석 사 학 위 논 문
Master's Thesis

초보 디자이너 지원을 위한 다중 모달리티 간의 연결

Mapping Multiple Modalities to Empower Novice Designers

2022

김 태 수 (金 太 洙 Kim, Tae Soo)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

초보 디자이너 지원을 위한 다중 모달리티 간의 연결

2022

김 태 수

한 국 과 학 기 술 원

전산학부

초보 디자이너 지원을 위한 다중 모달리티 간의 연결

김 태 수

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2021년 12월 13일

심사위원장 김 주 호 (인)

심 사 위 원 오 혜 연 (인)

심 사 위 원 이 탁 연 (인)

Mapping Multiple Modalities to Empower Novice Designers

Tae Soo Kim

Advisor: Juho Kim

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Daejeon, Korea
December 13, 2021

Approved by

Juho Kim
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MCS 김태수. 초보 디자이너 지원을 위한 다중 모달리티 간의 연결. 전산학부 . 2022년. 64+v 쪽. 지도교수: 김주호. (영문 논문)
Tae Soo Kim. Mapping Multiple Modalities to Empower Novice Designers. School of Computing . 2022. 64+v pages. Advisor: Juho Kim. (Text in English)

초 록

소프트웨어 애플리케이션은 초보자의 설계 능력을 확장시킬 수 있는 잠재력을 가진다. 그러나 소프트웨어는 제한된 형태의 입력만을 받아들이고 수행할 수 있기 때문에 초보자들이 이러한 소프트웨어 응용 프로그램을 적절히 활용하는 것에는 많은 어려움이 따른다. 즉 소프트웨어의 제약에 맞추어 초보자가 자신의 의도를 반영하기 위해서는 소프트웨어가 지원하는 입력 모달리티로 변형한 후, 소프트웨어의 내부 언어로 번역해야한다. 이 과정은 초보자들이 학습하기에 큰 부담으로 작용하여 시작 단계에서의 진입장벽을 높이게 된다. 따라서 본 학위논문에서는 여러 모달리티간의 매핑을 활용함으로써 초보자들이 멀티모달 입력을 이용하여 자연스럽게 소프트웨어를 이용한 설계작업을 할 수 있도록 지원하고자한다. 이 접근을 통해 초보자들은 소프트웨어가 지원하는 입력의 형태로 변환하지 않고, 여러 모달리티를 활용하여 자신의 의도를 유연하게 표현할 수 있다. 또한 여러 모달리티로부터 얻어진 데이터를 혼합하여 소프트웨어의 디자인 태스크를 지원하는 인터랙션 위젯이 생성되어 초보자들의 작업을 직접적으로 지원할 수 있다. 본 학위 논문에서는 음성 및 클릭의 멀티모달 입력에 기반한 비동기식 UI 디자인과 웹사이트 스타일링의 태스크를 지원하여 위 접근법을 활용하는 기법을 제안한다.

핵 심 낱 말 인간-컴퓨터 상호작용, 다중모달 상호작용, 음성 사용자 인터페이스, 사용자 인터페이스 설계, 자연언어

Abstract

Software applications have the potential to expand the design capabilities of novices. Adequately leveraging these software applications, however, can be prohibitive for novices as the input that software is designed to accept and understand is restricted. Novices must be able to adapt their intended input according to the software's constraints: condense it into the limited input modalities supported by the software and translate it to the software's internal language. The burden and learning curve of this process introduces an additional barrier to entry for novice designers. Consequently, novices rarely reach the potential that software applications grant. This thesis proposes a high-level approach to lower this barrier: support multimodal input natural to novices, but computationally map the multiple modalities to enable design tasks through the software. Through this approach, novices can leverage multiple modalities to flexibly express themselves without considering how to map their input to the software's accepted input. By extracting and combining the rich information embedded in it, the multimodal data can be transformed into interaction widgets that afford the software's design tasks. In this thesis, I present techniques for applying this approach with multimodal input of voice and clicks, and mapping operations according to temporal and contextual dimensions. Then, I present two interfaces that employ these techniques to facilitate two design-centric tasks involving novices: asynchronous UI design by collaborative student teams, and website styling by end-users.

Keywords Human-Computer Interaction, Multimodal Interaction, Voice User Interfaces, User Interface Design, Natural Language

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Software applications have expanded the capabilities of designers. Design applications empowers users with sophisticated tools that perform intricate manipulations to streamline the design process. For example, users can easily crop out objects from images by simply clicking on objects using the magic wand tool in Adobe Photoshop. Beyond easing the process, programming code and software engines have allowed designers to create new types of interactive and dynamic artifacts that would not be possible in the physical world (e.g., websites and video games). Further, by connecting designers distributed across the world, communication applications such as instant messaging or video conferencing allow designers distributed to share knowledge and collaborate—unbinding the design process from time- and spatial-related limitations.

While software extends the limits of **what** designers can create, it also limits **how** they design. Specifically, how the user interacts with the software and, in turn, designs is dependent on what input the software is designed to accept and understand. The constraints the software imposes on input can inhibit designers from taking full advantage of the software's potential.

1.1 Constraints on Input

Software constrains input in two main dimensions: the modalities and the software's language.

1.1.1 Modalities

Modalities refers to the input devices and the accompanying data that novices can transmit to the software. In the physical world, designers can communicate with great expressiveness and flexibility through a wide variety of modalities such as speech, gestures, and expressions. While there are various input devices that support this diversity of modalities (e.g., microphones, video cameras), the keyboard and mouse are the most commonly accepted devices. Thus, most software support the modalities afforded by these devices: typing and clicks. To communicate through software, then, designers must condense the ample expression possible in the physical world to one that can be communicated through these two modalities.

1.1.2 Language

In addition to modality restrictions, software is limited by the language it uses and understands. Rich applications may contain a diverse array of features, each with its own name, parameters, and form of function. For users to adequately leverage these features, they must learn this language—what features exist, what they do, and how to use them—and design according to it. This is the same case with programming-based design—the user must be able to read and write the coding language. While software can ease and extend designers' abilities, its potential is dependent on the designers' understanding of the relation between their own language and the software's language.

1.1.3 Impact on Novices

Due to the input constraints, designers must be able to condense and translate their natural expression into the software's modalities and language. While experts learn to adapt their input, for novices who are additionally inexperienced with the design process, this can be an unforgiving barrier. Limited modalities can make it challenging for novices to express themselves accurately in their teams [1, 2] and solutions require effortful workarounds (e.g., taking and sharing multiple screenshots in a chatroom). Due to the richness of features and support, the internal language of software can be incredibly vast [3] with learning curves that can be prohibitive for novices. Therefore, while computer software have the potential of enabling anyone to learn to become a designer, this potential is rarely reached due to the limitations imposed by the software itself.

1.2 Thesis Contribution

This thesis aims to overcome the constraints imposed by software on the design process of novices. Instead of requiring novices to adjust their input to match the software, this thesis investigates how to support the modalities and language that are natural for novices. However, as the software was not designed for the novices' desired form of input, it cannot adequately afford for design tasks through the input. For example, communicating through voice messages may be more comfortable for novices but there is no support for searching for design information in audio recordings, or it can be easier for the novice to explain the task they aim to perform but the software will not understand their goal.

This thesis contributes a high-level approach to bridge this gap: support natural input through multiple modalities, and computationally map them against each other to enable designing through the software. Multiple modalities grant greater expressiveness and flexibility to novices, while also providing the software with more interaction data. The approach of this thesis proposes automatically processing and combining this rich multimodal data to produce interaction widgets that afford for the intended design tasks. In a sense, the interaction widgets act as a membrane between the novices and the software—bridging the capabilities from both sides.

The thesis contributes the application of this approach in two design-related contexts: (1) asynchronous collaboration of novice UI design teams, and (2) website styling by end-users. In both contexts, the approach was applied on two lightweight modalities, voice and clicks. Two dimensions for mapping the modalities were investigated: (1) temporal, mapping data from the modalities that co-occur in time; and (2) contextual, mapping the intended meaning behind the modalities' input. The technical contributions are the following:

1. Temporal Mapping
   (a) Technique to extract bidirectional links between audio and design artifacts by temporally mapping the user's voice and click input.
   (b) Interaction techniques that facilitate navigation through and within audio messages.
2. Contextual Mapping
   (a) Computational pipeline that processes a user's spoken request and the design artifact clicked to recommend plausible design edits that can satisfy the user's intentions.
   (b) Interface that supports exploration and iteration of style edits on live websites.

In addition, the thesis contributes findings from controlled user studies that demonstrate the benefit and advantages of the proposed technical contributions:

- Findings from a qualitative study that demonstrate that voice and clicks, and the temporal mapping of these can decrease burden for senders and receivers in asynchronous communication.
- Findings from a between-subjects study that reveal how novices can familiarize themselves with and perform design tasks through the contextual mapping of voice and click interactions.

Chapter 2. Related Work

This thesis aims to tackle challenges related to constrained input when interacting with software during the design process. The thesis primarily builds upon prior work on two related fields: computer-mediated communication, and natural language interfaces.

2.1 Computer-Mediated Communication

Computers can constrain the communication between humans according to three main factors: expressiveness, contextuality, and timeliness.

2.1.1 Expressiveness

Due to the limited expressiveness of text, a rich body of work has investigated how to incorporate more expressive modalities and multiple modalities into communication support systems. Incorporating voice or speech is a common approach due to the expressive nuances it possesses, such as intonation and loudness [4]. For example, "voice comments" have been integrated into a variety of popular applications—such as Microsoft Word [5] or Google Docs through third-party plugins [6]—and, to deal with their slow speed of consumption, multiple systems have been designed to automatically transcribe [7, 8] or summarize [9, 10, 11] voice comments. Beyond voice, SketchComm [12] increases expressiveness in ideation by providing a shared canvas onto which designers can freely interweave audio, photo, video, and hand-drawn sketches to express their ideas. Similarly, RichReview [13] and RichReview++ [14] record voice annotations alongside ink and pointing gestures on a tablet to allow collaborators to discuss around a document as they would with a physical document if they were co-present. Other work such as Video Threads [15] relies on video and audio as the main components of communication by allowing users to create threads of video messages, and FamilyStories [16] incorporates voice and physical actions to kindle a feeling of togetherness between family members distributed in place and time. As seen, prior work investigated how to incorporate more expressive and diverse modalities in computer-mediated communication. In addition to exploring how integrating the modalities of voice and clicks can enhance expressiveness in the design process, this thesis investigates how these can be combined to also facilitate communication.

2.1.2 Contextuality

When collaborating around a shared document, it is essential that the communication of the team is contextualized with the document to better maintain a shared understanding [2]. An essential aspect to this contextualization is effective referencing of items in the document. However, while referencing items or sections of a document can be as simple as pointing in a F2F setting, this task becomes challenging and complex in online situations [17]. The user must provide detailed descriptions or rely on workarounds such as taking screenshots to adequately express the context and prevent confusion. To reduce the cost of creating contextual references, several systems have been designed to facilitate this process in diverse application scenarios. For discussions surrounding multimedia, Korero [18] supports referencing through linking to multiple portions of a video and, on the click of a button, Snapstream [19] instantaneously

creates annotatable snapshots of a live stream. In a different domain, systems like chat.codes [20] and Callisto [21] have also been developed to support communication between programmers by enabling 'pointing' to code segments in chat interfaces. Instead of incorporating the document context into the communication channel through referencing, an alternative method to contextualization is to anchor the communication on the context itself. For example, the tools by both Churchill et al. [22] and Zyto et al. [23] allow for anchoring discussions on specific locations in text documents. Similarly, LemonAid [24] anchors question-and-answering communication on UI components of a web application to allow the answerers to provide more contextually adequate help. This thesis investigates how multiple, co-current input modalities can be merged to support both forms of communication contextualization: referencing and anchoring.

2.1.3 Timeliness

A major benefit of computer-mediated communication is that it supports asynchronous collaboration, allowing parties to discuss without being temporally co-present. However, the delays in communication can have negative repercussions. For example, previous work has demonstrated that delays can hinder productivity of teams and cause social ramifications such as team members more negatively judging their fellow team members [25] and the overall task [26]. It has also been shown that communication delays can increase communicative burden on both sending and receiving parties. For receivers, browsing through the large amount of messages left when they were away can involve significant effort [27] and, for sender, thinking about the burden placed on receivers can cause them to abstain from communicating [28]. The detrimental consequences of delays in communication has motivated multiple researchers to design interventions to mitigate these. For example, Avrahami and Hudson [29] devised a notification system which distinguishes messages that require the user's immediate attention, and Pielot et al. [30] identified features that could predict a user's attentiveness to text messages, which could help manage expectations regarding response times. However, if the message receiver is certainly unavailable, these approaches will not sufficiently address existing challenges. This thesis proposes interaction techniques that allow senders and receivers to use multimodal input to more easily produce and consume messages in asynchronous communication.

2.2 Natural Language Interfaces

Novices struggle to translate high-level design goals into software operations due to the vocabulary problem [31]—the language used by the user and the software do not match. Thus, empowering users to be able to design by simply stating their high-level goals has been a long-standing goal for HCI researchers. In this section, we cover natural language interfaces that aim to accomplish this in three different types of tasks: (1) help-seeking, (2) designing, and (3) coding.

2.2.1 Help-Seeking with Natural Language

To bridge the gap between their language and the software's, novices can rely on external support by asking through natural language. While search engines partially satisfy this need, they are unable to adequately understand user's needs or consider their specific context. Thus, a line of work has investigated how to connect novices to human helpers instead. Codeon [32] and MicroMentor [33] connect a novice developers and 3D modelers, respectively, with remote helpers who can provide rich and

synchronous assistance. Similarly, a body of work investigated how crowdworkers could be employed as assistants. Apparition [34] and SketchExpress [35] employs crowdworkers to generate user interfaces from the sketches and spoken descriptions that the user provides. However, this type of human support incurs a financial cost which may not always be practical. Instead of relying on human helpers, ReMap [36] and RePlay [37] instead rely on existing online help resources. Both systems consider the users' application context to provide adequate tutorial videos according to the users' natural language search queries. Beyond help-seeking, this thesis and the other lines of work covered below aim to allow users to directly perform tasks using natural language.

2.2.2 Designing with Natural Language

Due to the richness of the support they provide, various natural language interface have been designed to facilitate the use of the diverse features in design software. For example, Query-Feature Graphs (QF-Graphs) [38] and CommandSpace [3] jointly modeled natural language descriptions with feature names in design applications (e.g., GIMP and Photoshop) to help users identify features based on their needs. Other systems support the use of natural language concepts to search for design references or components—images [39], graphic designs [40], or 3D models [41]. Beyond searching, several systems generate design artifacts (e.g., images [42] or icons [43]) based on the semantic meaning of words. Leveraging whole expressions instead of only words, Crosspower [44] and PixelTone [45] decompose expressions into operations for animation authoring and image editing, respectively. This thesis expands on this line of research by investigating how natural language expressions can be mapped against users' clicks to support design editing in the context of web pages.

2.2.3 Coding with Natural Language

A crucial step in programming is coding—writing instructions in the form of machine-readable syntax. To lower the barrier to coding, substantial effort has been dedicated to bridge natural language and complex programming languages [46]. For instance, researchers have used semantic parsers [47] and bimodal models [48] to map natural language to code. Such techniques enabled systems that allow novice coders to quickly search for code snippets [49, 50], and non-coders to code small programs by demonstrating and describing tasks [51, 52]. Beyond mapping, a line of work has also developed techniques that take natural language as input and generate code—e.g., Python [53, 54], Bash commands [55], SQL queries [56], or API calls [57]. Recent advancements in natural language processing (NLP), and especially in large language models (e.g., GPT-3 [58]), have led to performance boosts in natural language-based code generation [59]. OpenAI's Codex [60], a GPT-3-based model, is able to generate basic games from a few natural language sentences [61]. In this same line of research, this thesis investigates how to leverage NLP models to interpret novices' intentions expressed in natural language intentions to facilitate designing through CSS code.

Chapter 3. Winder

3.1 Introduction

Teams commonly collaborate around a shared visual document (e.g., user interface (UI) design or presentation slides). This is an essential process in both the workplace and academic settings. For productive and successful collaboration, frequent communication is necessary to develop a shared understanding between team members [62]. However, for teams of students, communicating effectively and efficiently can be challenging. Students are generally novices who lack the design knowledge necessary to express themselves effectively when discussing their design. Additionally, differences in course schedules and the lack of shared office space frequently restrict them to collaborate remotely and asynchronously [63].

Communication-related burdens introduced by an asynchronous setting can discourage students from discussing with their teams. Most asynchronous communication channels rely on text (e.g., text messaging and Google Docs' comments [64]), but typing involves high physical and cognitive demand [65]. Also, although effective referencing is essential in developing a shared understanding [2], text makes referring to visual objects in the document challenging—as pointing while typing is impossible. Unfortunately, students may lack the knowledge needed to overcome the restrictions of text [66].

Furthermore, regular scheduling issues between students [67] may prevent them from frequently checking on their team's messages and documents. This introduces a significant delay in communication and team members' communicative needs may not be satisfied in a timely manner. In addition, when students do check on messages, they may have to look through a large volume of messages [27], and references might not be adequate as the objects referenced may have changed after the messages were sent [21].

As shown by our formative study, feeling uncertain about receiving a response on time, and the aforementioned burdens when producing and consuming messages may lead to communication breakdowns in student teams [68]. Although professional teams have organizational support to handle these breakdowns [62], instructors may be ill-prepared or time-constrained to adequately provide similar support to students [69].

In this paper, we propose a novel way of communication with multimodal messages of voice and clicks—referred to as *linked tapes*—as an alternative form of asynchronous communication. Linked tapes are multimodal messages created by simply speaking while *pointing* at relevant visual objects through clicks, interactions which could require less effort when compared to typing text messages. When a tape is produced, the current version of the document is stored to preserve the temporal context of the tape to enable *change awareness* [70]. Additionally, bidirectional links between objects and voice snippets are automatically generated by temporally mapping the modalities onto each other (Fig. 4.1). With the voice-to-object link, playing back a voice message can display relevant objects to allow the receiver to effortlessly understand the references in the message. With the object-to-voice link, selecting an object of interest can filter through numerous voice messages to retrieve only those relevant to the object and facilitate the receiver's navigation. The ease of producing linked tapes with the multimodal input and the support provided by the bidirectional links can vitalize communication in asynchronous teams and improve a shared understanding.

We actualized communication based on linked tapes in *Winder*, a plugin for the collaborative UI

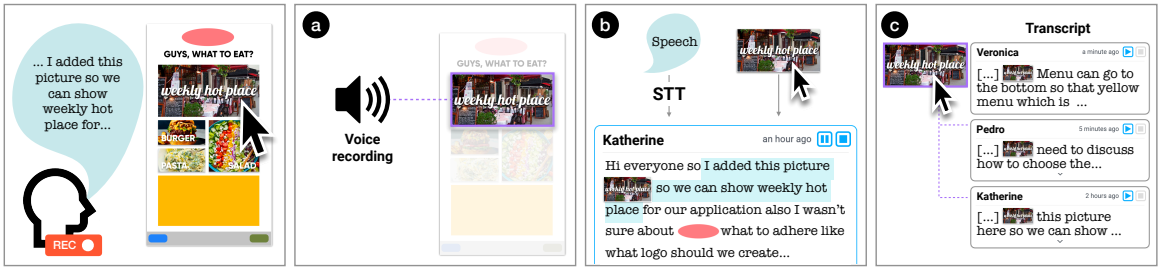

Figure 3.1: *Winder* allows for communication through linked tapes—multimodal recordings of voice and clicks on document objects. Winder supports three features for tape understanding and navigation: (a) highlighting objects on playback of voice recordings, (b) inline thumbnail images of objects on automatic transcripts, and (c) search of recordings based on objects.

design tool Figma. The plugin leverages the linked tapes' bidirectional links to implement three main features: (1) highlighting on playback (Fig. 4.1a), (2) inline thumbnails on transcripts (Fig. 4.1b), and (3) object-based search (Fig. 4.1c). In addition, Winder also aims to tackle the problem of communication delays. The plugin periodically prompts the user to produce linked tapes with the goal of preemptively obtaining information which may be needed by team members in the future. Tape-based communication could lessen the burden imposed by this approach—the effort of producing and consuming many messages. Thus, the drawbacks can be outweighed by the potential gains in shared understanding.

To investigate the effect of Winder on the collaboration process of asynchronous teams, we conducted a five-day study with eight teams of three students (N=24). Teams were tasked with designing the UI of a mobile application, which helps friends decide on what and where to eat. On the first day, they discussed ideas as teams and, on subsequent days, each team member worked on the design on different time slots. Our findings showed that the participant teams produced an average of 13.13 tapes and that the average tape length was 53.27 seconds. Analysis of survey responses revealed that, when compared to text messages, participants felt less burdened producing linked tapes due to the ease of speaking and clicking, and felt more confident that their messages would not be misunderstood. Participants also expressed that bidirectional links facilitated navigation through and within tapes, as well as their understanding of these tapes. Furthermore, the study results suggest that tapes recorded preemptively could allow for communication at hand without having team members at hand.

Our work contributes a novel multimodal asynchronous communication tool, *Winder*. Through lightweight interactions in production (i.e., click and voice) and bidirectional links for consumption, the system advances work in asynchronous communication by simultaneously decreasing burden for both senders and receivers—previous work facilitated either but not both. Furthermore, reducing at-the-moment burdens allows for an approach to tackle communication delays that would previously be overly burdensome: prompting users for preemptive recordings to satisfy future communication needs. As a secondary contribution, we present empirical findings that demonstrate the potential of Winder to reduce bilateral communication burden and overcome the detriment of delays in student teams.

3.2 Formative Study

To understand the challenges in communication between team members collaborating on a visual document in an asynchronous setting, we conducted semi-structured interviews with 10 undergradu-

ate students. We focused our investigation on student teams as they frequently collaborate asynchronously [63] and their collaboration experiences are meaningful as they allow for the learning of collaboration skills crucial in the workplace [71]. Additionally, student teams have little or no support available to handle the challenges of asynchrony—unlike teams in the workplace that may have workflows (e.g., Scrum [72]) or managers [62] in place to facilitate communication.

3.2.1 Interviews

We recruited a total of 10 undergraduate students (seven females and three males) at a technical university in South Korea. All participants had participated in at least one team project in their most recent semester in which the team members collaborated asynchronously on a visual document (e.g., UI design or presentation slides). We conducted one-hour long interviews in which participants were asked to reflect on their asynchronous collaboration experiences by freely looking through the previous interactions they had with their teams (e.g., chat logs or documents created). The interview questions mainly focused on three aspects: (1) what communication needs did students have while working asynchronously; (2) how the asynchronous setting affected their achievement of these needs; and (3) how failing to achieve their needs could impact their collaboration process.

Loss in Shared Understanding Led to Reworking or Subpar Outcomes

Initially, interviewees met synchronously with their team members through video conferencing tools—due to the COVID-19 pandemic—to discuss goals, tasks, and the assignment of these tasks. These discussions usually lasted approximately one hour and served to establish a shared understanding within the teams. After these discussions, each team member worked on their assigned tasks on their own time while communicating through familiar text messaging applications (e.g., Facebook Messenger). As the state of the visual document evolved with each member's contributions, interviewees frequently needed communication within their teams to understand what had changed and why. However, due to issues related to the team's asynchronous setting—which we discuss below—this communication would frequently not take place, which deteriorated the shared understanding of teams. As a consequence, work on the visual document would progress with misunderstandings remaining, which meant some team members had to redo their work later on or the team's outcome would be unsatisfactory. This finding parallels insights on remote work by Olson and Olson [68]—when building common ground, effort is required to resolve misunderstanding and, if this effort is too high, people may proceed without resolving them.

Burden When Producing and Sending Messages

Team members maintained a shared understanding by sharing progress updates or by asking questions about each other's work, but the process of producing and sending these messages was burdensome. Particularly, typing text messages required significant effort, especially when the message was referring to an object in the visual document. To prevent confusing team members, interviewees dedicated additional time thinking of how to write these messages and, in some cases, expended effort taking screenshots of the objects. In addition, interviewees also considered the perspective of team members on the receiving end when sending messages. Interviewees did not want to disrupt their team members and they also recalled on the burden they themselves had previously felt when receiving messages. Social costs related to the effort of producing messages and consideration of others when sending have also been identified

in online question and answering [28]. Due to the social costs or burdens, interviewees frequently hesitated to produce and send a message, or even completely withheld from doing so. However, unlike the professional teams studied by Bjørn et al. [73] that faced similar challenges, students had no managerial practices to encourage them to communicate despite the burden they felt.

Burden when Receiving and Consuming Messages

As mentioned previously, interviewees also felt burdened when they were on the receiving end of a message. Due to their schedules and priorities, interviewees were not able to frequently check their team's messages. This meant that, once they were ready to work on the visual document and check messages, they would be faced with a large amount of messages. Significant effort was required to read through all these messages and to understand them, if the messages were not clearly written or sufficiently detailed. If they were unable to read through all the messages, interviewees tried navigating through them to find useful information. But, as all messages are presented in the same way in text messaging applications, distinguishing important information within these messages was challenging. These findings are corroborated with those reported by Zhang and Cranshaw [27]. Thus, collaborating asynchronously placed communicative burdens on not only the sender but also the receiver.

Waiting or Working with Limited Understanding due to Communication Delays

Even if the interviewees overcame the burden of producing and sending a message and asked their team members about their work, team members may not be available to provide an answer. As the team members were working asynchronously, immediate responses to messages were the exception and not the rule. In these situations, some interviewees waited for a response, which could be frustrating and stall the team's progress. Other interviewees worked on the document with a limited understanding, which at times led them to redo their work later as it had been completed with a misunderstanding of the team's goals. Therefore, communication delays impacted the overall productivity of teams as they either halted the effort of team members or caused effort to be wasted.

Our findings expand knowledge on the challenges of remote and/or asynchronous collaboration by revealing difficulties specific to the context of students collaborating asynchronously on a visual document. We gained the insight that team members refrain from communicating in an asynchronous setting due to their own burden and the possibility of burdening their team members. Therefore, to develop and maintain a shared understanding in teams, reducing the effort of producing and consuming messages is crucial. With these communicative burdens reduced, we hypothesize that preemptively asking team members to explain their work while they are working can increase team productivity despite communication delays. If the explanations are created in advance, team members would not be affected by other members being unavailable to provide explanations as the explanations would already be at hand.

In accordance to the insights from our formative study, we set the following three design goals:

- DG1: Facilitate the production of explanations and referencing of objects in the visual document.
- DG2: Support navigation to required explanations, and the understanding of the explanations and their context.
- DG3: Preemptively obtain the user's explanations of their work and decisions while they are working on the visual document.

3.3 Winder

To instantiate our design goals, we first establish a concrete context regarding the type of team and task we aim to support. We consider temporary teams in which members have vastly different schedules—e.g., student teams or cross-institutional research teams. These teams have no set communication practices, which can lead to unclear and infrequent communication, and are unable to allocate substantial blocks of time to work synchronously. In terms of the task, we focus on UI design. Unlike other visual documents like presentation slides that may incorporate significant amounts of text and have inherent structures (e.g., slides are in chronological order and usually have titles at the top), UI designs are highly visual and open-ended in terms of structure. While we believe in the generalization of our design goals, setting a concrete context allows us to design more effective support.

Based on this context and our design goals, we present *Winder* (Fig. 3.2), a system to support the asynchronous communication of team members collaborating on a UI design document. Winder is a plugin built on top of the collaborative interface design tool Figma. Figma [74] was chosen as the base for our plugin as it is a free service that sees widespread use, and for its flexibility with regards to plugin development.

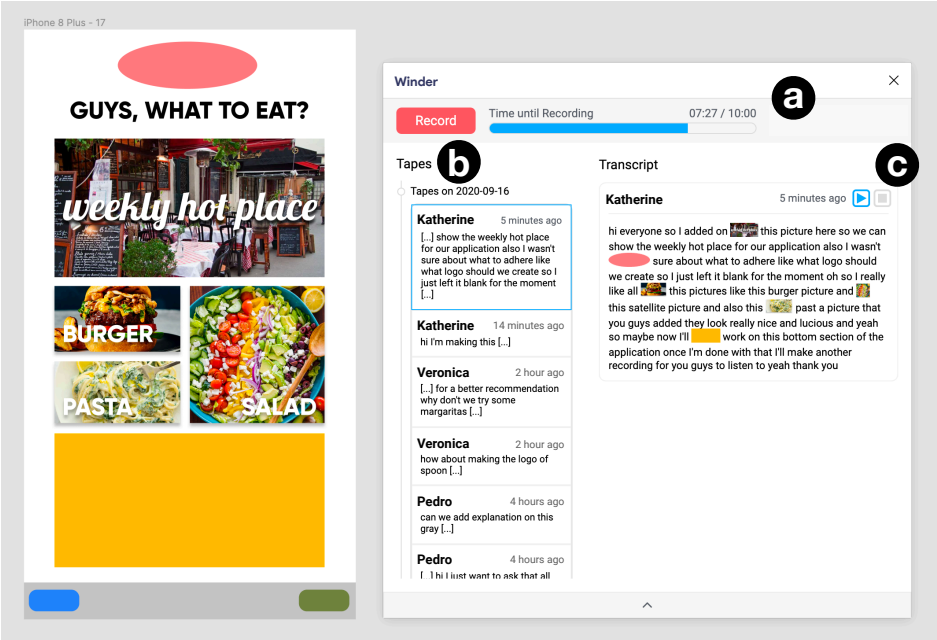

Figure 3.2: Winder (right) is shown on top of the Figma UI design document. The main components of the plugin's interface are (a) the top bar, (b) the list of linked tapes, and (c) the transcript space.

3.3.1 Overview of the Plugin

Winder supports asynchronous communication through *linked tapes*—multimodal recordings of voice comments and clicks on UI design objects. The plugin's screen consists of three main components: (a) the top bar, (b) the list of linked tapes, and (c) the transcript space. The top-bar (Fig. 3.2a) contains the "Record" button, which the user clicks to start a recording, and a timer, which alternates between states—i.e., "Time until Recording", "Recording" or "Playing". The list of linked tapes (Fig. 3.2b) shows all the tapes recorded by a team, grouped by the days in which they were recorded. Each tape entry

displays the user that recorded it, how long ago it was recorded, a "NEW" label if the user has not yet played that tape, and a short fragment from the transcript of the tape's voice recording. The fragment is extracted from the 10 seconds with the highest average amplitude in the voice recording. Clicking on a tape on the list displays the tape's full transcript on the transcript space (Fig. 3.2c). The top-right of the displayed transcript shows media control buttons to allow for playing, pausing, and stopping of a tape.

3.3.2 Recording Linked Tapes

Although the user can freely decide when to record a tape, they are also prompted to do so every 10 minutes (DG3). While they are working, the timer on the top-bar counts down from 10 minutes—this interval was chosen based on pilot studies—and, once the timer runs out, the user receives a notification at the bottom of Figma asking them to record a tape. To record a tape, the user clicks on the "Record" button. This stores the current version of the document, and starts recording audio and clicks. The user can then simply comment on their design through their voice and make references to objects in the UI design document by simply clicking on them (DG1). Inspired by the work by Joshi et al. [33] which showed that imposing a 3-minute limit on help sessions could lessen the burden on the communicating parties, we also limit the length of tape recordings to 3 minutes. The amount of time that has progressed since the start of the recording is shown in the top-bar timer. Additionally, our pilot studies revealed that 1 minute is usually sufficient time for a recording. To further encourage the user to be even more concise in their recordings, the timer also begins to blink after 1 minute of recording.

3.3.3 Playing and Navigating Through Linked Tapes

For each linked tape, Winder generates an automatic transcript of the voice recording. In addition, Winder identifies all the snippets in the tape's voice recording where an object in the UI document had been clicked on and remained selected. By identifying these snippets, the system generates bidirectional links between voice snippets and objects. These links serve as the basis for three features which facilitate the understanding of tapes and the identification of desired information (DG2): (1) highlighting on playback, (2) inline thumbnails on transcripts, and (3) object-based search.

Object Highlighting on Voice Playback (Fig. 3.3)

To play a linked tape, the user can click on the "Play" button on the top-right of a transcript. This shows to the user the version of the document when the tape was recorded, and starts playing the audio of the voice recording. As the audio of the voice recording is playing, Winder highlights the design objects that the creator of the tape had clicked on as they had been clicked during the tape's recording. Also, the object remains highlighted for the duration that it remained selected during the recording. This allows the user to easily distinguish, as they are listening to the voice explanation, what objects the creator of the tape had clicked on and was referring to. Highlighting is performed by making all other objects in the UI design more transparent, such that the highlighted object is more prominent. At any time, the user can pause the tape to pause the audio and highlighting to freely explore the previous version of the document. They can also stop the tape and go back to the current version.

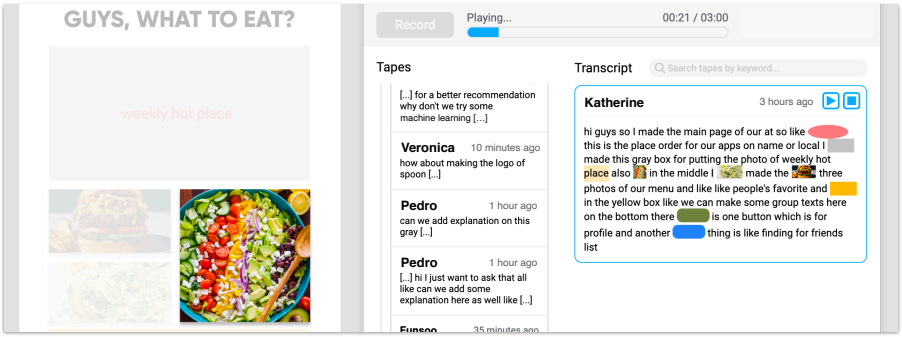

Figure 3.3: Winder is playing a linked tape, which plays the audio of the voice recording and highlights objects in Figma at the moments they were clicked and selected during the tape's recording (thus the picture of a salad is currently highlighted).

Inline Thumbnail Images on Voice Transcripts (Fig. 3.4)

For each linked tape, Winder embeds thumbnail images of the design objects into the transcript of the voice recording. These thumbnail images are obtained through the Figma API and are shown inline with the words of the transcript on the moments the objects were clicked during recording. With the thumbnail images, the user can see all the objects that were clicked during a tape's recording and also see when they were clicked. The user can quickly browse through tapes by looking at the thumbnail images in each tape. Also, the user can navigate to points in the voice recording by clicking on sections of the transcript. This allows the user to use the thumbnails as anchors for their navigation within a tape.

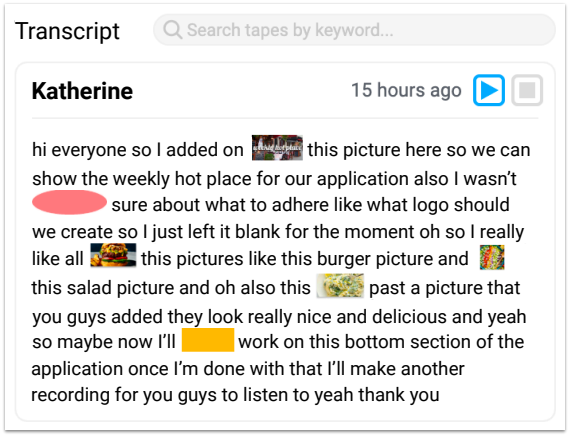

Figure 3.4: Winder presents an automatically generated transcript for each tape. Thumbnail images of the objects clicked on during the recording of the tape are shown in line with the transcript text to show when they were clicked.

Object-based Search of Voice Recordings (Fig. 3.5)

Winder also allows the user to search for voice recordings based on design objects of interest. Clicking on an object in the UI document retrieves all the tapes in which that object was clicked on during recording. The plugin then displays the retrieved tapes as a list in the transcript space. Each

entry in the list shows the segment of the tape's transcript during which the object had been clicked on and remained selected. The user can skim through this list to quickly get a grasp of all the information related to that object. Also, if the user finds a tape with interesting information, they can display the full transcript by clicking on the expand button at the bottom of the entry or play the tape with the media controls on the top-right.

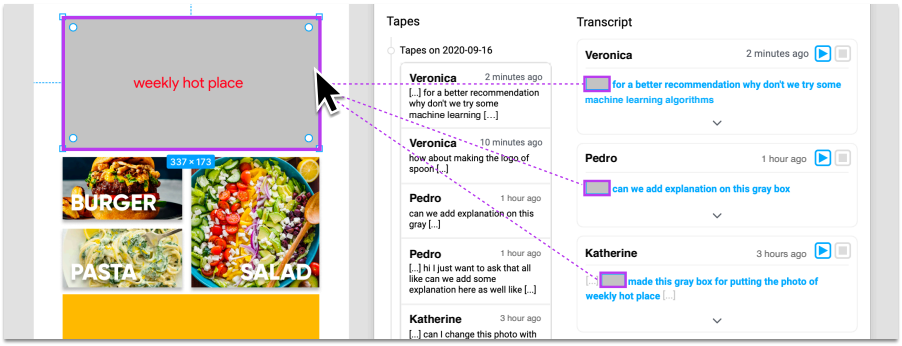

Figure 3.5: The user clicked on the gray rectangle object in Figma which displayed in Winder all the tapes in which that rectangle was clicked-on during the recording. In the list of tapes, each tape entry shows a transcript snippet of the voice recording during which the gray rectangle was selected.

3.3.4 Implementation

We implemented the interface of Winder with TypeScript, ReactJS, and CSS. For the backend of the system, we used Node.js for the server and MongoDB for the database. Due to the fact that microphone access is not allowed from within the frame that houses the plugin in Figma, we used a separate browser window to record audio and deliver it to the server. Socket.io [1] was used to allow the plugin to communicate with the browser window used for microphone access. The Google Cloud Speech-to-Text [2] service was used to generate the transcripts from the voice recordings.

3.4 Evaluation

We designed a five-day user study with 24 students assigned into teams of three to investigate how Winder affected the communication within teams asynchronously collaborating on a UI design document. Our study aimed to examine the usage of Winder, as a whole, within the context of close-to-real team dynamics. Thus, the study was not comparative as a valid comparison would require controlled lab studies evaluating each system component in isolation. Through the study, we aimed to answer the following questions:

- How does multimodal communication based on voice and clicks affect the burden of sending messages when compared to typing text?
- How do bidirectional links between the visual objects and voice recordings support the identification and understanding of information?

[1] https://socket.io/
[2] https://cloud.google.com/speech-to-text

- What type of content do preemptively recorded linked tapes contain and how do these impact team collaboration?

3.4.1 Participants

We recruited 24 participants (age M=22.25 and SD=2.98, 9 females and 15 males) who all reported to have no previous experience in UI design. Participants included 20 undergraduate students, 2 graduate students, and 2 industry workers. The industry workers were allowed to participate in the study as they worked in fields unrelated to design and expressed an interest in learning design in their spare time. Each participant was compensated KRW 70,000 ($59.00) for participating for approximately one hour every day for five days (total of five hours). Participants were randomly assigned into teams of three to produce eight teams in total. We formed teams of three as the formative study showed that this was a common size for student teams in course projects. The assignments were adjusted to ensure that none of the members in a team had previously known each other. Recruitment was carried out through online forums. In addition, participants were selected based on their fluency in English—judged through participants' backgrounds or examination scores—to ensure that participant teams did not face communication challenges due to language barriers and the accuracy of speech recognition.

3.4.2 Study Procedure

Due to the COVID-19 pandemic, the entire study was conducted remotely through Zoom [3]. On Day 1 of the study, participants were invited to a video call and introduced to the overall study procedure and the task. This task was to design the UI for a mobile application, which solved the following problem: "deciding on what and where to eat with friends is difficult and time-consuming." This problem was chosen from a list of past student projects in an "Introduction to HCI" course at our institution as we believed that it could be a relatable problem for participants and motivate engagement. To further motivate them, participants were also informed that the team with the best design—in terms of usefulness, uniqueness, simplicity, consistency, and completeness—would receive an additional KRW 5,000. Then, each team was sent to a separate video call room in which members were asked to first introduce themselves and then discuss about their design. A shared document was provided to each team which laid out discussion steps for the team to follow—brainstorm ideas, decide on the application's main screens, and distribute tasks. Team members could also make notes of their discussion in that shared document.

On Days 2 to 5 of the study, participants worked on their team's shared Figma document on different time slots to those of their team members. On Day 2, participants were provided with a short tutorial on Figma. Each study day consisted of 45 minutes of design work (including the duration of the tutorials) and 15 minutes of completing a post-survey. The survey asked participants about what they worked on that day and their perceptions on the communication they had with their team members. On their final day of the study (Day 5), participants also responded to an additional survey, which asked them about their overall collaboration and communication experiences. Throughout the four days, participants were allowed to communicate with their team members through the Google Docs document provided on Day 1, the comment feature in Figma, or a chatroom in KakaoTalk—a locally popular messaging application in South Korea. Other team members—those not working at that moment—were allowed to respond through these channels, but were not expected or required to do so.

[3] https://zoom.us

Since our participants had no previous experiences in UI design collaboration against which they could evaluate their experiences with our system, we aimed to provide them with traditional collaborative experiences during the study. For half of the designing days, they were not allowed to use Winder and were limited to traditional communication tools. Since communication patterns in early and late design stages can differ [75] and this can affect usage of Winder, we aimed to understand system usage throughout the whole design process by allowing half of the participant teams to use the system only during early stages and allowing the other half of teams to use it during later stages. Thus, half of the teams were allowed to use Winder during Days 2 and 3 of the study, while the other half used it only during Days 4 and 5. On the day they would start using the plugin (Day 2 or 4), participants were provided with a tutorial on Winder.

3.4.3 Measures

Various types of data were collected during the study, including: survey responses, the content of linked tapes produced, chat messages exchanged, text added to the Google Docs, and comments left in Figma. Survey questions asked participants about how and why they sent messages, how and why they tried to understand their team members' actions and intentions, and why either of these processes were easy or difficult. Participants' responses to these questions were used to determine their perceptions on burden during their collaborative experiences. For the linked tapes, the voice recordings were manually transcribed due to the inaccuracies in automatic transcription. For the qualitative analysis, two of the authors conducted a thematic analysis of all the survey responses to derive the main findings of the study. The other types of data were used to verify and supplement these findings.

Also, the same two authors open coded the content of the linked tapes to categorize them based on their purposes. The open coding process involved developing and revising categories by looking through the data, and individual coding. The individual coding resulted in an average Cohen's kappa of 0.56 which indicates moderate agreement—the lowest kappa value was 0.43 and the highest was 0.69 (all within the moderate agreement range). As the initial agreement between coders was not significantly high, the coders met to discuss, revise the categories' definitions, and resolve conflicts to reach a complete agreement on the codes.

3.5 Results

Due to difficulties in objectively measuring burden or success in collaboration, our findings instead focus on the experiences and perspectives of our participants. Findings from our study revealed how voice and clicks, and the bidirectional links in *linked tapes* could facilitate asynchronous communication. Additionally, prompting the user to record tapes was found to potentially satisfy some future communication needs, clear misunderstandings, and even allow team member to better coordinate their design work.

3.5.1 Linked Tapes Statistics and Categories

A summary of the statistics for the linked tapes produced during the study is presented in Table 3.1. Participants recorded a total of 107 tapes but 2 were not recorded properly due to technical issues with the participant's computers. These 2 tapes were not included in the analysis. The table shows that, on average, tapes were recorded around every nine minutes and their length was under one minute. This

Statistics on Tapes	Mean	STD	Max	Min
Tapes by Member (number)	4.38	2.04	9	2
Tapes by Team (number)	13.13	3.14	18	10
Objects Clicked (number)	3.14	3.39	17	0
Length (seconds)	53.27	32.98	167	9
Interval between (seconds)	529.85	379.02	1827	72

Table 3.1: Statistics for the number of tapes created by members or teams, number of objects clicked on in each tape, length of tapes, and intervals between tape recordings.

indicates that participants generally followed the system's suggestions for tape length and frequency. The linked tape categories derived through our open coding process are shown in Table 3.2, with details on each category. As seen in the table, describing, explaining one's own work, and coordinating tasks to complete ("Describe", "Justify", and "Coordinate" in Table 3.2) were the three most common purposes behind tapes during the study. However, tapes were also used for purposes beyond these. For example, some participants (T4M1, member 1 in team 4, and T2M2) mentioned leaving feedback on their other members' design through tapes which is also evidenced by the "Feedback" category. Further, as shown by the "Feedback" and "Clarify" categories, participants also engaged in back-and-forth communication by responding to each other's tapes. For instance, T6M1 mentioned: *"I left messages to answer back to the questions left by my teammates [and say] how I tried to solve their design concerns."* Interestingly, the "Clarify" category was the least common which could possibly be due to misunderstandings being preemptively clarified by the "Describe" or "Justify" tapes.

Tape Categories	Description	Example Snippet	Percentage (%)
Describe	Describes what the explainer added, modified, or deleted.	**T1M3:** *"These two buttons I am modifying it for a better image."*	83.8
Justify	Explains the design rationale behind changes.	**T2M3:** *"I thought that feature is only necessary for this page only and you don't need that for results and reservation page so I got rid of those."*	46.6
Coordinate	Assigns tasks to others or reports on future tasks to be completed.	**T4M2:** *"I think it might be a good idea if you change the text for these to what you did here if you have the time to do it."*	40
Build on others	Invites others to work on one's own work or mentions having worked on another's work.	**T6M1:** *"[T6M2] told me that [they] added this button here what I actually erased it so you can't see."*	16.19
Feedback	Provides or asks for feedback on work.	**T7M3:** *"Instead of leaving the comments, it [would be] better if there's just the picture, the star rating, and the user comments."*	14.28
Social	Expresses social or emotional comments (e.g., praise, concern).	**T3M3:** *"I feel bad [as] I'm leaving too much work for [T3M2] but my time ran out."*	12.38
Clarify	Provides or asks for clarifications on completed work.	**T7M1:** *"My intention [of making] this page, this week's hot place, is to show some list[s] of some new restaurants and fancy restaurant."*	8.57

Table 3.2: The categories of tapes by the purpose of their content. These categories were not mutually exclusive—a tape could belong to multiple categories. Each entry in the table includes the name and description of a category, an example transcript snippet from a tape in that category, and the percentage of tapes which were coded with that category out of all tapes.

3.5.2 Evaluating the Modalities in Winder

Participants felt ease in communicating with the two modalities, clicks and voice, supported in Winder but also pointed at several drawbacks related to these.

Pointing through Clicks

Most participants expressed feeling ease when pointing at or referring to UI objects with clicks. For example, T5M3 felt comfort when referencing design objects through clicks instead of writing words: *"It's pretty convenient to just click and talk rather than using vague terms [in text messaging]."* When discussing burdens related to referencing, several participants (T2M3, T4M1, T4M3, T7M1, T8M1, and T8M3) considered the perspective of the message receiver. All of these participants mentioned how misunderstanding references is possible with text but, by using clicks, they felt more assured that receivers would know what the messages are referencing. For example, T7M1 expressed: *"[Clicking] was the best way to deliver explanations without [confusing team members]."* Thus, due to the lower perceived effort and perceived uncertainty, pointing through clicks can potentially decrease feelings of burden when referencing. However, as G1P2 mentioned, referencing objects with clicks could be "unfamiliar". This led to several participants to hover on objects—instead of clicking—which made the resulting tapes more confusing as deictic references (e.g., "this" or "that") would not be accompanied by an object.

Expressing through Voice

Participants generally reported feeling less burdened when speaking a message instead of typing. T2M3 mentioned how recording linked tapes was easy for her as she could just *"talk instead of typing"*, and T4M1 likened voice recording to having a *"casual conversation"* with her team members. While most participants found voice easier than typing, participants were polarized in terms of which they preferred. To illustrate, T7M3 expressed that, for him, *"audio messages are almost always better than text messages"* while T5M3 mentioned how he *"prefers text over talking"* due to the awkwardness of recording his voice.

This polarization could be attributed to speaking-related burdens. A couple of participants felt conscious of their pronunciation as English was not their first language: *"My pronunciation is really bad, so I do not think that the system [will transmit] my message clearly."* (T8M2). Pressure to speak coherently was also felt by multiple participants, with some feeling that they had to improve on this: *"I am not sure if I explained everything well... I might have to improve on the flow of my recordings."* Previous work [14] also reported on self-consciousness due to pronunciation, but not on that due to coherence. This could be attributed to the time limit on voice recordings imposed by our system. Despite the initial burdens, participants reported gradually becoming accustomed to using their voice. In particular, for example, T8M3 continued sending voice recordings in his team's chatroom on the days he could not use the plugin.

Interweaving Clicks and Voice

Overall, communicating through both clicks and voice was unfamiliar for various participants, and simply familiarizing themselves to this was time-consuming: *"It took me really long to [...] get used to the plugin."* (T3M1). Nonetheless, several participants reported becoming accustomed to this form of communication within one study day (less than one hour). For instance, T2M1 mentioned how he became *"less anxious about [creating] messages"*. One participant (T5M2) became so accustomed to

Group	Day 2		Day 3		Day 4		Day 5		Total	
	Tapes	Other	Tapes	Other	Tapes	Other	Tapes	Other	Tapes	Other
T1	739	0	1016	0	-	185	-	81	1755	266
T2	-	0	-	165	959	0	1275	0	2234	165
T3	-	757	-	854	702	294	655	305	1357	2210
T4	863	28	1154	164	-	426	-	310	2017	928
T5	361	135	398	193	-	50	-	188	759	566
T6	-	50	-	87	577	35	928	0	1505	172
T7	-	0	-	49	365	0	615	0	980	49
T8	255	0	551	46	-	184	-	209	806	439
								Average	1426.63	599.38

Table 3.3: Total word counts in the transcripts of tapes recorded and other communication channels for each team and study day. Days in which each team used Winder are filled in green or yellow. The average total number of words communicated by a team through linked tapes was 1426.63 (max=2234, min=759, SD=554.15) and through other channels was 599.50 (max=2211, min=49, SD=708.88).

the modalities that, once they could not rely on them in the second half of the study, they felt more alone without them: *"It felt more like I was working on my own now that I couldn't leave detailed voice messages."* Also, it is possible that the perceived lower burden in communicating with the two modalities encouraged all participant teams, apart from T3, to mostly send information through linked tapes (see "Total" in Table 3.3). Our analysis in Table 3.3 focuses on the number of words, instead of the number of messages/tapes, due to the high variance in their amount of content—the shortest message/tape contained one word while the longest contained more than 100. Thus, measuring the number of words could better represent the amount of information transferred within teams [76].

3.5.3 Bidirectional Links for Navigation and Understanding

The bidirectional links in Winder's linked tapes support three main features: (1) highlighting on playback, (2) inline thumbnails on transcripts, and (3) object-based search.

Facilitating Understanding with Object Highlighting

Without the plugin, one participant (T3M1) explained how he had to manually map discussions onto the related objects in the UI design: *"I went through the chat room discussions one by one and tried to match them with the corresponding [objects in the] UI design."* On the other hand, with the highlighting of objects when playing back recordings, participants noted how understanding what members were talking about felt easier: *"I could [easily] see what they were talking about so I quickly understood their [messages]."* (T6M2). Several participants (T2M2, T4M1, T4M3, T5M2, and T8M1) also felt that the highlighting allowed them to more precisely pinpoint what they should focus their attention on and track document changes: *"The recordings were useful to highlight exactly which parts had changed."* Additionally, some participants (T3M1 and T6M1) expressed that the highlighting made them feel, at least momentarily, that their team members were co-present: *"With the voice recordings and the feature that showed what the users clicked on as they talked, it was as if we were working together."*

Thumbnail Images to Support Navigation Through and In Recordings

Our findings reveal that some participants were indeed able to use the additional information provided by these thumbnails when navigating to points of interest in voice recordings: *"That was helpful because it allowed me to listen to what I wanted to hear [and reduce] time lost [..] listening to all the boring [recordings]."* (T1M2). When navigating through different tapes, T3M1 explained how the thumbnails gave him an overall sense of what the recording was about at a glance: *"[You could] look right away at the [UI objects] that were clicked on in one transcript."* Beyond simply perceiving what was clicked on, T5M3 indicated how he could also gain a high-level view of his team's design with the thumbnail images: *"It also gave me a general sense of [our team's UI] themes."* However, as noted by T5M2 and T7M3, the shapes of the objects affected the usability of the thumbnails as the images were resized to match the height of the text.

Potential to Enhance Productivity and Prevent Conflicts Through Object-based Search

Participants T5M1, T8M2, and T8M3 mentioned how object-based search could potentially increase their productivity by helping them quickly find the design rationale behind objects or identify what object-specific tasks needed to be completed. T7M1 and T7M3 used the feature to have a conversation anchored on a set of icons to clarify the purpose of those icons. Participant T8M2 also noted that the feature could increase his productivity as it helped him quickly find information on an incomplete object and he later completed it based on that information. Besides these deliberate uses, participant T2M3 mentioned how the feature unexpectedly showed her a recording, which allowed her to recall information: *"When I was changing [a group of four icons], I went back to listen to the message that was recorded previously."* Additionally, as pointed out by T1M3, object-based search allowed him to prevent a potential conflict in his team: *"There were certain parts that I wanted to modify and, if I clicked on those parts and there were recorded comments about [related future plans], I would not delete or edit those features in my own way."* This object-based search, however, was not used frequently by participants. Participants T1M2, T5M3, and T7M3 all mentioned not using the object-based search as they only had to listen to a small number of tapes, but saw how it could be useful if their designs were more complex.

3.5.4 Content and Effect of Preemptively Recording Linked Tapes

Our study results showed that teams used linked tapes for diverse purposes and that they could reduce some of their needs for direct communication. In turn, this could help with the coordination within teams and could also encourage team members to cooperate by working on top of each other's work.

Satisfying Communication Needs

A couple of participants (T7M2 and T1M1) explicitly stated that, when they had tapes recorded in advance by team members, they did not have to directly communicate with their team. T1M1 explained that the preempted tapes helped her gain an understanding of her team's work *"without chatting directly"* with her team members. Similarly, T6M3 noted that a team member assigned him a task on a tape so he could work without talking to his team: *"It helped me understand what I had to do. For example, one of our members left a message saying that I had to create a new page containing user information."* In addition, for some participants, listening to preempted tapes prevented misunderstandings and the potential back-and-forth in communication needed to resolve these confusions: *"The recordings left behind*

by my group members helped clarify some of the misunderstandings or confusions that I had." (T2M1). Furthermore, we observed that, on the days in which they had the plugin, all eight participant teams relied mostly on the linked tapes to communicate, with three teams (T1, T2, and T7) relying solely on the tapes (see Table 3.3).

Impact on Team Collaboration

Preemptively recorded linked tapes were used by some teams to coordinate their efforts. Half of the participants detailed how the tapes allowed them to identify remaining tasks and decide on new ones. Participant T2M3 mentioned: *"Understanding [my teammates'] intentions really helped me guideline what I needed to work on and how I could improve the [UI screens]."* Some participants also coordinated what "territories" [77]—sections of the document—they should or should not modify. Particularly, T1M3 kept himself from modifying his team members' work if it had not been allowed explicitly: *"It prevented me from deleting/editing [my team members'] work without their consent."* On the other hand, as seen by the "Build on others" category in Table 3.2, participants also welcomed team members into their own "territories" so that they could cooperatively iterate on specific parts of the design.

Beyond supporting their low-level task coordination, several teams used preempted tapes to support their collaboration at a higher level. For example, as noted by T4M3 and T5M1, listening to team members' tapes, allowed the team to maintain design consistency. As the task was designing a UI, it required maintaining a shared direction on what the application does in addition to what the design looks like. For this purpose, preempted tapes were also useful. T6M1 left a tape in which she clicked on design objects to illustrate the application's user flow: *"Using the [plugin] it was easy to show the decision-making journey, because I can click on icons in chronological order and explain the journey step by step."* The content of recorded tapes showed that more than a third of participants (N=9) also illustrated user scenarios in their tapes.

Influence on the Social Factors of Teams

Beyond supporting work-related aspects, preemptively recorded tapes were also used to positively affect social factors—such as confidence, motivation, and trust. One participant (T8M2) mentioned how the nuances of voice helped him gain a sense of how confident his team members felt: *"By listening to recordings on the plugins, you can figure out [...] what they feel confident about, based on their tones."* Three participants (T1M2, T2M1, and T8M3) expressed feeling more motivated to work on their UI design after listening to team members' tapes: *"Understanding [my team members] actions and intentions was fun somehow and made me work harder."* (T1M2). Also, by increasing understanding of team members' intentions, preemptively recorded tapes could also increase trust: *"Thankfully, I've understood that they all had their own plans, which made me trust in them."* (T1M3).

Burden of Prompting and Preempted Tapes

While preemptively recording tapes was beneficial, several participants expressed feeling burdened by these due to several reasons. Firstly, the timer, which notified members to record, pressured some of the participants (T2M1, T3M2, T5M3, T7M1, and T8M2). T8M2, for instance, mentioned: *"Even though the time limit does not have to be obeyed, it still made me feel a huge pressure to [record] any type of comment."* Similarly, T5M3 suggested having an option to personally change the timer length: *"An option to change the timer [length] would be helpful because the work segments everyday are not*

going to be same length." In addition, three participants (T3M2, T4M3, and T5M2) felt concerned that their recorded tapes would burden their team members due to their content: reminding on how much work is left (T5M2), assigning tasks (T3M2), or providing feedback (T4M3). Some participants (T1M3, T2M1, and T4M2) also felt pressured after listening to the tapes left by their team members. For example, T4M2 expressed that listening to his teammates *"preciously"* describe their design burdened him to work harder. However, other participants (T1M3, T4M3, and T8M1) also reflected positively on pressure as they argued that it is needed for successful collaboration.

We observed that each team in the study showed unique patterns of communication as well as collaboration outcomes. In this section, we present in-depth case studies of two representative teams from the study. These teams were selected as their patterns of collaboration and communication revealed Winder's strengths and weaknesses. Through a grounded analysis, we investigated each team's cases along two dimensions: (1) working patterns, and (2) communication uses (with a focus on their use of Winder).

3.6 Case Studies

We observed that each team in the study showed unique patterns of communication as well as collaboration outcomes. In this section, we present in-depth case studies of two representative teams from the study. These teams were selected as their patterns of collaboration and communication revealed Winder's strengths and weaknesses. Through a grounded analysis, we investigated each team's cases along two dimensions: (1) working patterns, and (2) communication uses (with a focus on their use of Winder).

3.6.1 Team 1: Narrow Use of Winder Limits Improvement but Nonetheless Boosts Productivity

Team 1's initial discussion (Day 1) seemed to impact their communication quantity and purposes during the rest of the study. All the team members expressed satisfaction with that discussion in their survey responses as *"each feature was thoroughly discussed"* (T1M3) and they divided their individual tasks *"absolutely"* (T1M2). Members mostly kept to their own assigned screens, and most of their tapes focused on explaining edits and announcing future tasks they planned to complete. This is reflected by the categories of their tapes: all tapes (100% out of 18 tapes) were of the "Describe Work" category and 44.44% of the "Coordinate Tasks" category (both percentages were higher than the averages shown in Table 3.2). Team 1's members rarely used the tapes to provide each other with feedback (11.11% of "Feedback" category) or to invite others to work on top of their own work (11.11% of "Build on Others" category). As T1M3 mentioned in frustration, there was *"no sense of teamwork"*. Even though Team 1 did not use Winder for diverse purposes, its unavailability on Days 4 and 5 impacted their productivity. Without the plugin, communication virtually halted—aside from two conversations initiated by T1M3 (see row "T1", columns "Day 4" and "Day 5" in Table 3.3). As mentioned by T1M1, this made it "hard to know" what else she could do. Overall, due to the success of the discussion on Day 1 and the initial availability of the plugin, Team 1 still designed a UI which all the team members felt highly satisfied about. However, as noted by T1M3 on his last day, their narrow use of Winder prevented the team from further developing their original ideas—they only designed features that were discussed on the first day.

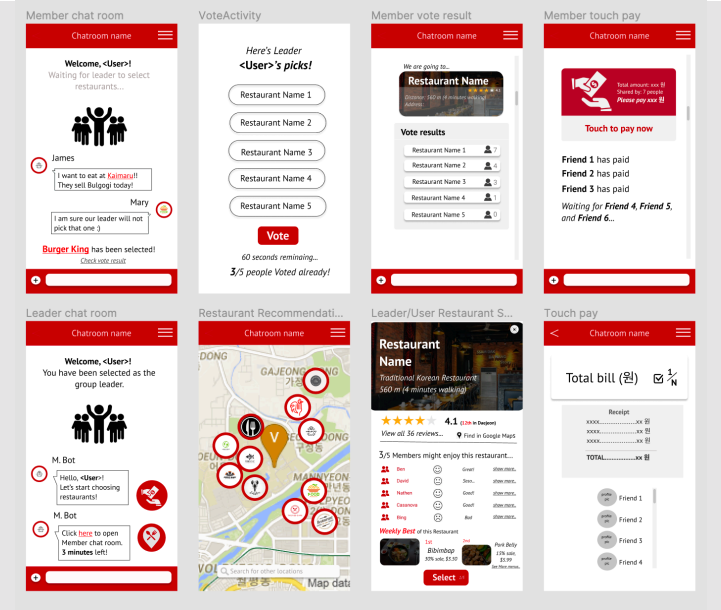

Figure 3.6: The final UI of the application designed by Team 1. They created an app named 'Meets' which allows groups to create specialized chatrooms for searching restaurants, voting, and splitting the bill.

3.6.2 Team 3: Frequent Tape Recording could Remedy the Detriment of Unequal Contribution

In contrast to Team 1, Team 3 frequently communicated through various channels (e.g., Google Docs, chatroom, and Winder), but still had problems reaching a shared understanding. T3M2 was the team's 'de-facto leader': she set the design layout and decided on tasks. The two other members—T3M1 and T3M3—worked based on her design. To communicate, T3M2 mainly relied on the team's Google Docs to write detailed updates. This accounted for Team 3 having the highest usage of other channels among participant teams (see "Total" in Table 3.3). Both T3M1 and T3M3 expended a significant amount pf time understanding T3M2's updates and were unable to work much during the limited session time. This problem was partially remedied once Winder was provided as T3M1 and T3M3 could easily understand T3M2's updates in the context of the document. However, T3M2 preferred *"writing memos to voice messages"* and kept using Google Docs—she had the lowest number of tapes recorded (N=2) among all study participants. Due to T3M2's reluctance to communicate through tapes and the time it took to understand her written comments, T3M1 and T3M3 had to frequently use the tapes to ask T3M2 to complete tasks for them. This is reflected by their most common tape category being "Coordinate Work" (85.71% out of 14 tapes). Overall, Team 3's case reveals that Winder can facilitate shared understanding in teams with largely unequal contribution levels. However, the effectiveness is dependent on whether the largest contributor records frequently or not.

3.7 Discussion

In this paper, we propose linked tapes, a novel form of asynchronous communication that integrates multimodal input and visual referencing. Our evaluation of Winder, a system which instantiates linked tapes, showed that tapes could lessen the perceived effort of producing comments and references to

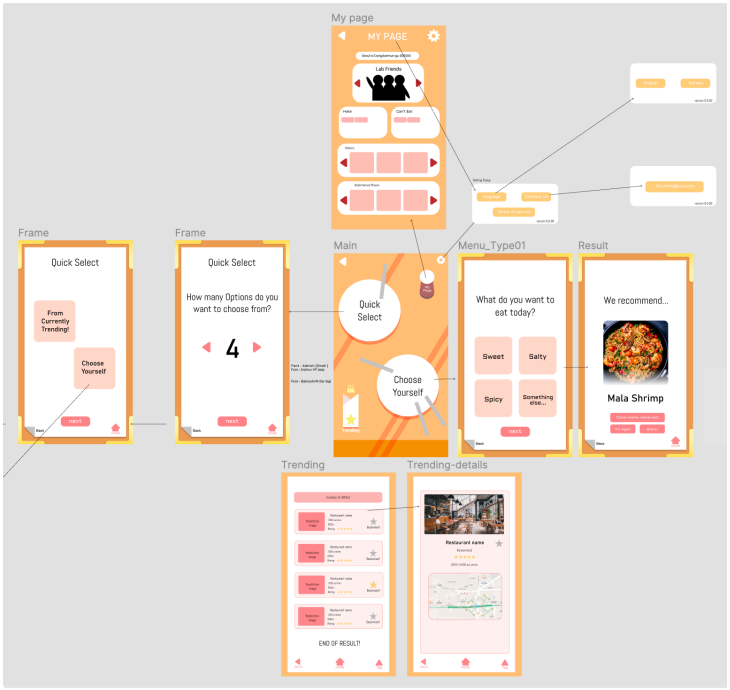

Figure 3.7: The final UI of the application designed by Team 3. They created a unique menu decision-making interface based on a roulette of restaurants, and screens to explore preferences and trends. They marked connections between the screens to show the user flow.

visual objects, while also facilitating navigation and understanding of others' comments. The study also showed that preempted tapes could potentially satisfy future communication needs. In this section, we provide a summary of what worked well and what did not with Winder. Then, we discuss how linked tape communication can be used in practice and generalized to other tasks. Finally, we discuss the implications of our work on asynchronous communication in general.

3.7.1 Winder Increases Shared Understanding and Augments Collaboration

Our evaluation focused on a holistic evaluation of Winder. As the features are interrelated, it is challenging to discern the particular effect of individual features. However, based on participants' comments regarding the features, we interpret the possible relationships between the features and how they could have led to the observed effects in our study.

Winder can potentially encourage students to communicate more frequently with their team members. This observed effect could be attributed to both the prompting and users perceiving less effort when producing messages with voice and clicks. While the greater transfer of information between team members can benefit shared understanding, it can also lead to greater burden. Despite participants generally considering production to be easier with Winder's modalities, the effort is not negligible, the use of voice can be awkward, and prompting was seen as a source of burden. Also, more messages produced entails greater burden on receivers. Thus, while Winder reduced feelings of burden related to typing, it could also introduce other types of burden and, at times, with no benefit for the teams as tapes could contain meaningless information—some participants recorded tapes simply because they were prompted.

Referencing visual objects through clicks appeared to reassure students that their messages would not be misunderstood. Our participants' comments regarding the perceived clarity of references—through the

object highlighting—infers that this reassurance was justified. Aside from supporting clarity in references, processing clicks and voice also allowed for inline thumbnails and object-based search, which supported students navigation through and within tapes. While participants expressed that this support could reduce their perceived effort in consumption, most of them still listened to all of their team members' tapes to gain a complete understanding. Several participants described this process as "tedious", showing that the bidirectional links had a limited effect on reducing consumption burden. To resolve this, future work could leverage bidirectional links and the document history to generate automatic summaries to allow for a general understanding of changes. These summaries could be similar to the *workflow histories* by Grossman et al. [78] but enriched with the information shared in the voice recordings.

By prompting users to record linked tapes as they are working, Winder obtains preemptively recorded tapes, which could, to a certain extent, substitute direct communication. Preempted tapes were used for diverse purposes—such as coordinating tasks and encouraging cooperation on the same UI screens. During the study, however, most of the tapes involved simple descriptions of completed work. As seen by the case study of Team 1, only providing these low-level comments can limit the benefits of preempted tapes. As tape recording resembles reflection—thinking and talking about one's own actions—providing guides based on reflection literature could result in high-level reflections that are more beneficial to teams' collaborative processes. For example, future work could adopt the levels of reflection proposed by Fleck and Fitzpatrick [79].

3.7.2 Integrating Linked Tapes into Teams' Application Ecosystems

While our study revealed that certain teams communicated exclusively through linked tapes, in practical scenarios, student teams may already rely on several familiar channels for communication (e.g., chat). Stacking linked tapes on top of these channels can be more detrimental than beneficial. As seen from our study, there is effort involved in keeping track of this additional channel and students may feel dissatisfied due to its unfamiliarity. Thus, for the success of Winder and linked tapes, it is crucial to establish how they fit in students' existing ecosystems. Course instructors could recommend ideal arrangements students could follow to use Winder alongside other channels. A partially successful arrangement which was observed in the study is to use a shared document to track high-level tasks, chat for semi-synchronous discussions on goals and direction, and linked tapes for more detailed and document-centric comments. In addition, Winder could be connected to communication channels already used by students to lower their barriers to entry [80]. For example, the system could send transcript snippets and object thumbnails from linked tapes to social chat applications already used by students.

3.7.3 Generalizability of Winder and Linked Tapes

While our investigation focused on UI design, we believe that Winder could be modified for other types of document-related tasks performed in online education settings. In particular, tasks in which referencing is common and students' frequently communicate asynchronously could be benefited by a Winder-like system. Two potential tasks are collaboration on presentation slides and discussion on lecture videos. For asynchronous collaboration on presentation slides, adapting Winder would be straightforward—presentations contain discrete objects (e.g., slides, text boxes, and shapes). However, the system might not be as beneficial in this task as presentations have an inherent structure—e.g., chronological order of slides—which could make referencing through text less challenging. For asynchronous discussions about lecture videos [81], computer vision techniques, such as video object segmen-

tation [82], could be used to extract discrete objects from the video. Then, students could click on these objects to discuss the lecture content.

3.7.4 Implications for Asynchronous Communication

Our work advances research in asynchronous communication through linked tapes, which bidirectionally integrate the communication channel and the document context. Previous work has primarily explored integration in a single direction—only referencing, which brings the context into the discussion, or only anchored communication, which incorporates the discussion into the context. As our study suggested, however, bidirectional integration allows for both discussions 'across' the document (higher-level and considering the whole picture) and 'into' objects (in-depth and specific to details). We suggest future researchers to consider bidirectional integration to support asynchronous collaboration. Additionally, our initial investigation into the impact of preemptive communication on asynchronous collaboration showed potential for future development. By prompting team members to communicate in advance to a system, it could allow a future user to have communication at hand without waiting for the availability of their team members. While our approach was naive—fixed intervals in which the user was softly notified—future work can explore this concept in more sophisticated ways. For example, prompting can be more contextual—based on intermediate steps detected in the user's workflow [83].

3.8 Limitations

Our work has limitations which we address in this section.

First, as we evaluated Winder in a controlled study instead of a deployment study, participants' communicative behaviors might have been affected by the setting. It is possible that they were pressured to record tapes as they were being observed by researchers. On the contrary, participants might have communicated less as the monetary award for the team with the best design might have not been enough incentive. Future work could explore the longitudinal use of Winder through a deployment study in a university project course.

Second, as evaluating the quality of designs is subjective and quality might be largely dependent on the characteristics of the members, we did not analyze how the use of tapes affected the quality of the outcomes. Therefore, we were unable to concretely measure how Winder impacted the quality of teams' collaborations.

Third, we conducted our study with participants who had no previous experience in UI design and were assigned into teams of three. The experiences of team members and the size of teams can significantly affect collaboration and communication behaviors. For example, professional designers may have established communication practices (e.g., routines and terminology) which can lead to different usage patterns of Winder and reactions to the system compared to those observed in our study. Future work could also investigate how Winder can be used by domain experts and bigger team sizes.

Finally, our study was carried out with a sample size of 24 participants, or 8 teams. More samples are needed to gain more conclusive findings. Also, our study focused on investigating participants' perceptions on burden when using Winder, as a whole, in close-to-real settings. For more conclusive findings on the effect of our system's features on burden, controlled studies quantitatively evaluating each feature against baselines are needed. For example, a possible setup could be measuring the time taken to transmit a certain amount of information with voice and clicks and comparing that to the time

taken through typing.

3.9 Conclusion

This paper presents *Winder*, a novel system that supports asynchronous communication between students in UI design collaboration. Winder provides communication through linked tapes—multimodal recordings of voice and clicks that contain bidirectional links between the comments and document objects. Additionally, by prompting the user to record tapes, Winder preemptively obtains information that can substitute direct communication when satisfying team members' needs, thus reducing the impact of communication delays. A five-day user study showed the effectiveness of linked tapes and preemptive recording in the collaborative processes of students. Finally, we discussed how Winder can be used in practice and generalized to other contexts, and the implications of linked tapes on general asynchronous communication.

Chapter 4. Stylette

4.1 Introduction

The web is inherently malleable. Websites are rendered out of documents—HTML, CSS, and JavaScript code—which are transmitted to the user's browser and, thus, can be readily accessed and modified on the user side. This malleability allows users to sculpt their experiences on the web by personalizing pages [84], self-repairing existing issues [85], or even enhancing pages with additional features [86, 87, 88]. The appeal of this malleability has led to the Greasemonkey [89] and Tampermonkey [90] plugins, which manage user scripts for these types of modifications, to collectively amass more than 10 million users. However, although such plugins allow users to install modifications designed by others, designing their own personal modifications may be out of reach for general end-users. To edit a web page's visual design or style, for example, users must be able to edit the underlying HTML and CSS files, but this requires an understanding of the code's language and structure. Thus, without the necessary expertise, most users are unable to mold websites into their own design.

To make the web more malleable for everyone, various end-user programming tools [85, 91, 92] have been designed to allow users with no expertise to directly manipulate a web page's visual design—abstracting away the underlying code. While these approaches allow users to focus on the visual representation, they require the user to manually perform several low-level operations (e.g., scrubbing on a color picker, typing in values) which can be tedious and effortful. Additionally, users must be able to decompose their high-level goals into the low-level operations supported by these tools—a task that inexperienced users frequently struggle with in other design-related tasks [45, 3]. Thus, to be able to easily transform a web page's design according to their goals, users require another level of abstraction.

Natural language interfaces allow users to perform complex, compound operations by simply saying or writing their intentions. The promise of this form of interaction has led to the development of various general-purpose voice assistants—e.g., Apple's Siri, Google Assistant, or Amazon's Alexa. In addition, task-specific natural language interfaces have also been designed to help inexperienced users perform complex tasks such as photo editing [45] or data visualization [93]. Similarly, if users could simply say what change they want to see, they could easily manipulate a web page without thinking about the underlying code or the low-level operations.

To investigate what language users would use when changing the style of a web page and how they would expect such changes to be presented, we conducted novice-expert sessions (N=8). In these sessions, novices used their voice to request changes on a web page's visual design and the expert, a developer, would then directly perform the changes using an in-browser developer tool. Our findings revealed that novices were frequently vague in their requests: omitting specific details (e.g., what color for the background), or using abstract terms that could not be clearly mapped to specific changes (e.g., "modern" or "vivid"). In addition to being vague due to inexperience, novices were also purposefully ambiguous as they wanted to explore the design space by seeing the expert's changes. Thus, novices expected the expert to make assumptions and provide a set of alternative changes that they could test and further iterate on.

Based on these findings, we designed Stylette, a natural language-based interface that assumes the user's intentions to provide a *palette* of web design properties and values. Stylette allows the user to

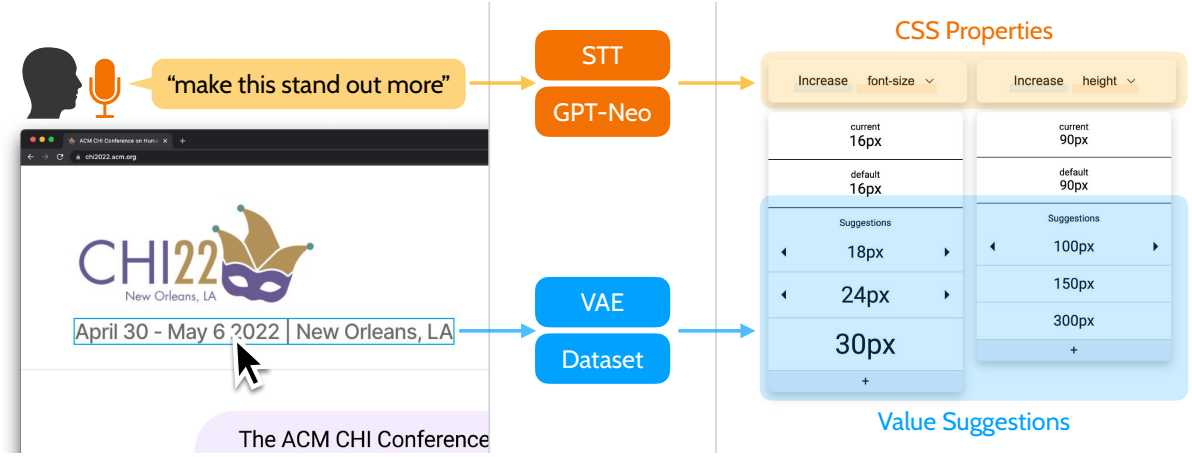

Figure 4.1: *Stylette* enables end-users to change the style of websites they visit by clicking on components and saying a desired change in natural language. A computational pipeline (1) transcribes the request and predicts plausible CSS properties with a large language model, and (2) encodes the clicked component using a convolutional neural network to identify and extract styling values from similar components in our large-scale dataset. These outputs are then presented in a *palette* that the user can use to iteratively change the component's style.

modify a web component by clicking on it, and then saying or typing their desired change (e.g., "increase the size" or "make this cleaner"). Based on the user's input, the system provides a palette that contains (1) a set of CSS properties that could be changed to satisfy the request, and, (2) for each property, a set of alternative values to explore and sample. The user can then simply change the component by applying the different property values found in the palette. To generate these palettes, we designed a computational pipeline that processes and combines the two input modalities, natural language and clicks. Specifically, a GPT-Neo-based architecture predicts suitable CSS properties from the natural language request, and a variational autoencoder (VAE) model encodes the clicked-on component to extract the values of similar components from our dataset of 1.7 million components.

To evaluate Stylette, we conducted a between-subjects study (N=40) in which participants performed a design recreation task and an open-ended design task with either our system or DevTools, the Chrome Browser's developer tool. Our study revealed that Stylette helped participants perform styling changes 35% faster and with a higher success rate—80% of Stylette participants successfully recreated a design within the allowed time while only 35% succeeded with DevTools. Additionally, our system led participants to experiment with and familiarize themselves with a more diverse set of properties. As participants acquired more knowledge with Stylette, however, natural language interaction limited their productivity as they could not apply this knowledge to directly make changes themselves. These insights suggest a need for a hybrid approach: natural language interaction to initially support quick familiarization with a tool, and then gradually phasing in more direct interaction methods.

This paper presents the following contributions:

- Stylette: A novel system that allows users to change the design of websites by using natural language to express their goal, and then iterating with the set of alternatives presented by the system.
- A computational pipeline that combines NLP and CV techniques to process a natural language request and a web component into a set of plausible CSS property and value changes.

- Findings from a between-subjects study that reveals how natural language support can help novices familiarize with and perform a previously unknown design/coding task.

4.2 Formative Study

We conducted a formative study to investigate how novices would change the design of websites and how they would naturally request such changes. In this study, participants freely browsed through a website and requested styling changes by speaking aloud. One of the researchers, with several years of development experience, acted as an expert and made these changes on-the-go.

4.2.1 Participants

We invited 8 participants (5 female, 3 male), all of whom had no background in web development. Each participant sat alongside the expert or, if participating remotely, shared their screen through a video conferencing tool[1]. To reduce the time participants spent familiarizing themselves with a website and to prompt more realistic requests, participants chose a website they frequently visit for the study. Most participants chose either our university's web portal or its learning management system.

4.2.2 Study Procedure

During the study, participants examined the website and requested styling changes from the expert. On their own computer, the expert used the Chrome Browser's DevTools[2] to directly edit the CSS code. The expert would then share the edits and, if participants were not satisfied, they could ask for further edits. After around 30 minutes of editing, the participants were then asked a couple of questions about their experience. Sessions lasted a maximum of 40 minutes and participants made 8.38 requests on average.

4.2.3 Requests were Vague and Abstract

Despite being able to concretely specify which web component they wanted to edit, participants struggled to concretely explain how it should be changed. Participants generally relied on vague phrases (e.g., "more readable" or "emphasize this") or abstract terms (e.g., "modern", "vivid" or "dull") that did not immediately reveal what visual aspect of the component should be changed or how. Even if they were specific about which aspect to change, participants would also tend to be vague about the value to set for that aspect. For instance, a participant said "more transparent" without specifying how much more transparent.

We observed that the behavior of our participants was beyond not knowing the names of CSS properties—like the vocabulary problem observed in other tasks [31]. Participants also struggled to specify the visual aspects of the web components even without using the actual property names. For example, a participant requested a text component to be highlighted but, when asked if the text should be bolder or colored differently, they were unable to provide a definite answer. Participants explained that their hesitation was either because (1) they were unsure about which aspect to change, or (2) they could decide on an aspect but were not confident that it would "look good".

[1] https://zoom.us

[2] https://developer.chrome.com/docs/devtools/

4.2.4 Assumptions Over Questions

To concretize the participants' vague requests, the expert asked questions to prompt further details. For example, when a participant asked to make a component "less tacky", the expert asked about what aspect made it appear "tacky". While participants recognized how these questions helped them decompose and iteratively reach their goals, they found this back-and-forth to be tedious. As participants were unsure about the details, they did not want to dedicate the mental effort to ponder about the details and, instead, expected the expert to assume the details for them. They mentioned that it would be easier to distinguish what they liked or disliked if the expert made these assumptions and presented a visual result. Additionally, instead of one outcome for each request, participants wanted various options for the same request in order to explore the design space.

4.2.5 Natural Language is Not a Panacea

For most participants, the use of voice or natural language was a major positive aspect about interacting with the expert. Participants mentioned how it was "comfortable" to use natural language to simply explain what they wanted to change. However, while they felt that natural language helped to get the editing process started, participants desired more direct control when iterating on edits. Specifically, when deciding on a value for a property, they felt frustrated about having to test different values by turn-taking with the expert. Instead, participants wanted to be presented with widgets that allowed them to test alternative values by themselves.

Based on the insights from our study, we derive the following design goals:

- DG1: Interpret users' vague requests to present plausible changes.
- DG2: Provide multiple alternative properties and values that could satisfy one request.
- DG3: Allow users to directly iterate on the details for a change.

4.3 Stylette

Based on our design goals, we present Stylette (Fig. 4.2), a system that enables end-users to change the visual design of any website by simply clicking on a component and saying what change they want to see. The system interprets the user's request through an NLP pipeline trained on vague language (DG1) to present a *palette* that consists of multiple CSS properties (DG2) that could be changed to satisfy the request. To iteratively edit each property, the user can directly adjust values and experiment with various suggestions extracted from a large-scale dataset (DG2, DG3). Stylette is implemented as a Chrome Extension and, using a method similar to Tanner et al.'s [91], it saves the user's changes in the browser's memory so that they persist when the user returns to the page.

4.3.1 User Scenario

To illustrate how Stylette can be used, let's follow Sofia, a sociologist with no web development experience, as she browses through the CHI 2022 website while preparing for her paper submission.

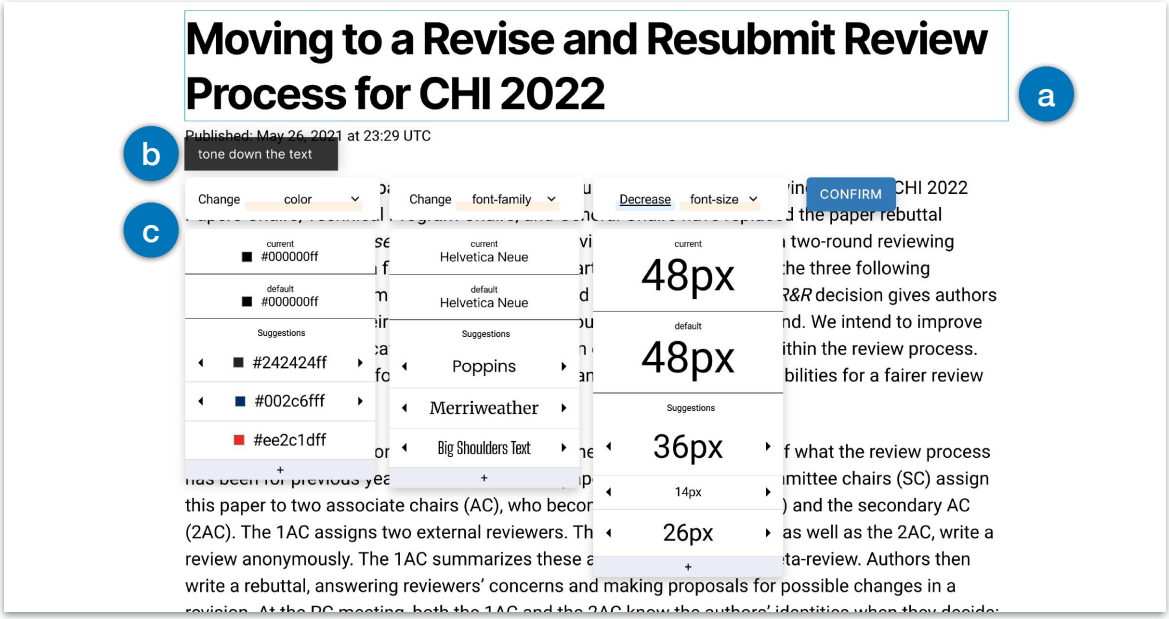

Figure 4.2: *Stylette* is shown overlaid on a website. When activated, the system shows a blue border (a) over components the user has hovered-on or clicked. After the user selects a component and records a request, Stylette transcribes the request (b) and displays a *palette* that contains CSS properties and values.

Selecting a Component

As soon as she enters the CHI website, Sofia is slightly taken aback by the size of the text on the main page. To started editing, she clicks on the Stylette icon on her extension toolbar. Now, she can select components to edit so she clicks on the first header in the page (Fig. 4.2a).

Making a Verbal Request

With the component selected, Stylette overlays a transcript box on the website, prompting Sofia to say her request. To do so, she holds down the **Ctrl** key and says: *"tone down the text"* (Fig. 4.2b). After releasing the **Ctrl** key and a short processing period, the transcript box now displays a transcript of what Sofia said. In case the transcription is wrong, Sofia can correct it by typing directly on the box and pressing **Enter** to process the corrected transcript. In addition, a *palette* is now presented, showing three different CSS properties that Sofia can edit to satisfy her needs (Fig. 4.2c): she can make the text smaller with *font-size*, change it to a slimmer *font-family*, or apply a lighter *color*.

Iterating with the *palette*

Under each property, the *palette* presents a list of values: the current value for the property (Fig. 4.3a), the default or original value (Fig. 4.3b), and a set of value suggestions (Fig. 4.3c). For properties with numerical values, like *font-size*, the system also interprets whether the user wants to increase or decrease the current value (Fig. 4.3d) and provides suggestions accordingly. As Sofia hovers over the suggested values for *font-size*, Sofia can see how the header would look with that *font-size*. After finding one she feels satisfied with, she clicks on it to apply that change. If Sofia actually wanted to

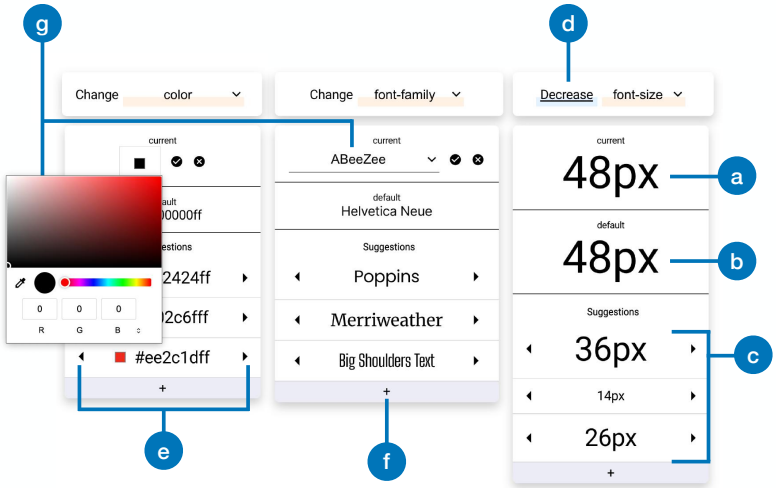

Figure 4.3: For each property, the *palette* presents the current value (a), the default or original value before any changes (b), and a list of suggested values (c). For numerical values, the palette presents suggested values that are either larger or smaller than the current value based on the system's prediction (d). To see other similar suggestions, the user can click on the arrows next to a suggested value (e). To see different suggestions, the user can click on the "+" button (f). The user can also click on the current value to reveal widgets to manually set values (g): input box for numerical properties (e.g., *font-size*), drop-down menu for nominal properties (e.g., *font-family*), or color picker for colors.

increase the *font-size* and the system gave an incorrect prediction, she could click on "Decrease" next to the property name to switch it to "Increase" and the suggestions would change accordingly. If she wanted to change another property similar to *font-size*, she could also click on the property name to see a drop-down of other properties with similar names (e.g., *font-style*, *font-weight*).

After setting the *font-size*, Sofia also notices the *color* property. As she feels that this could also be toned down a bit, she clicks on the lighter black color ("#242424ff") in the suggestions. After seeing this change, she feels that the header's color should be even lighter, so she clicks on the arrows next to that suggestion (Fig. 4.3e) to see other similar suggestions. Going through the carousel, she finds a color that she likes so she clicks on it. If she is unsatisfied with the suggestions, she can see other different suggestions by clicking on the "+" at the bottom (Fig. 4.3f), or manually set her own value by clicking on the current value to expose manual change widgets (Fig. 4.3g).

4.3.2 Pipeline

To support the interaction presented in the scenario, we present a computational pipeline that processes the two input modalities, voice and click, to generate the *palettees* (Fig. 4.4). For voice, the audio is recorded and automatically transcribed. For clicks, a screenshot of the component selected by the user is automatically captured. These inputs are then processed separately by the computational pipeline.

Processing Natural Language

Our pipeline's NLP module takes the transcribed request, and predicts relevant CSS properties and the direction of the change (e.g., increase, decrease, or neither). For this purpose, we employ the

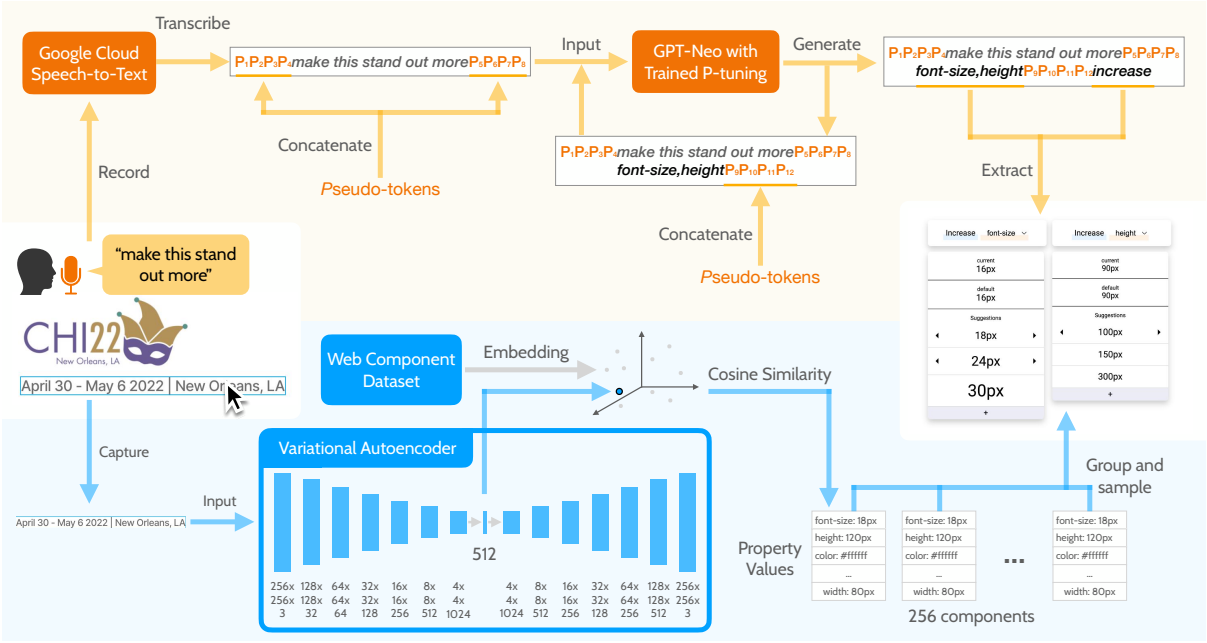

Figure 4.4: Our computational pipeline integrates a natural language processing (NLP) module (top, orange) and a computer vision (CV) module (bottom, blue). The diagram illustrates the pipeline at inference time—processing user's input of natural language and clicks to generate a set of CSS property alternatives and value suggestions.

2.7 billion parameter version of the GPT-Neo model [94], an open-source implementation of OpenAI's GPT-3 model [58]. With a well-crafted prompt and a small number of examples, these models have been shown to achieve high performance on previously unseen tasks. However, hand-crafting prompts can be a time-consuming and very imprecise process—small alterations can lead to significant differences in performance.

Architecture: Instead, we implement the P-tuning technique [95] that automatically searches for a prompt with high performance. In this technique, prompts are composed by concatenating *pseudo-tokens* to the natural language input and training the embeddings for these pseudo-tokens. In our pipeline, we use 12 pseudo-tokens. For training, we template the prompt as $[P_{1:4}$,R,$P_{5:8}$,C,$P_{9:12}$,D$]$, where $p_{1:12}$ are the pseudo-tokens, R is the natural language request, C is the CSS properties separated by commas, and D is the change direction (i.e., "increase", "decrease", or "none"). During inference, we template the prompt as shown in Figure 4.4 ("Concatenate"): $[P_{1:4}$,R,$P_{5:8}]$. This templated prompt is passed as input to the model and the model's output is controlled to generate at least three CSS properties—to provide multiple alternatives to users ("Input" and "Generate" in Fig. 4.4). Then, the generated CSS properties and the remaining pseudo-tokens are concatenated to the initial prompt, and this result is passed to the model again to generate the change direction.

Dataset: Another merit of the P-tuning technique is that it only requires a small amount of data for training. To train our pseudo-tokens, we created a small-scale dataset consisting of 300 triplets of (1) vague natural language requests, (2) CSS property sets, and (3) change directions. As a first step in creating this dataset, we requested 29 web developers to each write three hypothetical vague requests that a user could ask when wanting to change a website's design. Then, each developer looked at the requests written by another person and wrote CSS properties to change and the direction for the change that could satisfy each request. We removed requests that were too specific (e.g., included

	CSS Property Prediction				Change Direction
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
P-tuning	0.557	0.670	0.761	0.653	0.819
Hand-crafted	0.509	0.623	0.648	0.585	0.413

Table 4.1: With trained P-tuning, the GPT-Neo model achieved higher performance when predicting CSS properties and change directions, when compared to using a hand-crafted prompt as input.

property names), and added requests from our formative study and system's pilot studies. The CSS properties in this initial data are the ones supported in our system (Table 4.2). We then expanded the dataset by automatically augmenting the initial data with synonym/antonym replacement [96, 97], and/or backtranslation [98]—one of the authors checked and corrected the augmentations. Finally, as performance can deteriorate significantly due to class imbalance [99], we ensured that each CSS property appeared in at least 10% of the requests—the representation of CSS properties in the dataset is shown in Table 4.2.

Training: In the training process, we used 200 triplets for training and validation (80%-20% split), and reserved 100 for testing. The pseudo-tokens were trained on the generative loss from the GPT-Neo model with the Adam optimizer, until early stopping on the validation loss. We used an initial learning rate of 0.0001, batch size of 8, weight decay value of 3e-7, and gradient clipping value of 5. When compared to the model with our best hand-crafted prompt, GPT-Neo with trained P-tuning achieved a higher F1-score when predicting CSS properties and higher accuracy when predicting change direction (Table 4.1). Additionally, the recall with P-tuning exceeds 75% which suggests that, for the average request, the model will likely return most of the properties that the user might need.

Processing Web Components

As our formative study revealed, users can struggle when deciding on a value for a change (e.g., what color for the background) and may benefit from seeing various alternatives. The components in other websites can be a rich source for these alternatives. However, as the style of a component depends on what that component represents (e.g., the *font-size* for a header vs that for a paragraph), selecting random components would not lead to sensible and useful alternatives. Thus, it would be more beneficial to identify components in other websites that are similar to the one the user wants to change. Similarity could be measured by calculating property differences and aggregating these into one measure, but, as properties differ in the scale and type of values, this requires the difference and aggregation calculations to be carefully formulated.

Architecture: As an alternative, we leverage a variational autoencoder (VAE) model [100] ("Variational Autoencoder" in Fig. 4.4) to automatically learn a concise representation of the visual features of components. In our pipeline, we use this VAE model to encode the screenshot image of a component into a 512-dimensional vector. Through cosine similarity, this vector is then compared to the vector representations of all the components in our large-scale dataset to identify 256 similar components and retrieve their property values ("Cosine Similarity" in Fig. 4.4). To provide coarse diversity but also more fine-grained alternatives, the *palette* presents suggested values in two levels: (1) different values as separate rows, and (2) similar values as a carousel in the same row. To support this, the pipeline groups the retrieved values according to specific rules ("Grouping Method" in Table 4.2) and each group

CSS Property	Grouping Method	In Dataset
height	Interval binning (N=20)	11.7%
width	Interval binning (N=20)	11.7%
margin	Interval binning (N=10)	11.3%
padding	Interval binning (N=10)	12.9%
color	K-means clustering (N=6)	12.9%
background-color	K-means clustering (N=6)	12.5%
opacity	Interval binning (N=2)	10.4%
font-size	Interval binning (N=10)	13.3%
font-family	Google Fonts categories (N=5)	13.8%
font-style	Nominal value	11.3%
font-weight	Interval binning (N=10)	11.7%
text-align	Nominal value	11.3%
text-decoration	Nominal value	11.3%
border-width	Interval binning (N=10)	11.3%
border-color	K-means clustering (N=6)	11.3%
border-radius	Interval binning (N=10)	11.3%

Table 4.2: The CSS properties supported by Stylette. The table presents the representation of each property in the natural language request dataset. Each row also shows how values for a property are grouped when suggested to the user: interval binning into N equally-spaced intervals, K-means clustering with the elbow method, based on the categories from Google Fonts[3], and no grouping for properties with nominal values.

represents a suggestion row. Then, a maximum of 10 values are randomly sampled for each group and these alternatives are presented through the carousel. For color-related properties and the *font-family* property, users in the pilot studies wanted more diverse values so, for these properties, we populate other suggestion groups by retrieving the values from random components in the dataset.

Dataset: Although there are datasets for mobile UI components [101] or for whole web pages [102], there are none for individual web components. Thus, to train the VAE model, we constructed our own dataset. We first compiled a list of websites from various sources: the S&P500, the Webby Awards [103], and the Open PageRank dataset [104]. We removed any websites that (1) could not be accessed, (2) had very similar URLs, or (3) had less than 16 components. This led to a final list of 7,565 websites. For each website's main page, we used a crawler to capture each component's CSS properties and screenshot image. After removing components that were less than 10 pixels wide or tall, the final dataset consisted of 1,761,161 components.

Training: The VAE model is composed of six convolutional layers for encoding, one linear layer as a bottleneck, and six transposed convolutional layers for decoding. The dimensions of the outputs at each layer are shown in Figure 4.4 ("Variational Autoencoder"). During training, the image of a component is encoded into a vector using the encoding and bottleneck layers, and then this vector is passed through the decoding layers to recreate the image. The model is trained to maximize the evidence lower bound (ELBO) value between the original image and the recreated image. We trained our model for 3 epochs with an Adam optimizer, using a learning rate of 0.0001 and batch size of 256.

4.3.3 Implementation

We implemented the interface of Stylette as a Chrome Extension, using JavaScript, HTML, and CSS. For the backend, we used a Node.js server to pre-process requests from the interface and transcribe the audio with the Google Cloud Speech-to-Text API[4]. To serve the computational pipeline, we used a Flask server running with DeepSpeed[5].

4.4 Evaluation

We conducted a between-subjects study where we compared Stylette against the Chrome Browser's DevTools, a tool widely available for general end-users to edit websites with. The study was composed of (1) a well-defined task of redesigning a website to look like a given outcome, and (2) an open-ended task of styling a website to follow the design direction of provided references. We designed these two tasks to investigate how Stylette helped participants style components when (1) they have a clear idea about how it should change, or (2) they only have a vague sense of direction. Specifically, we pose the following research questions:

- RQ1. How does Stylette help novice users find the CSS properties required to perform desired styling changes?
- RQ2. Can Stylette encourage novices to perform a greater number of changes and use more diverse CSS properties?
- RQ3. How does novices' usage of Stylette affect their self-confidence regarding their own web designing abilities?

4.4.1 Participants and Apparatus

We recruited 40 participants (11 female, 29 male; age M=21.5 and SD=3.05) who all reported having no previous experience with web design or coding (no knowledge of HTML and CSS). We also verified that participants were relatively fluent in spoken English to reduce frustration due to the performance of speech-to-text technologies. Participants were divided into two equally-sized groups and each group was assigned to use either Stylette or Chrome DevTools. As six participants mentioned having other prior design experiences and this could affect performance (e.g., the term "padding" is used in other design tasks), they were also equally split into each condition. To simulate a realistic setting, participants who used Chrome DevTools were also allowed to freely use search engines to find resources and information. The study lasted a maximum of 90 minutes and participants were compensated with 30,000 KRW (approximately 26 USD).

4.4.2 Study Procedure

The study took place face-to-face, strictly following the COVID-19 guidelines: participants had to wear masks and plastic gloves, and their temperature was checked before sessions. Each participant was provided with a computer with a Chrome browser installed and their assigned tool, Stylette or DevTools,

[4]https://cloud.google.com/speech-to-text
[5]https://www.deepspeed.ai/
[5]https://fonts.google.com/

Component Tag	Properties to Change	Success Range
h2	font-size (FSz)	80px - 120px
p	font-weight (FW)	700 - 900
span	background-color (BgC)	(0, 0, 150) - (50, 50, 255)
	color (C)	(255, 255, 255) - (200, 200, 200)
video	border-radius (BR)	60px - 100px
button	border-width (BW)	6px - 10px
	border-color (BC)	(200, 100, 0) - (255, 200, 50)
div	text-align (TA)	"center"
h2	font-style (FSt)	"italic" or "oblique"
button	padding (P)	30px - 60px
img	width (W)	600px - 800px
h3	font-family (F)	Any family in the "cursive" category
h3	text-decoration (TD)	"underline"
div	margin (M)	80px - 120px
img	height (H)	350px - 450px
img	opacity (O)	0.3 - 0.7

Table 4.3: List of the components that participants had to change during Task 1, in the order that they had to be changed. For each component, the table shows its tag type and the properties that had to be changed. For the properties, the list also shows their abbreviated names (which are used hereafter), and the range of values that were accepted as successful changes (color values shown as RGB triplets).

already opened. After reading and signing the informed consent form, participants were first provided with a brief walkthrough of their assigned tool and were then allowed to test the tool for a total of 5 minutes. After this, participants completed a short pre-task survey.

After the survey, participants started Task 1. Participants were tasked with using their assigned tool to redesign our institute's "About" web page[6] to look as close as possible to a provided final design. This final design was provided as a before-after image with circling around components to change and labels showing how many properties to change for each component. Natural language explanations of the changes were not provided to prevent biasing the language used by Stylette participants. The task involved changing 14 different components and all of the 16 CSS properties supported by our system (Table 4.3). Participants were asked to change the components in the order that they appeared in the website, but were allowed to skip challenging components and come back to them later. A researcher verified that a component had been successfully changed once the values of the correct properties were within the accepted success range ("Success Range" in Table 4.3). Participants had 30 minutes to successfully change all the components. After the task, participants completed a short survey.

After Task 1, participants started Task 2 after a 5-minute break. To ensure that all participants started Task 2 with the same amount of knowledge, those that did not complete Task 1 were first shown how to perform the changes that they did not complete. The aim of Task 2 was to investigate how participants used their assigned tool when only provided with a vague direction for changes and allowed greater flexibility. Participants were tasked with changing a given website such that it followed the

[6] Anonymized

design direction of four reference websites (Fig. 4.5). These references were chosen as they shared a similar modern aesthetic, but also differed in how their content was structured. Participants were given 25 minutes for this task. After this task, participants completed a short survey. Finally, a short interview was conducted asking participants about their experiences during both tasks.

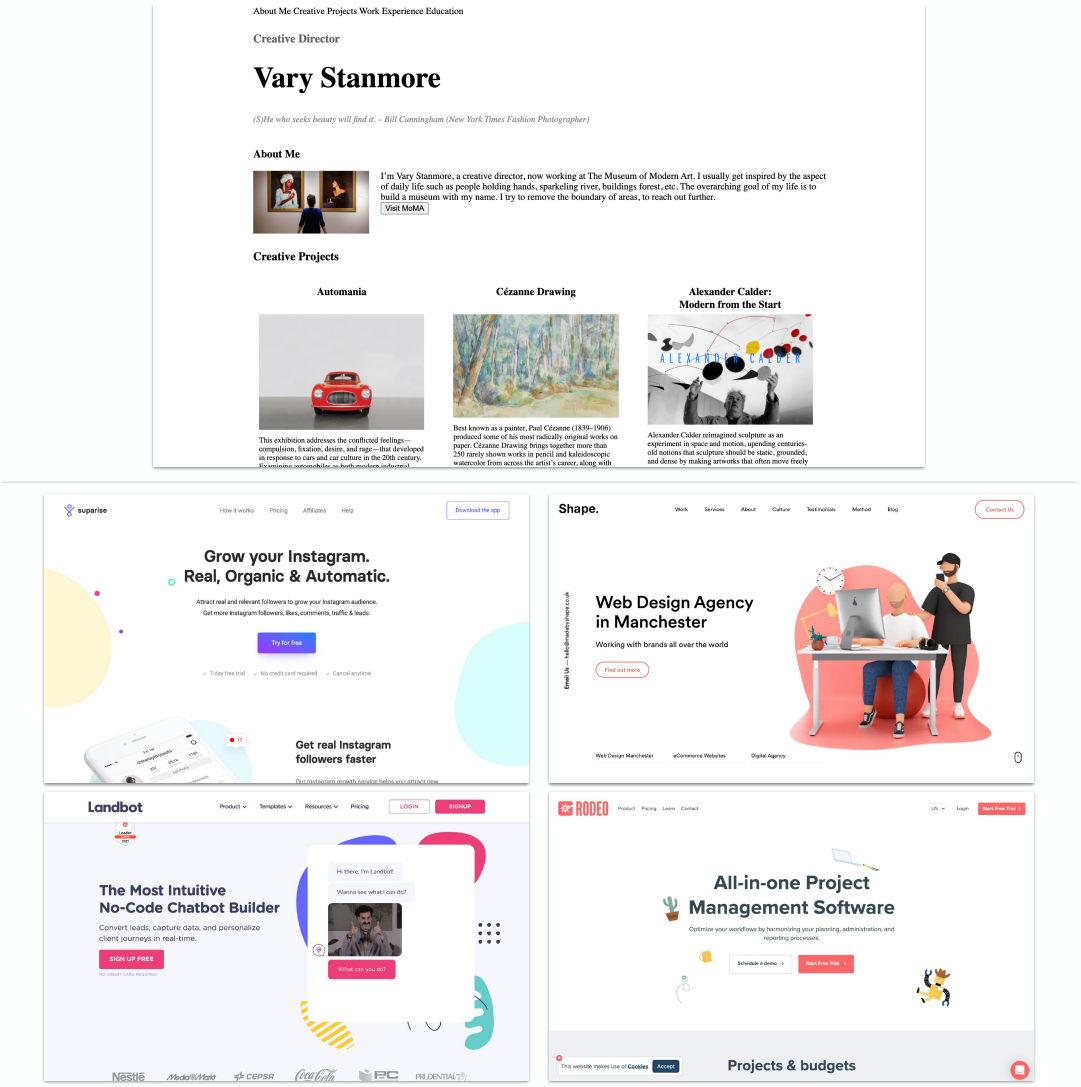

Figure 4.5: (Top) The website that participants styled in Task 2 mimics the portfolio of a creative director for a museum. The website only has basic styling to encourage participants to be creative and make many changes. (Bottom) The four reference websites that were provided during the task: Suparise (https://suparise.com), MadeByShape (https://madebyshape.co.uk), Landbot (https://landbot.io), and Rodeo (https://getrodeo.io).

4.4.3 Measures

We collected responses to the pre-survey and the post-surveys after each task. All of the surveys contained four questions asking participants to rate, on a 7-point Likert scale, their self-confidence with respect to their ability to (1) perform a website design changing task, (2) plan design changes, (3) iterate on changes, and (4) use the given tool. We averaged the responses to these questions to derive

one score for self-confidence. The two post-surveys included the six questions from the NASA-TLX questionnaire [105] to measure participants' perceived workload.

We also quantitatively measured task-related metrics. For Task 1, we measured the time taken to successfully change each component and to complete the whole task. We hypothesized that Stylette would help participants find properties faster, and therefore complete changes in less time. Although all participants had no previous web design experiences and those with other design experiences were equally divided into each condition, individual design interest and skill could still affect the quality of the final designs in Task 2. Due to this reason, we did not rate these designs and, instead, we measured how many property changes were made in total. We hypothesized that Stylette participants would make more changes as they could explore diverse properties and values. A value close to 0 indicates equal usage, spread across various properties, and one close to 1 indicates unequal usage, few properties were used excessively.

For qualitative data, we analyzed participants' responses during the short interviews to understand their perceptions of the given tool and how they leveraged it for their purposes. We also iteratively coded the requests used by Stylette participants in Task 2 to classify them according to the vagueness of their content and language.

4.5 Results

Our results demonstrated that Stylette helped participants perform styling changes faster and with greater success in Task 1, but it did not enhance productivity in Task 2. For the statistic analysis of each measure, we first conducted a Shapiro-Wilk test to determine if the data was parametric. When comparing between conditions, we used an independent t-test (if parametric) and a Mann-Whitney U test (if non-parametric). When comparing between tasks within the same condition, we used a paired t-test (if parametric) and a Wilcoxon signed-rank test (if non-parametric).

4.5.1 Task 1: Well-Defined Task

For Task 1, we provide an analysis of participants' performance and perceived workload.

Performance

Overall, participants using Stylette significantly outperformed DevTools participants in this task (Fig. 4.6). While only 7 out of 20 DevTools participants completed all changes, 16 out of 20 Stylette participants completed the task. Additionally, when comparing only those who completed the task, Stylette participants (M=971.8s, SD=314.4s) completed the task in 35% less time than those that used DevTools (M=1493.0s, SD=295.9s, t=-3.72, p=0.001). Comparing the time taken to successfully change each component revealed that DevTools participants struggled significantly with specific properties (e.g., *border-radius* (BR) and *padding* (P) in Fig. 4.6). These struggles generally involved two scenarios: (1) vague search queries led to unhelpful results, or (2) the name of a CSS property did not immediately reveal its visual function.

To illustrate the first scenario, several participants tried queries like *"enlarge the border in CSS"* when searching for the *padding* property, but this only returned results for *border-width*—the search engine took them "too literally" (D2, D7, D9, D14). In other cases, participants' vague queries returned search results for more advanced changes beyond their needs. For example, to adjust the *height* or *width*,

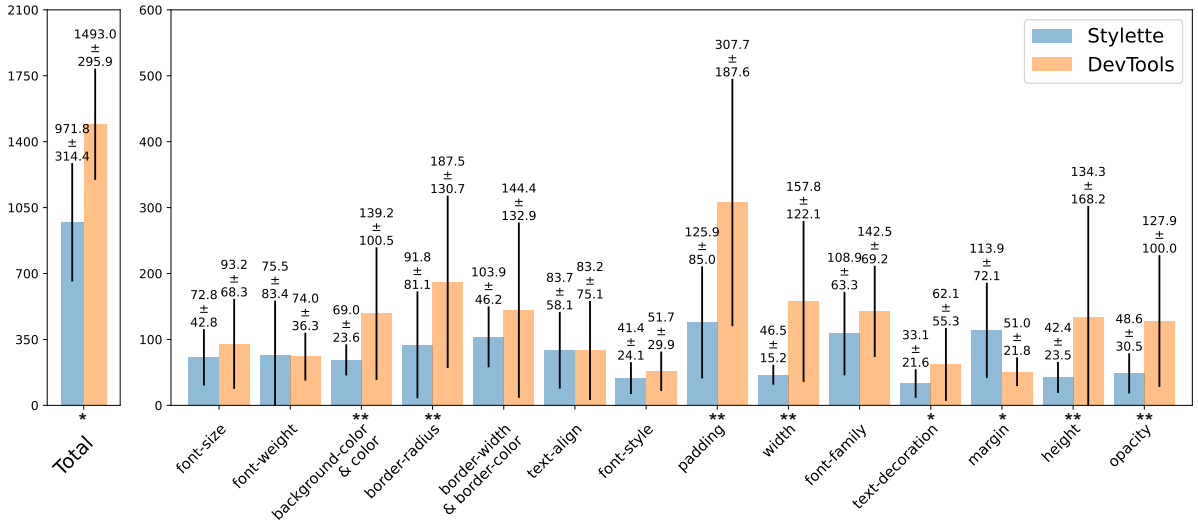

Figure 4.6: The average time taken for participants to successfully change each component using Stylette or DevTools. Each component is represented with the abbreviated names of the properties changed (Table 4.3). For each property, the figure shows if the difference in time taken for each condition was statistically significant (*: p <.05, **: p <.01).

participants searched "resize image in CSS" but this returned results about "responsive images". In contrast, as our system was trained on vague requests and presents multiple properties for one request, Stylette participants had more success using similarly vague language—requesting *"enlarge the border"* to the system returned *padding* among the options.

The second scenario involved properties with names that could be unclear for novices, such as *border-radius* or *text-decoration*. In these situations, the properties were frequently found in the search results, but DevTools participants would overlook them as they could not immediately visualize the functions from the names. Stylette participants also overlooked properties when they mismatched with their mental models. However, hovering over the suggested values allowed them to quickly use and test the functions of properties. S5 mentioned: *"By applying [the recommendations], I could understand what [visual] concept the [margin and padding] were related to"*.

Perceived Workload

In Task 1, responses to the NASA-TLX questions revealed that the effect of Stylette on perceived workload was mixed (Table 4.4). Stylette participants reported experiencing significantly less temporal

Task 1	Mental	Physical	Temporal	Effort	Performance	Frustration
Wonder	3.90 (1.41)	2.55 (1.57)	3.45 (1.73)	3.15 (1.79)	5.45 (1.10)	3.00 (1.52)
DevTools	4.35 (1.42)	1.65 (0.93)	4.50 (0.95)	4.00 (1.45)	4.85 (1.53)	2.25 (1.21)
p	0.14	**0.02**	**0.01**	**0.03**	0.11	**0.05**

Table 4.4: For Task 1, participants' average ratings on the perceived workload questions (NASA-TLX) showed that temporal demand and effort were significantly lower with Stylette, but physical demand and frustration were significantly higher.

demand (U=118.0, p=0.0118) and effort (U=133.0, p=0.0336) than those that used DevTools. DevTools participants felt significant time pressure due to the lengthy and effortful process of thinking about what to search, skimming through search results, and reading resources. In comparison, Stylette participants could simply say something and look through the three to five properties presented by the system.

However, Stylette participants also experienced significantly higher frustration when compared to DevTools participants (U=139.5, p=0.0472). According to participants, this frustration was partially attributed to the fact that the coupled AI algorithms (i.e., speech-to-text and property prediction) could both fail. For example, as they did not notice the transcription errors, several participants were confused when concrete requests (e.g., "underline text") did not return the correct properties. Other participants were overly preoccupied with the transcription and immediately corrected any errors—failing to notice that the system had already returned desired properties. When fixing errors, participants also had to alternate between modalities (i.e., voice, text, and clicks) which could explain why Stylette participants reported feeling a higher physical demand (U=127.0, p=0.0187).

4.5.2 Task 2: Open-Ended Task

For Task 2, we measured productivity (i.e., how many changes were made and whether varied properties were used), and perceived workload. Samples of the participants' final designs (Fig. 4.7) show their creativity and how they each focused on different aspects of the website.

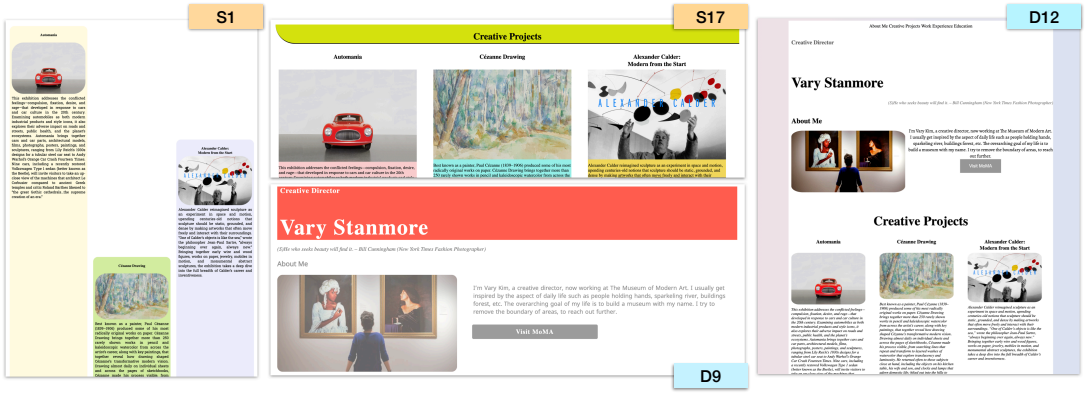

Figure 4.7: Sample of designs created by Task 2 participants. S1 used *padding* to spread the middle content vertically such that each item would appear gradually as the user scrolls down the page. S17 serendipitously found the *border-width* property and used it to add a "shadow" to the container for the "Creative Projects" subheader. D9 used *opacity* in several components to lighten the web page's content. D12 increased the *border-width* and added *border-color* to add colored bars on the sides of the page.

Productivity

While participants in the Stylette condition (M=42.85, SD=12.18) made more property changes than those in the DevTools condition (M=39.30, SD=12.71), this difference was not statistically significant (t=0.901, p=0.3729). The lack of a statistical difference could be attributed to the benefits and drawbacks of each tool's interaction method. DevTools participants spent more time searching for information, but, once they had the required knowledge, they could directly make changes. Stylette participants could use natural language to easily find properties, but, even if they already knew which property to change, they expended time waiting for the system to process requests and fixing any AI-related errors. Several

Task 2	Mental	Physical	Temporal	Effort	Performance	Frustration
Wonder	4.75 (1.29)	2.90 (1.71)	4.35 (1.66)	4.25 (1.48)	4.05 (1.43)	3.55 (1.47)
DevTools	4.90 (1.21)	2.00 (1.45)	4.55 (1.50)	4.55 (0.83)	4.45 (1.39)	2.65 (1.50)
p	0.34	**0.03**	0.69	0.23	0.24	**0.03**

Table 4.5: For Task 2, participants' average ratings on the perceived workload questions (NASA-TLX) showed that physical demand and frustration were still significantly higher with Stylette, and that temporal demand and effort no longer differed significantly.

participants (S5, S6, S7, S15) noted that, after learning the properties in Task 1, they wanted to directly change the properties in Task 2—without using natural language.

Additionally, as Stylette presents other options in the *palette*, participants appeared to spend additional time browsing through them. While this exploration could increase effort, it also appeared to encourage familiarization with a wider range of properties. The Gini index for property usage shows that Stylette participants tried various properties (M=0.292, SD=0.045) while DevTools participants mostly stuck with a few properties that they were accustomed to (M=0.325, SD=0.052, t=-2.169, p=0.0364). Beyond encouraging experimentation with more properties, in some cases, the system also led participants to serendipitously find alternative uses for known properties. S17 mentioned, *"Accidentally I just found [border-width] while trying to change the radius [so I changed it] and it shows a shadow effect that looks really, really good."* (design shown in Fig. 4.7).

Perceived Workload

Similar to the results of Task 1, participants in the Stylette condition reported experiencing higher physical demand (U=131.5, p=0.0278) and frustration (U=129.0, p=0.0258) than those in the DevTools condition (Table 4.5). Unlike Task 1, however, Stylette participants no longer reported feeling significantly less temporal demand or effort. It is plausible that, due to the open-ended nature of Task 2, Stylette participants now spent more time and effort exploring the design space through the alternatives presented by the system—Gini index results support this explanation.

Usage Patterns of Stylette

As Task 2 allowed for more flexible and natural use, we also analyzed participants' usage of Stylette during this task. Participants issued an average of 36.8 requests (max=58, min=18, SD=11.2) and the requests had an average length of 3.2 words (max=12, min=1, SD=1.2).

Our categorization of these requests showed that, unlike our formative study results, the requests were frequently specific and became more specific and less vague towards the end of the task (Table 4.6). Participants' interviews revealed that this gradual specificity was due to various reasons. For one, the tool helped participants learn property names so they could now use them in requests (S5, S8, S13, S19). Others observed that the system was more accurate if they were more specific, so they adjusted their requests accordingly (S3, S4, S16, S20). A sample of participants' requests (Table 4.7) shows that the system was indeed more likely to predict users' expected properties if the requests included more specific information.

Like our formative study, however, around half of the requests were vague ("PP", "PV" and "A"

Request Type	Description	Examples	Percentage	Q1	Q4
Property Specific (PS)	Specific property name expressed.	*"change background color"* *"align text in the center"*	48.1% (352)	46.6%	56.0%
Property Partial (PP)	Property name partially expressed.	*"add border"* *"change the font"*	35.3% (258)	35.2%	34.7%
Property Vague (PV)	Property name not clearly apparent.	*"make this bigger"* *"increase the spacing"*	11.5% (84)	11.9%	4.7%
Property Total	-	-	94.9% (694)	93.8%	95.3%
Value Specific (VS)	Specific value expressed.	*"change to dark grey color"* *"increase font size to 14 px"*	11.4% (83)	14.0%	9.8%
Value Vague (VV)	Vague direction given for value.	*"decrease the height"* *"make the edges rounder"*	20.0% (146)	25.9%	15.0%
Value Total	-	-	31.2% (229)	39.9%	24.9%
Abstract (A)	Abstract change description.	*"make it look more stylish"* *"make it more playful"*	3.3% (24)	3.6%	3.1%

Table 4.6: Coding of the participants' requests during Task 2. Requests can either mention both properties and values, only properties or only values, or be abstract. The percentage of requests for each category are shown. The table also shows the percentage for each category for the first quartile (Q1) and last quartile (Q4) of participants' requests.

in Table 4.6). Several vague requests were due to participants not remembering the name of a property, but they were able to quickly remember them by seeing Stylette's predicted properties. In other cases, vagueness was to deliberately get the system to act in a certain way. Several participants (S5, S6, S7, S14, S18, S19) mentioned using requests as "macros"—being vague (e.g., "change font") so the system returned several related properties that could be changed in one go. Others (S1, S2, S4, S8, S15) used vague requests to explore what other styling changes they could make.

Regarding the value suggestions, there were three particular uses: (1) as a "starting point", (2) as a "guideline", or (3) as a "shortcut". For the first type, participants (S4, S11, S14, S17, S20) picked a suggested value and then manually adjusted it more to their preference. Others (S2, S5, S9, S10, S18) used the suggestions as a guideline—hovering through values to mentally map numerical differences to visual differences. Finally, as similar values would be suggested for similar components, several participants (S1, S7, S15) looked for the same suggestion when editing multiple similar components—as a sort of "value shortcut".

4.5.3 Self-Confidence Across Tasks

To understand how self-confidence changed depending on the tool used, we analyzed intra-condition differences in participants' responses (Fig. 4.8). Stylette participants' self-confidence increased significantly between the pre-survey (M=4.30, SD=1.12) and the end of Task 1 (M=5.13, SD=1.22, z=18.5, p=0.0035). Participants felt satisfied about completing Task 1, and mentioned that it was easy to learn about and make changes using Stylette: *"It gave me the feeling of learning and becoming familiarized with web development terms."* (S12). Surprisingly, DevTools participants' self-confidence also increased signif-

Request	Type	Expected.	Predidcted				
"change the font family to Helvetica" (S7)	PS,VS	FF	**FF**	FSz	FSt	FW	
"increase padding" (S8)	PS,VV	P	BW	M	**P**	W	
"change text color" (S6)	PS	C	BgC	**C**	FSt	O	TD
"change the picture radius to 24" (S18)	PP,VS	BR	BC	**BR**	C	FSz	W
"increase the size" (S16)	PP,VV	FSz	BR	BW	H	P	W
"change borders" (S11)	PP	BW	BC	BR	**BW**	C	W
"make it go in the middle" (S15)	PV,VS	TA	H	M	P	W	
"add some spacing at the bottom" (S2)	PV,VV	P	BR	H	M	**P**	
"change the distance" (S5)	PV	M	BW	C	H	P	W
"make this modern" (S19)	A	FF	BW	H	M	P	W

Table 4.7: A sample of participants' requests in Task 2, ordered from most specific to most vague/abstract. For each property, the table shows the request type, the property expected by the user, and the properties predicted by the system.

icantly between the pre-survey (M=4.01, SD=1.26) and Task 1 (M=4.81, SD=1.25, z=34.5, p=0.0148). Despite most of these participants not completing Task 1, they were satisfied with what they had accomplished as they expected that CSS code would be exceptionally challenging.

For similar reasons, DevTools participants' self-confidence increased between Task 1 (M=4.81, SD=1.25) and Task 2 (M=4.99, SD=1.32), although this was not statistically significant (t=0.540, p=0.595). These participants felt proud about their own effort and learning during the study: *"This is my first time handling [CSS] but I did this!"* (D14). In contrast, self-confidence for Stylette participants decreased significantly between Task 1 (M=5.13, SD=1.22) and Task 2 (M=4.58, SD=1.16, t=-3.204, p=0.0047). Unlike DevTools participants' self-reflective comments, Stylette participants' comments mostly focused on the tool. Some participants (S9, S16, S17, S20) mentioned how the system presented too many possibilities, making it difficult to decide on changes: *"It was hard [to choose] because the suggestions were all cute."* (S16). On the other hand, several participants (S4, S7, S11, S15) felt limited by the tool's possibilities—expecting the system to reveal new properties or support more complex changes (e.g., adding a "sparkle" animation).

4.6 Discussion

In this paper, we propose Stylette, a system that allows users to easily edit a website's design through a suggested set of properties and values generated from natural language requests. Stylette can be generalized to a variety of applications: expanded with a community feature for users to share website modifications, implemented as an IDE plugin to support web developers' help-seeking, or integrated into tools for user feedback. In this section, we further elaborate on the potential of Stylette and suggest opportunities for future work.

4.6.1 Stylette as a Web Designing Springboard

In our study, Stylette allowed users with no prior knowledge to quickly perform desired styling changes on websites. Unlike search engines that can take the meaning of queries "literally", our system

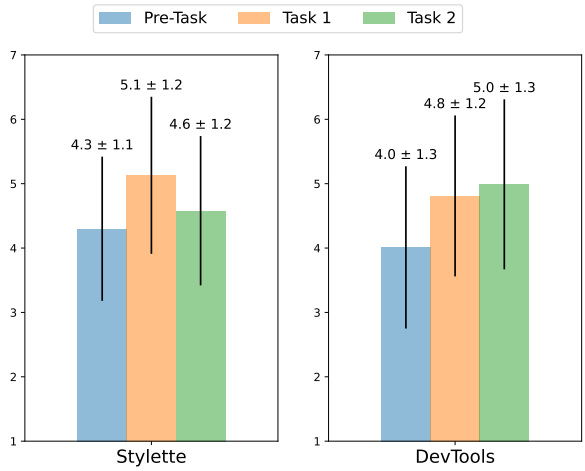

Figure 4.8: For both conditions, participants' reported self-confidence increased significantly between the pre-survey and the post-Task 1 survey. However, self-confidence decreased significantly for Stylette participants after Task 2, but did not change significantly for DevTools participants.

interpreted the vagueness behind users' requests to present more varied and suitable solutions. Stylette also allowed users to "learn-by-doing" by immediately testing the functions of properties by hovering on value suggestions—instead of having to skim through search results. As a side effect of interpreting vagueness, the system appeared to encourage creativity by presenting users with alternatives beyond their initial intentions. Together, these insights suggest that Stylette can support novices to explore and learn about CSS with continued usage.

The back-to-back tasks in our study provided a window into such continued usage of Stylette. We observed that users gradually developed knowledge about concrete CSS property names and values. For these now more knowledgeable users, the system still provided benefit: enabling the request of multiple properties for increased efficiency, supporting exploration of the design space, and helping users quickly remember forgotten information. However, we also observed that perceived effort could increase with continued usage and users' learning. This owed to the fact that, even after acquiring the knowledge to directly make changes by themselves, user still had to interact with the underlying, probabilistic AI—waiting for its processing and correcting any errors.

Thus, while Stylette is well-suited for novices to learn about CSS, its benefit may decrease with users' increasing knowledge due to the form of interaction. Elaborating on Amershi et al.'s guidelines [106], this suggests how human-AI interaction should be designed for over time use in the context of novice support systems. For future work, we propose an adaptive approach based on user knowledge: initial natural language interaction to help users acquire knowledge about properties, and then gradually exposing direct manipulation widgets for properties that users have acquired knowledge about—measured based on repeated usage of a property or the use of its specific name in requests.

4.6.2 Leveraging Large Language Models to Support Software Use

The grand scale of large language models (e.g., GPT-3 [58] or GPT-Neo [94]), in terms of architecture and datasets, has allowed them to perform previously unseen tasks with only a few data points. We leveraged this quality and the P-tuning technique [95] to allow novices to interact with website designs by constructing only a small dataset of 300 requests. Similar approaches can be taken to enable novices to

use natural language to use various complex software—overcoming the vocabulary problem [31]. While a rich body of work has enabled similar natural language interaction to support software usage [3, 38, 107, 37, 36], their approaches relied on a wealth of user-generated content. Thus, these approaches are not possible for new applications or features as such content might not exist. Moreover, as shown by the struggles of DevTools participants in our study, the language used in such content may also differ greatly from the vague language used by novices as the content is usually created by intermediate or advanced users. With our approach, in contrast, natural language interaction can be enabled for new applications with only the effort of creating a small dataset of examples, and, by including representative examples of novices' language, the support can be designed specifically for novices.

4.6.3 Natural Language Coding as a Learning Tool

Our natural language interface helps novices learn about a coding language by demonstrating how the code realizes high-level goals—lowering the selection, coordination, and use barriers identified by Ko et al. [108]. In addition, by exposing novices to multiple alternatives for an intended goal, we observed that our approach allowed users to acquire a greater breadth of knowledge about the code—familiarizing with more properties and learning new uses for properties. However, the study also revealed that DevTools participants appeared to feel more satisfaction about their learning experience when compared to Stylette. We suspect that this is due to DevTools participants expending more deliberate effort searching for and reading through resources. Based on these insights, we first suggest that natural language coding tools should provide multiple code alternatives for the same goal. Then, by incorporating interventions that prompt users to reflect on these alternatives—similar to prompts used in video learning [109]—to gain a wider understanding about the code through a deliberate learning experience.

4.6.4 Beyond CSS

Stylette aims to make the web more malleable for general users with no prior knowledge. Our work focuses on CSS code and allows novices to simply describe their high-level goal to start modifying it—without requiring the user to decompose the goal themselves [91] or look for examples [110, 111, 112]. However, websites are also composed of HTML (structure) and JavaScript code (functionalities). As structure-related changes might be more suitable for direct manipulation, Stylette could be combined with systems that already support this [84, 85]. Finally, to allow end-users to program new functionalities, models like OpenAI's Codex [61], which can generate JavaScript code from natural language descriptions, could be coupled with Stylette. By integrating these three types of support into one coherent system, future work could enable all users to fully access the web's malleability.

4.7 Limitations

Our work has several limitations which we address in this section.

- Stylette currently supports 16 different CSS properties. These were the ones used the most in the creation of our request dataset. While this limits our current implementation, more properties can easily be supported by expanding the dataset.
- In our evaluation, we compared Stylette against using DevTools and search engines. A possible concern is that DevTools participants could change more properties and might have misdirected

effort into these. Although the average DevTools participant only tried around two properties that were not supported in Stylette, we acknowledge that this could have affected results in Task 1.

- We relied on a dataset of 300 requests to train and evaluate our computational pipeline. While participants were generally satisfied with the pipeline's predictions, evaluating on a larger dataset would provide a better understanding of its performance. Also, while P-tuning has demonstrated high performance with even smaller datasets (N=32) [95], a larger dataset could increase our pipeline's performance and robustness.
- As we focused on a controlled evaluation of Stylette, it is still unclear how users would modify websites in the real-world. Future work could conduct a deployment study to understand how Stylette integrates into users' actual web experiences.

4.8 Conclusion

This paper presents *Stylette*, a novel system that allows users to describe a styling request in natural language to change the visual design of websites. By combining a GPT-Neo-based model and a convolutional VAE model, our computational pipeline processes the user's request and the component they clicked. The processed outputs are then combined to generate a *palette* of CSS properties and values that the user can experiment and iterate on to reach their desired style. A user-study revealed that Stylette could help users familiarize themselves with CSS properties in a shorter amount of time and with greater breadth. Insights from the study regarding the benefits and limitations of natural language support can guide the design of future work on novice support systems.

Chapter 5. Discussion

While diverse and powerful software programs, interfaces, and applications are developed every day, their maximum utility of these can only be reached if they are accessible and usable. This thesis makes a contribution by enhancing the accessibility and usability of design software through a high-level approach that supports lightweight input modalities and maps these modalities to enable complex design tasks.

Two techniques were introduced in this thesis were developed according to this approach: temporal mapping and contextual mapping. The temporal technique mapped the modalities against each other by extracting segments that were co-current in time and leveraged these segments to facilitate asynchronous message production and consumption. The contextual technique processed the modalities through machine learning models to extract additional related information that was used to support exploration of design style alternatives. These techniques demonstrated how supporting novices to interact in their desired form—instead of adapting to the software's input form—enabled them to better leverage the potential of design-related software.

While there is potential for applying the approach to other design tasks and even other domains, its application requires the careful distillation and consideration of users' needs and challenges with existing software. The techniques, and their respective interfaces, introduced in the thesis integrated users' needs in all three constituent components of the approach: input, mapping, and output. Below, I discuss design considerations regarding these three components.

5.1 Input

The two techniques proposed in this thesis focused on the same two modalities: clicks and voice. The main reason for this choice of modalities was that they were both lightweight, involving minimal effort from users, and it resembled how novices interacted in the real world, talking while pointing at objects. While other modalities could also match these two considerations (e.g., sketching), however, those were not chosen as the majority of users would not have access to the needed input devices (e.g., tablets). As the goal was to increase accessibility to novices, it was crucial that most novices had access to the needed devices—microphones and mouses being generally common. A final consideration was that these two modalities could be used concurrently. While various software support interaction through multiple modalities, these are rarely used in tandem (e.g., it is difficult to use both keyboard and mouse at the same time and voice interaction usually only involve voice). However, this limits the amount of information the system can receive and process at a time, which in turn limits the support it can provide. To summarize, there were four main considerations when deciding on input modalities: effort, familiarity, commonality, and co-currency.

5.2 Mapping

By mapping the multiple modalities against each other according to certain dimensions, the techniques are able to enhance the users' input and support interaction widgets for specific design tasks. As the computational pipelines automatically performs the mapping and bears the effort, the user can interact with the software through lightweight modalities and little effort. The techniques introduced used

a wide-variety of AI modules to perform the processing and mapping: speech-to-text, natural language processing, and computer vision. Recent advancements in AI have enabled these pipelines to produce outputs that have high performance and are human usable. Even so, however, careful considerations were made to ensure that novices could deal with incorrect outputs—e.g., showing UI components in the automatic transcripts and providing several styling options instead of only one. For similar techniques to work successfully, it is crucial that similar error-recovery mechanisms are included due to potential errors from the pipeline.

5.3 Output

By processing and mapping the modalities, the techniques introduce produce interaction widgets that allow users to perform design-related tasks: navigate messages on the design or iterate on the style of a website's components. Instead of implementing these widgets into separate, novice-specific software, these widgets were implemented as plugins for existing software. This design choice aimed to ensure that novices still become exposed to, interact with and, eventually, learn to use existing software. While these software may be difficult to use for novices, they contain a rich catalogue of powerful features and, with appropriate expertise, they can be used in effective and efficient workflows. Thus, as a separate consideration, it is beneficial to phase out the widgets produced by the introduced techniques as the users gain more expertise as they could gradually cause more harm than benefit.

Chapter 6. Conclusion

This thesis contributes the high-level approach of mapping multiple modalities to empower novices to perform design-related tasks. This approach was embodied in two novel interactive systems, *Winder* and *Stylette*, that support (1) asynchronous collaboration on a UI design, and (2) style editing of websites. In this chapter, we summarize the contributions of this thesis and outline three potential directions for future work.

6.1 Summary of Contributions

Winder is a system that supports asynchronous communication of novice team members collaborating on a UI design document. It allows the user to communicate by simply recording their voice while clicking on relevant components in the design document. By temporally mapping the segments in the voice recording with segments where a component was clicked-on and selected, the system is able to support interaction mechanisms that allow receivers to easily navigate within and through received messages. With the provided support, the system can help novice teams to more easily maintain a shared understanding of the design despite being distributed in space and time.

Stylette is an interface that allows end-users to edit and iterate on the style of any website. With the system, the user can simply click on a component in the website and explain in natural language how they wish the component's style would change. By using an NLP model to process the natural language request and a CV model to process the click on component, the underlying pipeline identifies plausible properties and values that can satisfy the user's request The output are then mapped into a palette of CSS properties and values which the user can navigate through and apply to the selected component. This allows end-users to easily edit the style of websites with no prior knowledge about CSS code.

6.2 Future Work

The high-level approach introduced in this thesis can be serve as a foundation upon which a diverse array of tasks can be supported.

6.2.1 Leveraging the Artifacts

The techniques introduced in the thesis involve multimodal input of voice and clicks. As a consequence, the resulting design artifacts are embedded with natural language expressions. For example, the UI designs produced with Winder contain the messages shared between team members, and the websites styled with Stylette have natural language requests alongside the changed components. The natural language expression and UI component pairs can be further processed to support a variety of tasks. Two promising ones are (1) design communication, and (2) design documentation. For the former, by compiling and further processing the justifications and descriptions left on a UI design with Winder, it could be possible to automatically generate a summary that explains why the UI was designed as such. For the latter, Stylette users could share specific styling edits they have made alongside the requests. The

resulting document could serve as learning resources for CSS beginners to understand how to accomplish certain styling changes.

6.2.2 Beyond Design

Similar approaches as those introduced in the thesis could be applied to other domains to increase usability and efficiency. By enabling similar multimodal input in programming or creative writing, the user could click on chunks of text, and record voice messages or make a voice request to generate a code or writing fragment. Similarly, by extracting objects from video frames, it could be possible to allow the user to navigate or edit a video by clicking on objects in the video and expressing an intended goal with their voice. Beyond usability, multimodal input and mapping techniques could be employed to increase the accessibility of interfaces. For example, to interact with a smartphone, blind or low vision (BLV) users must employ multiple gestures to perform an interaction that would require non-disabled user one gesture—e.g., swiping multiple times to select an app vs tapping on an app. By allowing BLV users to interact by using touch gestures while explaining their intentions in natural language, it could be possible to map the intentions behind the gesture and the natural language request to reduce the amount effort needed.

6.2.3 More Modalities, More Dimensions

In the thesis, the modalities focused on were voice and clicks, and the dimensions for mapping explored were time and context. However, the approach could support different and diverse tasks by applying it on other modalities and with different mappings. Beyond communication during UI design, the process of creating the design could be supported by allowing users to express a desired design by sketching and talking, and then mapping the sketches and natural language requests to generate plausible designs. With more advanced technology, other modalities such as gaze could also be leveraged to support varied interactions. For example, mapping gaze and typing temporally could allow users to reply to various comments in a discussion thread by simply gazing at the comments and typing—without using their mouse. The thesis investigated modality mapping through the temporal and contextual dimensions. Additional dimensions that could be investigated are visual (i.e., visual similarity or mapping visual attributes from various modalities), tonal (i.e., force or mood in the modality), spatial (i.e., co-current in space), etc.

Bibliography

[1] Sharon Oviatt. Ten myths of multimodal interaction. *Commun. ACM*, 42(11):74–81, November 1999.

[2] Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer supported cooperative work*, 11(3-4):411–446, 2002.

[3] Eytan Adar, Mira Dontcheva, and Gierad Laput. Commandspace: Modeling the relationships between tasks, descriptions and features. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 167–176, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Barbara L. Chalfonte, Robert S. Fish, and Robert E. Kraut. Expressive richness: A comparison of speech and text as media for revision. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, page 21–26, New York, NY, USA, 1991. Association for Computing Machinery.

[5] Susan Harkins. Insert voice comments into a word document, Nov 2009.

[6] Texthelp. Read&write literacy support software — texthelp, 2015.

[7] Otter.ai. Otter.ai: Otter voice meeting notes, 2019.

[8] Ian Arawjo, Dongwook Yoon, and François Guimbretière. Typetalker: A speech synthesis-based multi-modal commenting system. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, page 1970–1981, New York, NY, USA, 2017. Association for Computing Machinery.

[9] Patrick Ehlen, Matthew Purver, John Niekrasz, Kari Lee, and Stanley Peters. Meeting adjourned: Off-line learning interfaces for automatic meeting understanding. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, page 276–284, New York, NY, USA, 2008. Association for Computing Machinery.

[10] Gokhan Tur, Andreas Stolcke, Lynn Voss, Stanley Peters, Dilek Hakkani-Tur, John Dowding, Benoit Favre, Raquel Fernández, Matthew Frampton, Mike Frandsen, et al. The calo meeting assistant system. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1601–1611, 2010.

[11] Senthil Chandrasegaran, Chris Bryan, Hidekazu Shidara, Tung-Yen Chuang, and Kwan-Liu Ma. Talktraces: Real-time capture and visualization of verbal content in meetings. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–14, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Guang Li, Xiang Cao, Sergio Paolantonio, and Feng Tian. Sketchcomm: A tool to support rich and flexible asynchronous communication of early design ideas. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, page 359–368, New York, NY, USA, 2012. Association for Computing Machinery.

[13] Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. Richreview: Blending ink, speech, and gesture to support collaborative document review. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 481–490, New York, NY, USA, 2014. Association for Computing Machinery.

[14] Dongwook Yoon, Nicholas Chen, Bernie Randles, Amy Cheatle, Corinna E. Löckenhoff, Steven J. Jackson, Abigail Sellen, and François Guimbretière. Richreview++: Deployment of a collaborative multi-modal annotation system for instructor feedback and peer discussion. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work and Social Computing*, CSCW '16, page 195–205, New York, NY, USA, 2016. Association for Computing Machinery.

[15] Jeremy Barksdale, Kori Inkpen, Mary Czerwinski, Aaron Hoff, Paul Johns, Asta Roseway, and Gina Venolia. Video threads: Asynchronous video sharing for temporally distributed teams. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, page 1101–1104, New York, NY, USA, 2012. Association for Computing Machinery.

[16] Yasamin Heshmat, Carman Neustaedter, Kyle McCaffrey, William Odom, Ron Wakkary, and Zikun Yang. Familystories: Asynchronous audio storytelling for family members across time zones. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Yuan-Chia Chang, Hao-Chuan Wang, Hung-kuo Chu, Shung-Ying Lin, and Shuo-Ping Wang. Alpharead: Support unambiguous referencing in remote collaboration with readable object annotation. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, page 2246–2259, New York, NY, USA, 2017. Association for Computing Machinery.

[18] Soon Hau Chua, Toni-Jan Keith Palma Monserrat, Dongwook Yoon, Juho Kim, and Shengdong Zhao. Korero: Facilitating complex referencing of visual materials in asynchronous discussion interface. *Proc. ACM Hum.-Comput. Interact.*, 1(CSCW), December 2017.

[19] Saelyne Yang, Changyoon Lee, Hijung Valentina Shin, and Juho Kim. Snapstream: Snapshot-based interaction in live streaming for visual art. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[20] Steve Oney, Christopher Brooks, and Paul Resnick. Creating guided code explanations with chat.codes. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW), November 2018.

[21] April Yi Wang, Zihan Wu, Christopher Brooks, and Steve Oney. Callisto: Capturing the "why" by connecting conversations with computational narratives. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Elizabeth F. Churchill, Jonathan Trevor, Sara Bly, Les Nelson, and Davor Cubranic. Anchored conversations: Chatting in the context of a document. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, page 454–461, New York, NY, USA, 2000. Association for Computing Machinery.

[23] Sacha Zyto, David Karger, Mark Ackerman, and Sanjoy Mahajan. Successful classroom deployment of a social document annotation system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 1883–1892, New York, NY, USA, 2012. Association for Computing Machinery.

[24] Parmit K. Chilana, Amy J. Ko, and Jacob O. Wobbrock. Lemonaid: Selection-based crowdsourced contextual help for web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 1549–1558, New York, NY, USA, 2012. Association for Computing Machinery.

[25] Oliver J Sheldon, Melissa C Thomas-Hunt, and Chad A Proell. When timeliness matters: The effect of status on reactions to perceived time delay within distributed collaboration. *Journal of Applied Psychology*, 91(6):1385, 2006.

[26] Sun Young Hwang, Negar Khojasteh, and Susan R. Fussell. When delayed in a hurry: Interpretations of response delays in time-sensitive instant messaging. *Proc. ACM Hum.-Comput. Interact.*, 3(GROUP), December 2019.

[27] Amy X. Zhang and Justin Cranshaw. Making sense of group chat through collaborative tagging and summarization. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW), November 2018.

[28] Haiwei Ma, Bowen Yu, Hao Fei Cheng, and Haiyi Zhu. Understanding social costs in online question asking. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[29] Daniel Avrahami and Scott E. Hudson. Qna: Augmenting an instant messaging client to balance user responsiveness and performance. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, page 515–518, New York, NY, USA, 2004. Association for Computing Machinery.

[30] Martin Pielot, Rodrigo de Oliveira, Haewoon Kwak, and Nuria Oliver. Didn't you see my message? predicting attentiveness to mobile instant messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 3319–3328, New York, NY, USA, 2014. Association for Computing Machinery.

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

[32] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S. Lasecki, and Steve Oney. Codeon: On-demand software development assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 6220–6231, New York, NY, USA, 2017. Association for Computing Machinery.

[33] Nikhita Joshi, Justin Matejka, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. Micromentor: Peer-to-peer software help sessions in three minutes or less. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[34] Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. *Apparition: Crowdsourced User Interfaces That Come to Life as You Sketch Them*, page 1925–1934. Association for Computing Machinery, New York, NY, USA, 2015.

[35] Sang Won Lee, Yujin Zhang, Isabelle Wong, Yiwei Yang, Stephanie D. O'Keefe, and Walter S. Lasecki. Sketchexpress: Remixing animations for more effective crowd-powered prototyping of interactive interfaces. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 817–828, New York, NY, USA, 2017. Association for Computing Machinery.

[36] C. Ailie Fraser, Julia M. Markel, N. James Basa, Mira Dontcheva, and Scott Klemmer. Remap: Lowering the barrier to help-seeking with multimodal search. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 979–986, New York, NY, USA, 2020. Association for Computing Machinery.

[37] C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer. Replay: Contextually presenting learning videos across software applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery.

[38] Adam Fourney, Richard Mann, and Michael Terry. Query-feature graphs: Bridging user vocabulary and system functionality. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 207–216, New York, NY, USA, 2011. Association for Computing Machinery.

[39] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. Cueflik: Interactive concept learning in image search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 29–38, New York, NY, USA, 2008. Association for Computing Machinery.

[40] Youwen Kang, Zhida Sun, Sitong Wang, Zeyu Huang, Ziming Wu, and Xiaojuan Ma. Metamap: Supporting visual metaphor ideation through multi-dimensional example-based exploration. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[41] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. Attribit: Content creation with semantic attributes. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 193–202, New York, NY, USA, 2013. Association for Computing Machinery.

[42] Michelle S. Lam, Grace B. Young, Catherine Y. Xu, Ranjay Krishna, and Michael S. Bernstein. Eevee: Transforming images by bridging high-level goals and low-level edit operations. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[43] Nanxuan Zhao, Nam Wook Kim, Laura Mariah Herman, Hanspeter Pfister, Rynson W.H. Lau, Jose Echevarria, and Zoya Bylinskii. *ICONATE: Automatic Compound Icon Generation and Ideation*, page 1–13. Association for Computing Machinery, New York, NY, USA, 2020.

[44] Haijun Xia. Crosspower: Bridging graphics and linguistics. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 722–734, New York, NY, USA, 2020. Association for Computing Machinery.

[45] Gierad P. Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. *PixelTone: A Multimodal Interface for Image Editing*, page 2185–2194. Association for Computing Machinery, New York, NY, USA, 2013.

[46] Rada Mihalcea, Hugo Liu, and Henry Lieberman. Nlp (natural language processing) for nlp (natural language programming). In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 319–330, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[47] Chris Quirk, Raymond Mooney, and Michel Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888, Beijing, China, July 2015. Association for Computational Linguistics.

[48] Miltos Allamanis, Daniel Tarlow, Andrew Gordon, and Yi Wei. Bimodal modelling of source code and natural language. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2123–2132, Lille, France, 07–09 Jul 2015. PMLR.

[49] Xin Rong, Shiyan Yan, Stephen Oney, Mira Dontcheva, and Eytan Adar. Codemend: Assisting interactive programming with bimodal embedding. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, page 247–258, New York, NY, USA, 2016. Association for Computing Machinery.

[50] Viktor Schlegel, Benedikt Lang, Siegfried Handschuh, and André Freitas. Vajra: Step-by-step programming with natural language. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 30–39, New York, NY, USA, 2019. Association for Computing Machinery.

[51] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 577–589, New York, NY, USA, 2019. Association for Computing Machinery.

[52] Mehdi Manshadi, Daniel Gildea, and James Allen. Integrating programming by example and natural language programming. 2013.

[53] Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, July 2017.

[54] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609, Berlin, Germany, August 2016. Association for Computational Linguistics.

[55] Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings*

of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[56] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning, 2017.

[57] Sebastian Weigelt, Vanessa Steurer, Tobias Hey, and Walter F. Tichy. Programming in Natural Language with fuSE: Synthesizing Methods from Spoken Utterances Using Deep Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4280–4295, Online, July 2020. Association for Computational Linguistics.

[58] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[59] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps, 2021.

[60] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[61] Wojciech Zaremba, Greg Brockman, and OpenAI. Openai codex, 2021.

[62] Pamela J Hinds and Suzanne P Weisband. Knowledge sharing and shared understanding in virtual teams. In C.B. Gibson and S.G Cohen, editors, *Virtual teams that work: Creating conditions for virtual team effectiveness*, pages 21–36. Jossey-Bass, San Francisco, CA, USA, 2003.

[63] Chiara Rossitto, Cristian Bogdan, and Kerstin Severinson-Eklundh. Understanding constellations of technologies in use in a collaborative nomadic setting. *Computer Supported Cooperative Work (CSCW)*, 23(2):137–161, 2014.

[64] Google. Use comments & action items - computer - docs editors help, 2016.

[65] Barry M Kroll. Cognitive egocentrism and the problem of audience awareness in written discourse. *Research in the Teaching of English*, 12(3):269–281, 1978.

[66] Alonso H. Vera, Thomas Kvan, Robert L. West, and Simon Lai. Expertise, collaboration and bandwidth. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, page 503–510, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[67] Heng-Yu Ku, Hung Wei Tseng, and Chatchada Akarasriworn. Collaboration factors, teamwork satisfaction, and student attitudes toward online collaborative learning. *Computers in human Behavior*, 29(3):922–929, 2013.

[68] Gary M Olson and Judith S Olson. Distance matters. *Human–computer interaction*, 15(2-3):139–178, 2000.

[69] Linda Riebe, Antonia Girardi, and Craig Whitsed. A systematic literature review of teamwork pedagogy in higher education. *Small Group Research*, 47(6):619–664, 2016.

[70] James Tam and Saul Greenberg. A framework for asynchronous change awareness in collaboratively-constructed documents. In *International Conference on Collaboration and Technology*, pages 67–83, Berlin, Germany, 2004. Springer.

[71] Stephanie Bell. Project-based learning for the 21st century: Skills for the future. *The clearing house*, 83(2):39–43, 2010.

[72] Lene Pries-Heje and Jan Pries-Heje. Why scrum works: A case study from an agile distributed project in denmark and india. In *2011 Agile Conference*, pages 20–28, New York, NY, USA, 2011. IEEE.

[73] Pernille Bjørn, Morten Esbensen, Rasmus Eskild Jensen, and Stina Matthiesen. Does distance still matter? revisiting the cscw fundamentals on distributed collaboration. *ACM Trans. Comput.-Hum. Interact.*, 21(5), November 2014.

[74] Figma. Figma: the collaborative interface design tool., 2019.

[75] James A. Landay and Brad A. Myers. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 43–50, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[76] Yla R. Tausczik and James W. Pennebaker. Improving teamwork using real-time language feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 459–468, New York, NY, USA, 2013. Association for Computing Machinery.

[77] Jennifer Thom-Santelli, Dan R. Cosley, and Geri Gay. What's mine is mine: Territoriality in collaborative authoring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 1481–1484, New York, NY, USA, 2009. Association for Computing Machinery.

[78] Tovi Grossman, Justin Matejka, and George Fitzmaurice. Chronicle: Capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, page 143–152, New York, NY, USA, 2010. Association for Computing Machinery.

[79] Rowanne Fleck and Geraldine Fitzpatrick. Reflecting on reflection: Framing a design landscape. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*, OZCHI '10, page 216–223, New York, NY, USA, 2010. Association for Computing Machinery.

[80] Paige Abe and Nickolas A Jordan. Integrating social media into the classroom curriculum. *About Campus*, 18(1):16–20, 2013.

[81] Gayle Christensen, Andrew Steinmetz, Brandon Alcorn, Amy Bennett, Deirdre Woods, and Ezekiel Emanuel. *The MOOC phenomenon: who takes massive open online courses and why?* Available at SSRN 2350964, 2013.

[82] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2663–2672, New York, NY, USA, 2017. IEEE.

[83] Minsuk Chang, Ben Lafreniere, Juho Kim, George Fitzmaurice, and Tovi Grossman. Workflow graphs: A computational model of collective task strategies for 3d design software. In *Proceedings of Graphics Interface 2020*, GI 2020, pages 114 – 124, Toronto, 2020. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine.

[84] Michael Nebeling and Anind K. Dey. *XDBrowser: User-Defined Cross-Device Web Page Designs*, page 5494–5505. Association for Computing Machinery, New York, NY, USA, 2016.

[85] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. Crowdadapt: Enabling crowdsourced web page adaptation for individual viewing conditions and preferences. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, page 23–32, New York, NY, USA, 2013. Association for Computing Machinery.

[86] Jane Im, Sonali Tandon, Eshwar Chandrasekharan, Taylor Denby, and Eric Gilbert. *Synthesized Social Signals: Computationally-Derived Social Signals from Account Histories*, page 1–12. Association for Computing Machinery, New York, NY, USA, 2020.

[87] Xiong Zhang and Philip J. Guo. Fusion: Opportunistic web prototyping with ui mashups. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 951–962, New York, NY, USA, 2018. Association for Computing Machinery.

[88] Amanda Swearngin, Amy J. Ko, and James Fogarty. *Genie: Input Retargeting on the Web through Command Reverse Engineering*, page 4703–4714. Association for Computing Machinery, New York, NY, USA, 2017.

[89] Greasemonkey. Greasespot, 2021.

[90] Jan Biniok. Tampermonkey, 2021.

[91] Kesler Tanner, Naomi Johnson, and James A. Landay. *Poirot: A Web Inspector for Designers*, page 1–12. Association for Computing Machinery, New York, NY, USA, 2019.

[92] Google Chrome Labs. Visbug, 2018.

[93] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, page 489–500, New York, NY, USA, 2015. Association for Computing Machinery.

[94] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow, 2021.

[95] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too, 2021.

[96] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657, 2015.

[97] Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

[98] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics.

[99] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models, 2021.

[100] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[101] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. Gallery d.c.: Design search and knowledge discovery through auto-created gui component gallery. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.

[102] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. Webzeitgeist: Design mining the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 3083–3092, New York, NY, USA, 2013. Association for Computing Machinery.

[103] The Webby Awards. Top websites and mobile sites — the webby awards, 2021.

[104] DomCop. What is open pagerank?, 2021.

[105] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[106] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-ai interaction. In *CHI 2019*. ACM, May 2019. CHI 2019 Honorable Mention Award.

[107] C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. Discoveryspace: Suggesting actions in complex software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, DIS '16, page 1221–1232, New York, NY, USA, 2016. Association for Computing Machinery.

[108] Amy J. Ko, B.A. Myers, and H.H. Aung. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 199–206, 2004.

[109] Hyungyu Shin, Eun-Young Ko, Joseph Jay Williams, and Juho Kim. *Understanding the Effect of In-Video Prompting on Learners and Instructors*, page 1–12. Association for Computing Machinery, New York, NY, USA, 2018.

[110] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. *Bricolage: Example-Based Retargeting for Web Design*, page 2197–2206. Association for Computing Machinery, New York, NY, USA, 2011.

[111] Michael J Fitzgerald et al. *CopyStyler: Web design by example*. PhD thesis, Massachusetts Institute of Technology, 2008.

[112] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. *Designing with Interactive Example Galleries*, page 2257–2266. Association for Computing Machinery, New York, NY, USA, 2010.

Acknowledgment

First of all, I want to thank my advisor, Juho Kim, who was my introduction to this wonderful field, and whose trust, support, and endless stream of research and life advice has helped me grow as a researcher and as a person. I thank my collaborators Yoonseo Choi, DaEun Choi, and Seungsu Kim for all that they have taught me and for making this research become a reality. I also thank all of my colleagues at KIXLAB for their support, feedback, and for being there during the hectic paper deadline sprints. Finally, I want to thank my parents and my brother for their never-ending love.

Curriculum Vitae in Korean

이 름: 김 태 수

생 년 월 일: 1996년 08월 30일

전 자 주 소: taesoo.kim@kaist.ac.kr

학 력

2002. 8. – 2015. 6. Academia Británica Cuscatleca

2015. 8. – 2020. 2. 한국과학기술원 전산학부 (학사)

2020. 3. – 2022. 2. 한국과학기술원 전산학부 (석사)

경 력

2020. 8. – 2021. 6. 한국과학기술원 전산학부 조교

연 구 업 적

1. **Tae Soo Kim**, DaEun Choi, Yoonseo Choi, and Juho Kim, "Wonder: Styling the Web with Natural Language," *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22),* ACM (to appear).
2. Yoonjoo Lee, John Joon Young Chung, **Tae Soo Kim**, Jean Y. Song, and Juho Kim "Promptiverse: Scalable Generation of Scaffolding Prompts through Human-AI Knowledge Graph Annotation," *In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22),* ACM (to appear).
3. **Tae Soo Kim**, Seungsu Kim, Yoonseo Choi, and Juho Kim, "Winder: Linking Speech and Visual Objects to Support Communication in Asynchronous Collaboration," *In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21),* ACM.
4. **Tae Soo Kim**, Nitesh Goyal, Jeongyeon Kim, Juho Kim, and Sungsoo (Ray) Hong, "Supporting Collaborative Sequencing for Small Groups through Visual Awareness," *Proceedings of the ACM on Human-Computer Interaction, 5(CSCW1) (CSCW '21),* ACM.
5. **Tae Soo Kim**, Sungsoo (Ray) Hong, Nitesh Goyal, Jeongyeon Kim, and Juho Kim, "Consensus Building in Collaborative Sequencing with Visual Awareness," *In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20),* ACM.
6. Sungsoo (Ray) Hong, Minhyang (Mia) Suh, **Tae Soo Kim**, Irina Smoke, Sangwha Sien, Janet Ng, Mark Zachry, and Juho Kim "Design for Collaborative Information-Seeking: Understanding User Challenges and Deploying Collaborative Dynamic Queries," *Proceedings of the ACM on Human-Computer Interaction, 3(CSCW), (CSCW '19),* ACM.