticipate that Interface Layers for representing conceptually rich, overarching ideas will be important for other domains as well.

One adjacent domain is in human-centered AI model development, where researchers have also been interested in supporting designers and domain experts in working at the conceptual-level when defining AI models. For example, the Model Sketching technical framework [**92**] refocuses practitioner attention on composing high-level, human-understandable concepts that the model is expected to reason over (e.g., profanity, racism, or sarcasm in a content moderation task) using zero-shot concept instantiation. This framework for concept-based AI model development is promising because it prompts high-level modes of thought when modeling decisions. However, model designers also mentioned needing more assistance when brainstorming subconcepts for an overarching idea (e.g., "racism" is one way a social media post could be hateful, but what other ways?). You might notice that this is an example of how designers can struggle to recall or discover other human-understandable concepts to describe to AI systems. It has a striking similarity to one of the bridging problems our Expression Interface Layer addressed.

The main implication of my work is that designers can face specific cognitive challenges when expressing a conceptually-rich idea; and cognitive bridging tools targeted towards these challenges can enhance their expression process. Therefore, I'd recommend understanding the exact bridging problems that designers can face when doing concept-based AI model development. What can get in the way of brainstorming ideas? Are there challenges in developing models of subconcepts that precisely operate as the designer intends? After understanding the bridging problems, we can consider what techniques might address these challenges. For example, if challenges arise in cognitive fixation, designers can adopt principles from design-ideation and analogical design when designing a new Expression Interface Layer—similar to how we were inspired when developing the reflect and expand prompts.

Future development of the Expression Interface Layers for the domain of modeling high-level concepts can incorporate other techniques, such as LLMs, when overcoming cognitive challenges such as brainstorming. Since chat-based LLMs have demonstrated exceptional abilities to generate creative ideas, dialogues with them could also help in this conceptual mapping and expression process. Developing visual interfaces to capture concepts, and map out the conceptual space, will likely be an important complement to this iterative, construction process.

5.4.3. Generalizing and Extending Execution Interface Layers

The Execution Interface Layer work studies a class of intelligent agents that can fulfill users' instructions and focuses on improving the likelihood that these agents can meet a user's expectations when executing tasks. These potential use cases for intelligent agents span across a wide variety of domains and tasks. Software or user interface agents have been one use-case long-envisioned for intelligent agents [**123, 130**] with recent technical innovations bringing them much closer to a reality [**94, 1**]. These intelligent software agents can fulfill user requests by automating digital actions on various software tools on mobile and desktop. Example tasks include "order a latte" via the Starbucks app [**94**] to "find me a list of homes suitable for a family of 4. Budget is $600k" via the Redfin real estate site [**1**]. Generally, these tasks can span simple uses like executing a repeatable set of actions in a user interface, to more complex tasks like information gathering. Several intelligent agents have been developed for this use case. Researchers in HCI have developed programming-by-demonstration systems that empower users to establish automation for digital tasks using actions on UI elements in smartphones and computers [**94**]. Similar to this, companies like Adept AI [**1**] are engaged in training foundational AI models with the ability to interpret text instructions and translate them into digital actions spanning a diverse array of computer tools.

Similar to the challenges driving the development of our Execution Interface Layer, errors and breakdowns due to unfamiliar scenarios during execution prompt the creation of interactive tools to identify and manage them. For instance, an intelligent agent tasked with automating smartphone activities might encounter unfamiliar user interfaces during execution, necessitating appropriate handling strategies. When such unfamiliar situations can be caught, an effective approach is to defer to the human requester or operator during execution for further clarification so this case and similar future cases can be handled [**94**].

With more sophisticated, intelligent agents capable of executing tasks involving multiple digital actions, the significance of addressing discrepancies in user expectations grows. Consider the request of "order my breakfast groceries from the Amazon app. My list consists of blueberries, oatmeal, milk". Even if an agent does not encounter technical issues using the UI elements to purchase the required items, users can still be disappointed if the agent purchases blueberries that are much too expensive relative to the person's internal expectations, which can occur due to seasonal price fluctuations. Such breakdowns in implicit user expectations are impossible for intelligent agents to catch if they don't know about these expectations. As we expand our view of execution issues, from technical errors to user expectation breakdowns, additional user interfaces, such as those advanced by the Execution Interface Layer, will be increasingly important so humans can recognize and state their implicit expectations so intelligent agents can respect them.

While working on designing the Execution Interface Layer, I discovered that simulating various execution scenarios was valuable in identifying issues with implicit expectations before the actual execution phase. This approach is akin to other efforts to use large amounts of past and simulated data to discover issues and enhance the robustness of AI execution in open-world scenarios [**68**]. Taking the example of intelligent agents ordering grocery items, conducting simulations throughout different times of the year could unveil seasonal price fluctuations. This information could then assist in making requests more attuned to a user's price sensitivities.

Finding good strategies for adjusting to constraints, and communicating them to intelligent agents is an important area, which my designs for the Execution Interface Layer could shed light on. The approach to reformulating hard-to-meet requests included scaffolds that helped users identify the larger goal behind their request so they could find alternative formulations. Therefore, if blueberries are often out of stock in an online marketplace, it's important to recognize that fruit on top of oatmeal was the core intention behind the user's request so that appropriate substitutes can be found. Moreover, this approach to encoding values over different execution scenarios could allow users to more effectively meet their expectations so intelligent agents could decide to make adjustments more autonomously. It's possible that an Execution Interface Layer for these intelligent agents could provide a means for users to communicate their values over different scenarios (e.g., when different options are available, with variations in their price and freshness), so that when users request "purchase the best fruit to accompany my breakfast this week" decision-theoretic mechanisms could use their encoding of preferences to choose the best one on behalf of the user. Although these usage scenarios are hypothetical, they illustrate the different ways that an Execution Interface Layer could elicit implicit ideas for a user's goals to drive adjustments so intelligent agents can better satisfy the user's full set of goals and expectations.

5.5. Promoting the Values of Human Engagement

As AI gets better, a subset of developers will be eager to create AI-powered applications that can automatically do tasks, such as content-creation, that requires very little user involvement. However, the negative effects of this eagerness for scaling AI capabilities is already being felt by artists, where image generators trained as text/image pairs have caused harms, including economic loss due to these AIs being viewed as replacements for human artists work [**84**]. This automation viewpoint places a sole priority on speed of content creation and scale at which a task that required craft can be completed. But the automation viewpoint lacks an appreciation that activities like making art or composing music are uniquely human endeavors that involve human sensibilities and engagement. For example, creating music that sounds more sad than happy, for instance, doesn't necessarily enable that creator to reflect on their own experience of sadness and to convey the tenors and textures of their personal understanding of that emotion. These activities of self-expression are important because creators derive personal satisfaction when they can craft pieces uniquely associated with their experience of creation, and that capture their personal perspective and emotions [**10**].

My work with Interface Layers emphasized the importance of reconfiguring the interface for communicating with AI to respect the human's role and perspective when creating artifacts and experiences enabled by AI. A Human-AI Interface Layer supports an engaged process of reflecting upon and discovering ways to convey nuanced ideas in artifacts through the use of AI capabilities. I designed them with an awareness that self-expression is characterized by an emergent making process, requiring human engagement from start to end. Through my studies of Steering Interface Layers, for example, I found that such a process not only helps to better achieve creative goals but also supports greater feelings of ownership and self-efficacy during creation. I believe Human-AI Interface Layers will be a useful approach to promote the values of crafting for AI-assisted creation artifacts and experiences, especially as AI's default interfaces may not be intentionally designed to promote human engagement in these activities.

Human engagement in the creation process is not only good for achieving artifact quality that reflects a creator's self-expression. For novices, in particular, retaining an engaged role in the process is important for developing confidence and mastery. Yet, as AI becomes increasingly capable of performing tasks thought to be uniquely in the realm of human intelligence, novices will have the option to request assistance from AI. This will make it increasingly tempting to ask AI to do the entire task on their behalf—which would be detrimental to their own development and mastery. This tension is a great challenge that educators will have to navigate to support novice skill development, given the eagerness to incorporate AI and technological assistance in human endeavors such as crafting activities. In this direction, recent research has used workshops with craft educators to understand their perceptions of the opportunities and concerns of using text-to-image generative AI. They found opportunities and challenges in using AI in targeted stages of the design process (ideation, externalization, design constraints, embodied making, and assessment) [**128**]. Instead of taking an all-or-nothing perspective on the use of AI, learners and educators can decompose any human activity into its constituent subprocesses, and thoughtfully reflect on the positive benefits and negative consequences that AI assistance can have for learning. Along this line of thinking, approaches like Human-AI Interface Layers can configure the usage of AI so it effectively assists in some parts of the process, while learners can focus their attention on another subskill they are doing targeted practice on.

CHAPTER 6

Conclusion

This thesis proposes a crucial layer in an AI system's application stack called *the Human-AI Interface Layer*. This layer serves as a mediator enabling creators to effectively communicate their ideas and intentions while crafting artifacts and facilitating experiences with an AI capability. Through exposing intuitive and functional constructs, Human-AI Interface Layers enable modifications to the operations of pre-existing AI capabilities. This support leads to a more straightforward expression of ideas. Human-AI Interface Layers enable an effective process so creators can form, evaluate, and clarify the ideas they desire to convey to AI.

This thesis contributes three kinds of Human-AI Interface Layers that address corresponding challenges creators can face when using conventional AI interfaces to form and articulate their ideas and intentions to AI systems.

- I contributed a *Steering Interface Layer* that partitioned and constrained an existing generative AI model's outputs. This allowed creators to exercise fine-grained control while progressively guiding the co-creation of a generated artifact toward desired directions.

- I proposed an *Expression Interface Layer* that supports an effective process for encoding a high-level, overarching idea in terms of an AI system's low-level constructs. It provided a visual workspace and cognitive bridging tool that helped a designer flesh out their high-level concepts and forage for relevant and precisely-matching constructs, leading to comprehensive and accurate communication of their intentions.
- I advanced an *Execution Interface Layer* that helps humans consider realistic constraints when automated systems execute, so they can recognize and state their implicit expectations so AI systems can adapt to respect them.

The fundamental idea of this thesis is to improve a user's ability to communicate their ideas despite AI's inherently limited interfaces for doing so. Human-AI Interface Layers expose new user interface constructs with accompanying computational system components so people can communicate personal and meaningful ideas to AI systems that would have otherwise been impossible to do with an unmodified AI capability. Human-AI Interface Layers facilitate the incremental forming and refining of a user's intent, avoiding the failure of expecting a user's comprehensive communication upfront. By promoting incremental processes with pauses for evaluating, adjusting, and reflecting on one's content, a dialogue can be established to enable humans to refine different pieces of their ideas and their strategies for communicating them to the AI system. We believe that Human-AI Interface Layers will be a powerful approach to empowering humans to imbue their AI systems with an awareness and sensitivity to their personal and subjective ideas and desires—ranging from personal emotions, social situations, and social norms—leading to a future where AI systems can foster the human goals, expectations, and values core to engaging in personal pursuits and human endeavors.

Bibliography

[1] Adept ai - company home page. https://web.archive.org/web/20230807023233/https://www.adept.ai/, 2023. [Online; accessed 12-August-2023].

[2] Abowd, G. D., and Mynatt, E. D. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI) 7*, 1 (2000), 29–58.

[3] Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., and Frank, C. Musiclm: Generating music from text, 2023.

[4] Agrawala, M. Unpredictable black boxes are terrible interfaces, Mar 2023.

[5] Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T. Power to the people: The role of humans in interactive machine learning. *Ai Magazine 35*, 4 (2014), 105–120.

[6] Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., and Horvitz, E. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2019), CHI '19, ACM, pp. 3:1–3:13.

[7] Andersen, K., and Knees, P. The dial: Exploring computational strangeness. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2016), CHI EA '16, Association for Computing Machinery, p. 1352–1358.

[8] Avrahami, D., Fussell, S. R., and Hudson, S. E. Im waiting: Timing and responsiveness in semi-synchronous communication. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work* (New York, NY, USA, 2008), CSCW '08, Association for Computing Machinery, p. 285–294.

[9] Ball, L. J., and Christensen, B. T. Advancing an understanding of design cognition and design metacognition: Progress and prospects. *Design Studies 65* (2019), 35–59.

[10] Bardzell, S., Rosner, D. K., and Bardzell, J. Crafting quality in design: Integrity, creativity, and public sensibility. In *Proceedings of the Designing Interactive Systems Conference* (New York, NY, USA, 2012), DIS '12, Association for Computing Machinery, p. 11–20.

[11] Bayer, J. B., Ellison, N. B., Schoenebeck, S. Y., and Falk, E. B. Sharing the small moments: ephemeral social interaction on snapchat. *Information, Communication & Society 19*, 7 (2016), 956–977.

[12] Bellotti, V., and Edwards, K. Intelligibility and accountability: human considerations in context-aware systems. *Human–Computer Interaction 16*, 2-4 (2001), 193–212.

[13] Benjamini, Y., and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological) 57*, 1 (1995), 289–300.

[14] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. Dbpedia-a crystallization point for the web of data. *Journal of web semantics 7*, 3 (2009), 154–165.

[15] Boothby, E. J., Smith, L. K., Clark, M. S., and Bargh, J. A. Psychological distance moderates the amplification of shared experience. *Personality and Social Psychology Bulletin 42*, 10 (2016), 1431–1444.

[16] Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning* (2012).

[17] Bowsher-Murray, C., Jones, C. R., and von dem Hagen, E. Beyond simultaneity: Temporal interdependence of behavior is key to affiliative effects of interpersonal synchrony in children. *Journal of Experimental Child Psychology 232* (2023), 105669.

[18] Brade, S., Wang, B., Sousa, M., Oore, S., and Grossman, T. Promptify: Text-to-image generation through interactive prompt exploration with large language models. *arXiv preprint arXiv:2304.09337* (2023).

[19] Braun, V., and Clarke, V. Using thematic analysis in psychology. *Qualitative Research in Psychology 3*, 2 (2006), 77–101.

[20] Brooks, M., Amershi, S., Lee, B., Drucker, S. M., Kapoor, A., and Simard, P. Featureinsight: Visual support for error-driven feature ideation in text classification. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2015), pp. 105–112.

[21] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[22] Burke, M., Kraut, R., and Marlow, C. Social capital on facebook: Differentiating uses and users. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2011), ACM, pp. 571–580.

[23] Cheng, J., and Bernstein, M. Catalyst: triggering collective action with thresholds. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing* (2014), pp. 1211–1221.

[24] Chung, J. J. Y., and Adar, E. Promptpaint: Steering text-to-image generation through paint medium-like interactions. *arXiv preprint arXiv:2308.05184* (2023).

[25] Clark, E., Ross, A. S., Tan, C., Ji, Y., and Smith, N. A. Creative writing with a machine in the loop: Case studies on slogans and stories. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces* (2018), ACM, pp. 329–340.

[26] Compton, K., and Mateas, M. Casual creators. In *Proceedings of the Sixth International Conference on Computational Creativity (ICCC 2015)* (Park City, Utah, June–July 2015), H. Toivonen, S. Colton, M. Cook, and D. Ventura, Eds., Brigham Young University, pp. 228–235.

[27] Cong, N., Cheng, K., Zhang, H., and Louie, R. Collective narrative: Scaffolding community storytelling through context-awareness. In *Companion Publication of the 2021 Conference on Computer Supported Cooperative Work and Social Computing* (New York, NY, USA, 2021), CSCW '21, Association for Computing Machinery, p. 40–43.

[28] Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. Simple and controllable music generation, 2023.

[29] Crilly, N., and Cardoso, C. Where next for research on fixation, inspiration and creativity in design? *Design Studies 50* (2017), 1–38.

[30] Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: A simple approach to controlled text generation, 2020.

[31] Davis, N., Hsiao, C.-P., Yashraj Singh, K., Li, L., and Magerko, B. Empirically studying participatory sense-making in abstract drawing with a co-creative cognitive agent. In *Proceedings of the 21st International Conference on Intelligent User Interfaces* (New York, NY, USA, 2016), IUI '16, ACM, pp. 196–207.

[32] Dearman, D., Sohn, T., and Truong, K. N. Opportunities exist: Continuous discovery of places to perform activities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2011), CHI '11, Association for Computing Machinery, p. 2429–2438.

[33] Dearman, D., and Truong, K. N. Identifying the activities supported by locations with community-authored content. In *Proceedings of the 12th ACM international conference on Ubiquitous computing* (2010), pp. 23–32.

[34] Depping, A. E., and Mandryk, R. L. Cooperation and interdependence: How multiplayer games increase social closeness. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (2017), ACM, pp. 449–461.

[35] Dey, A. K., Abowd, G. D., and Salber, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction 16*, 2-4 (2001), 97–166.

[36] Dey, A. K., Hamid, R., Beckmann, C., Li, I., and Hsu, D. a cappella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2004), ACM, pp. 33–40.

[37] Dey, A. K., and Newberger, A. Support for context-aware intelligibility and control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2009), CHI '09, Association for Computing Machinery, p. 859–868.

[38] Dey, A. K., Sohn, T., Streng, S., and Kodama, J. icap: Interactive prototyping of context-aware applications. In *International Conference on Pervasive Computing* (2006), Springer, pp. 254–271.

[39] Dinculescu, M., Engel, J., and Roberts, A. Midime: Personalizing a musicvae model with user data. In *Workshop on Machine Learning for Creativity and Design, NeurIPS* (2019).

[40] Dinculescu, M., and Huang, C.-Z. A. Coucou: An expanded interface for interactive composition with coconet, through flexible inpainting, 2019.

[41] Donahue, C., Simon, I., and Dieleman, S. Piano genie. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (New York, NY, USA, 2019), IUI '19, ACM, pp. 160–164.

[42] Dourish, P. Seeking a foundation for context-aware computing. *Human–Computer Interaction 16*, 2-4 (2001), 229–241.

[43] Dourish, P. Re-space-ing place: " place" and" space" ten years on. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (2006), pp. 299–308.

[44] D'Zmura, M. Behind the scenes: popular times and live busyness information. `https://blog.google/products/maps/maps101-popular-times-and-live-busyness-information/`, 2020.

[45] Eck, D., and Schmidhuber, J. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing* (2002).

[46] Fan, J. E., Dinculescu, M., and Ha, D. collabdraw: An environment for collaborative sketching with an artificial agent. In *Proceedings of the 2019 on Creativity and Cognition* (2019), ACM, pp. 556–561.

[47] Farbood, M. M., Pasztor, E., and Jennings, K. Hyperscore: a graphical sketchpad for novice composers. *IEEE Computer Graphics and Applications 24*, 1 (2004), 50–54.

[48] Fiebrink, R. A. Real-time human interaction with supervised learning algorithms for music composition and performance. *PhD dissertation, Princeton University* (2011).

[49] Fogarty, J., Hudson, S. E., Atkeson, C. G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. C., and Yang, J. Predicting human interruptibility with sensors. *ACM Trans. Comput.-Hum. Interact. 12*, 1 (mar 2005), 119–146.

[50] Fraser, N. Ten things we've learned from blockly. In *Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE* (2015), IEEE, pp. 49–50.

[51] Fukayama, S., Yoshii, K., and Goto, M. Chord-sequence-factory: A chord arrangement system modifying factorized chord sequence probabilities. *International Society for Music Information Retrieval* (2013).

[52] Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. The vocabulary problem in human-system communication. *Communications of the ACM 30*, 11 (1987), 964–971.

[53] Gero, K. I., and Chilton, L. B. Metaphoria: An algorithmic companion for metaphor creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), ACM, p. 296.

[54] Gillick, J., Roberts, A., Engel, J., Eck, D., and Bamman, D. Learning to groove with inverse sequence transformations. *arXiv preprint arXiv:1905.06118* (2019).

[55] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT press, 2016.

[56] Granger, J., Aviles, M., Kirby, J., Griffin, A., Yoon, J., Lara-Garduno, R., and Hammond, T. Lumanote: A real-time interactive music composition assistant. In *Intelligent User Interfaces Workshops* (2018).

[57] Grote, F., Andersen, K., and Knees, P. Collaborating with intelligent machines: Interfaces for creative sound. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2015), CHI EA '15, Association for Computing Machinery, p. 2345–2348.

[58] Guzdial, M., Liao, N., Chen, J., Chen, S.-Y., Shah, S., Shah, V., Reno, J., Smith, G., and Riedl, M. O. Friend, collaborator, student, manager: How design of an ai-driven game level editor affects creators. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2019), CHI '19, Association for Computing Machinery.

[59] Hadjeres, G., Pachet, F., and Nielsen, F. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning* (2017), pp. 1362–1371.

[60] Haresamudram, H., Beedu, A., Agrawal, V., Grady, P. L., Essa, I., Hoffman, J., and Plötz, T. Masked reconstruction based self-supervision for

human activity recognition. In *Proceedings of the 2020 International Symposium on Wearable Computers* (2020), pp. 45–49.

[61] Harrison, S., and Dourish, P. Re-place-ing space: the roles of place and space in collaborative systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work* (1996), pp. 67–76.

[62] Hart, S. G., and Staveland, L. E. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in Psychology*, vol. 52. Elsevier, 1988, pp. 139–183.

[63] Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. Enabling factorized piano music modeling and generation with the maestro dataset. In *International Conference on Learning Representations* (2019).

[64] Hayes-Roth, B., and Hayes-Roth, F. A cognitive model of planning. *Cognitive science 3*, 4 (1979), 275–310.

[65] Herlihy, M. P., and Wing, J. M. Specifying graceful degradation. *IEEE Transactions on Parallel and Distributed Systems 2*, 1 (1991), 93–104.

[66] Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-Or, D. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626* (2022).

[67] Hirskyj-Douglas, I., Kantosalo, A., Monroy-Hernández, A., Zimmermann, J., Nebeling, M., and Gonzalez-Franco, M. Social ar: Reimagining and interrogating the role of augmented reality in face to face social interactions. In *Conference Companion Publication of the 2020 on Computer Supported Cooperative Work and Social Computing* (2020), pp. 457–465.

[68] Horvitz, E. Ai, people, and the open world.

[69] Horvitz, E. Reasoning under varying and uncertain resource constraints. In *AAAI* (1988), vol. 88, pp. 111–116.

[70] Horvitz, E., Koch, P., and Apacible, J. Busybody: creating and fielding personalized models of the cost of interruption. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (2004), pp. 507–510.

[71] Horvitz, E. J. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence* (Arlington, Virginia, United States, 1987), UAI'87, AUAI Press, pp. 429–447.

[72] Horvitz, E. J., Cooper, G. F., and Heckerman, D. E. Reflection and action under scarce resources: Theoretical principles and empirical study. In *IJCAI* (1989), pp. 1121–1127.

[73] Hu, C. H. Self-disclosure for early relationship development through situated prompts in opportunistic collective experiences. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts* (2022), pp. 1–5.

[74] Huang, C.-Z. A., Cooijmnas, T., Roberts, A., Courville, A., and Eck, D. Counterpoint by convolution. *International Society for Music Information Retrieval.* (2017).

[75] Huang, C.-Z. A., Duvenaud, D., and Gajos, K. Z. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces* (2016), ACM, pp. 241–250.

[76] Huang, C.-Z. A., Hawthorne, C., Roberts, A., Dinculescu, M., Wexler, J., Hong, L., and Howcroft, J. The bach doodle: Approachable music composition with machine learning at scale. *International Society for Music Information Retrieval.* (2019).

[77] Huang, C.-Z. A., Koops, H. V., Newton-Rex, E., Dinculescu, M., and Cai, C. J. Ai song contest: Human-ai co-creation in songwriting, 2020.

[78] Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer. In *International Conference on Learning Representations* (2019).

[79] Huang, Q., Park, D. S., Wang, T., Denk, T. I., Ly, A., Chen, N., Zhang, Z., Zhang, Z., Yu, J., Frank, C., Engel, J., Le, Q. V., Chan, W., Chen, Z., and Han, W. Noise2music: Text-conditioned music generation with diffusion models, 2023.

[80] Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J., and Yang, J. Predicting human interruptibility with

sensors: a wizard of oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2003), ACM, pp. 257–264.

[81] Inc., Y. Yelp Open Dataset. `https://www.yelp.com/dataset`, 2020.

[82] Inc., Y. Yelp fusion api. `https://www.yelp.com/developers/documentation/v3`, 2021.

[83] Jacob, M., and Magerko, B. Interaction-based authoring for scalable co-creative agents. In *Proceedings of the Sixth International Conference on Computational Creativity (ICCC 2015)* (Park City, Utah, June–July 2015), H. Toivonen, S. Colton, M. Cook, and D. Ventura, Eds., Brigham Young University, pp. 236–243.

[84] Jiang, H. H., Brown, L., Cheng, J., Khan, M., Gupta, A., Workman, D., Hanna, A., Flowers, J., and Gebru, T. Ai art and its impact on artists. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society* (2023), pp. 363–374.

[85] Johnson, D. W. Social interdependence: interrelationships among theory, research, and practice. *American psychologist 58*, 11 (2003), 934.

[86] Jonker, T. R., Desai, R., Carlberg, K., Hillis, J., Keller, S., and Benko, H. The role of ai in mixed and augmented reality interactions. In *CHI2020 ai4hci Workshop Proceedings. ACM* (2020).

[87] Karimi, P., Maher, M. L., Davis, N., and Grace, K. Deep learning in a computational model for conceptual shifts in a co-creative design system. *arXiv preprint arXiv:1906.10188* (2019).

[88] Kaur, H., Williams, A. C., McDuff, D., Czerwinski, M., Teevan, J., and Iqbal, S. T. Optimizing for happiness and productivity: Modeling opportune moments for transitions and breaks at work. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020), pp. 1–15.

[89] Kim, Y., Gergle, D., and Zhang, H. Hit-or-wait: Coordinating opportunistic low-effort contributions to achieve global outcomes in on-the-go crowdsourcing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018), ACM, p. 96.

[90] Kim, Y., Harburg, E., Azria, S., Shaw, A., Gerber, E., Gergle, D., and Zhang, H. Studying the effects of task notification policies on participation and outcomes in on-the-go crowdsourcing. *AAAI HCOMP* (2016).

[91] Koch, J., Lucero, A., Hegemann, L., and Oulasvirta, A. May ai? design ideation with cooperative contextual bandits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2019), CHI '19, Association for Computing Machinery.

[92] Lam, M. S., Ma, Z., Li, A., Freitas, I., Wang, D., Landay, J. A., and Bernstein, M. S. Model sketching: Centering concepts in early-stage machine learning model design. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2023), CHI '23, Association for Computing Machinery.

[93] Laput, G., and Harrison, C. Sensing fine-grained hand activity with smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), pp. 1–13.

[94] Li, T. J.-J., Azaria, A., and Myers, B. A. Sugilite: Creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, Association for Computing Machinery, p. 6038–6049.

[95] Li, X. L., and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[96] Liang, F. Bachbot: Automatic composition in the style of bach chorales. *Masters thesis, University of Cambridge* (2016).

[97] Ling, C., Blackburn, J., De Cristofaro, E., and Stringhini, G. Slapping cats, bopping heads, and oreo shakes: Understanding indicators of virality in tiktok short videos. In *14th ACM Web Science Conference 2022* (New York, NY, USA, 2022), WebSci '22, Association for Computing Machinery, p. 164–173.

[98] Linsey, J. S., Markman, A. B., and Wood, K. L. Design by analogy: A study of the wordtree method for problem re-representation. *Journal of Mechanical Design 134*, 4 (04 2012). 041009.

[99] Liu, V., Qiao, H., and Chilton, L. Opal: Multimodal image generation for news illustration. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2022), UIST '22, Association for Computing Machinery.

[100] LLC, G. Google Awareness API. `https://developers.google.com/awareness/`, 2021.

[101] Louie, R., Coenen, A., Huang, C. Z., Terry, M., and Cai, C. J. Novice-ai music co-creation via ai-steering tools for deep generative models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2020), CHI '20, Association for Computing Machinery, p. 1–13.

[102] Louie, R., Engel, J., and Huang, A. Expressive communication: A common framework for evaluating developments in generative models and steering interfaces, 2021.

[103] Louie, R., Garg, K., Werner, J., Sun, A., Gergle, D., and Zhang, H. Opportunistic collective experiences: Identifying shared situations and structuring shared activities at distance. *Proceedings of the ACM on Human-Computer Interaction 4*, CSCW3 (2021), 1–32.

[104] Louie, R., Gergle, D., and Zhang, H. Affinder: Expressing concepts of situations that afford activities using context-detectors. In *CHI Conference on Human Factors in Computing Systems* (2022), pp. 1–18.

[105] Mahaney, S. R., and Schneider, F. B. Inexact agreement: Accuracy, precision, and graceful degradation. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing* (1985), pp. 237–249.

[106] Mayer, R. C., Davis, J. H., and Schoorman, F. D. An integrative model of organizational trust. *Academy of Management Review 20*, 3 (1995), 709–734.

[107] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[108] Morshed, M. B., Saha, K., Li, R., D'Mello, S. K., De Choudhury, M., Abowd, G. D., and Plötz, T. Prediction of mood instability with passive sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 3*, 3 (2019), 1–21.

[109] Oh, C., Song, J., Choi, J., Kim, S., Lee, S., and Suh, B. I lead, you help but only with enough details: Understanding user experience of co-creation with artificial intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), CHI '18, ACM, pp. 649:1–649:13.

[110] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton,

J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* (2022), S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., pp. 27730–27744.

[111] Payne, C. MuseNet, 2019.

[112] Rachuri, K. K., Hossmann, T., Mascolo, C., and Holden, S. Beyond location check-ins: Exploring physical and soft sensing to augment social check-in apps. In *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)* (2015), IEEE, pp. 123–130.

[113] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

[114] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation, 2021.

[115] Ramos, J. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First instructional Conference on machine learning* (2003), vol. 242, pp. 133–142.

[116] Rivero-Rodriguez, A., Pileggi, P., and Nykänen, O. A. Mobile context-aware systems: Technologies, resources and applications. *International Journal of Interactive Mobile Technologies (iJIM) 10*, 2 (Apr. 2016), pp. 25–32.

[117] Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. A hierarchical latent vector model for learning long-term structure in music. In *International Conference on Machine Learning (ICML)* (2018).

[118] Roberts, A., Hawthorne, C., and Simon, I. Magenta.js: A javascript api for augmenting creativity with deep learning. In *Joint Workshop on Machine Learning for Music (ICML)* (2018).

[119] RunwayML. Runwayml's ai magic tool infinite image.

[120] Sadilek, A., Krumm, J., and Horvitz, E. Crowdphysics: Planned and opportunistic crowdsourcing for physical tasks. In *Proceedings of the International AAAI Conference on Web and Social Media* (2013), vol. 7, pp. 536–545.

[121] Salber, D., Dey, A. K., and Abowd, G. D. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1999), ACM, pp. 434–441.

[122] Schwarzer, R., and Jerusalem, M. Generalized self-efficacy scale. *Measures in Health Psychology: A User's Portfolio. Causal and Control Beliefs 1*, 1 (1995), 35–37.

[123] Shneiderman, B., and Maes, P. Direct manipulation vs. interface agents. *Interactions 4*, 6 (nov 1997), 42–61.

[124] Siangliulue, P., Arnold, K. C., Gajos, K. Z., and Dow, S. P. Toward collaborative ideation at scale: Leveraging ideas from others to generate more creative and diverse ideas. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (New York, NY, USA, 2015), CSCW '15, Association for Computing Machinery, p. 937–945.

[125] Siangliulue, P., Chan, J., Dow, S. P., and Gajos, K. Z. Ideahound: Improving large-scale collaborative ideation with crowd-powered real-time semantic modeling. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (New York, NY, USA, 2016), UIST '16, Association for Computing Machinery, p. 609–624.

[126] Simon, I., Morris, D., and Basu, S. Mysong: Automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, Association for Computing Machinery, pp. 725–734.

[127] Ur, B., McManus, E., Pak Yong Ho, M., and Littman, M. L. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), ACM, pp. 803–812.

[128] Vartiainen, H., and Tedre, M. Using artificial intelligence in craft education: crafting with text-to-image generative models. *Digital Creativity 34*, 1 (2023), 1–21.

[129] Wang, J., Wang, Y., Weng, N., Chai, T., Li, A., Zhang, F., and Yu, S. Will you ever become popular? learning to predict virality of dance clips. *ACM Trans. Multimedia Comput. Commun. Appl. 18*, 2 (feb 2022).

[130] Wang, X. F., Zhang, S. C., and Kiliccote, P. K. K. H. Anytime algorithm for agent-mediated merchant information gathering. In *Proceedings of the Fourth International Conference on Autonomous Agents* (New York, NY, USA, 2000), AGENTS '00, Association for Computing Machinery, p. 333–340.

[131] Werner, J., and Sun, A. Cerebro: A platform for opportunistic collective experiences. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), CHI EA '18, Association for Computing Machinery, p. 1–6.

[132] Wikipedia contributors. Dixit (card game) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Dixit_(card_game)`, 2019. [Online; accessed 19-September-2019].

[133] Xu, X. T., Xiong, R., Wang, B., Min, D., and Dow, S. P. Ideaterelate: An examples gallery that helps creators explore ideas in relation to their own. *Proc. ACM Hum.-Comput. Interact. 5*, CSCW2 (oct 2021).

[134] Yang, D., Zhang, D., and Qu, B. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST) 7*, 3 (2016), 1–23.

[135] Yang, Q., Cranshaw, J., Amershi, S., Iqbal, S. T., and Teevan, J. Sketching nlp: A case study of exploring the right things to design with language intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2019), CHI '19, Association for Computing Machinery, p. 1–12.

[136] Young, H., Dumoulin, V., Castro, P. S., Engel, J., and Huang, C.-Z. A. Compositional steering of music transformers.

[137] Zhang, H., Law, E., Miller, R., Gajos, K., Parkes, D., and Horvitz, E. Human computation tasks with global constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), pp. 217–226.

[138] Zhou, P., Zheng, Y., Li, Z., Li, M., and Shen, G. Iodetector: A generic service for indoor outdoor detection. In *Proceedings of the 10th acm conference on embedded network sensor systems* (2012), pp. 113–126.

[139] Zilberstein, S. Using anytime algorithms in intelligent systems. *AI magazine 17*, 3 (1996), 73–73.