

Collaborative Accessible AI Agents

Yi-Hao Peng

CMU-HCII-26-111

August 2026

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213

THESIS COMMITTEE

Jeffrey P. Bigham	CMU HCII & LTI (Co-Chair)
Amy Pavel	UC Berkeley EECS (Co-Chair)
Jodi Forlizzi	CMU HCII
Graham Neubig	CMU LTI
Ming-Hsuan Yang	UC Merced EECS / Google DeepMind

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy*

Copyright © 2026 Yi-Hao Peng

This research was supported in part by Apple, Adobe, Google,
the U.S. National Science Foundation, and the Jacobs Foundation.

Keywords: Human-Computer Interaction, Machine Learning, Accessibility, AI Agents, UI Agents, Human-AI Collaboration, Personalization, Generative UIs and Design.

To my family, for their love, care, and endless support.

Abstract

AI agents can now operate real user interfaces, produce visual artifacts, and carry out multi-step tasks from a natural-language request. Capability alone does not always decide who should make each choice. At many points in a task, an agent has to choose between acting on its own and bringing the person into the decision. The stakes are higher for blind and low-vision (BLV) people, who often cannot inspect the visual context the agent is working from. An agent may quietly pick among options the user never knew existed, make an error the user cannot verify, or settle on a result that looks reasonable but leaves the user no way to tell whether it is any good, let alone to their taste.

This dissertation takes up the question of who should choose in two parts. The first part asks how an agent can preserve the person’s agency while it executes. I built a UI agent that detects steps where the instruction leaves a real choice open, pauses there, describes the alternatives, and lets blind and low-vision users supply the preference the agent is missing [1]. In a study over real-world web tasks, participants working with this agent completed more tasks and reached outcomes that matched their intent more often than participants relying on fully autonomous agents. Interaction of that kind rests on perception, so I also developed a synthetic data pipeline that generates slides and interfaces as code, keeping structural labels attached, which lets models learn to parse visual layouts without large hand-annotated datasets [2]. The same question extends beyond a single application to open-ended activity, where an assistant has to recognize what a person is doing and what help they need [3], and to place events in time before it can judge when to step in [4].

The second part asks how an agent should act once a person hands a choice over (especially a visual one). When blind presentation authors used a system I built to describe visual design changes in shared slides [5], they asked for more than a record of what changed; they wanted enough access to the visual decisions to judge and revise the result. Some design judgment is shared across people, and a model trained on synthetic design flaws and professional ratings can score interface quality in ways that designers broadly agree on [6]. A shared standard only goes so far. I collected twelve thousand pairwise judgments from twenty professional designers over the same generated interfaces [7]; the designers disagreed with one another substantially, and models personalized to a single designer predicted that designer’s choices better than models trained on much larger pooled data. The dissertation closes with a computational method that learns a new user’s design preferences through a short adaptive onboarding and reranks the candidates of a generative UI model [8]. In an online study, a few pairwise choices guided personalized generation more effectively than a written description of the user’s taste.

This dissertation argues that agency in human-agent collaboration has to be allocated deliberately, decision by decision, rather than granted to the agent up front. The argument is rooted in accessibility, where the cost of a silent choice is easiest to see, but it applies wherever a person hands work to an agent. Each decision should sit depends on how uncertain the agent is, whether the person can reach the relevant state on their own, how much the outcome rests on personal judgment, and how difficult a mistake would be to detect or reverse. Recognizing the moments when a decision belongs to the person requires grounded perception and temporal context. Acting on a choice the person has delegated means learning that person’s preferences quickly, because disagreement between people is information about individuals rather than noise to average away. These requirements point toward agents that weigh task progress against the person’s control and taste, revisit that balance at every step as circumstances change, and carry what they learn about a person from one task to the next.

Acknowledgments

As suggested in my thesis title, this dissertation has also been a “collaborative” effort among many people, who I am really lucky to have had along the journey. If you were at my actual Ph.D. defense, you would probably recognize the following structure, but I am reusing some of it here while sort of trying to make it a bit more polished, and not just like me rambling to myself. As I am not someone who will ever win an award like an Oscar, I would like to use the rest of this writeup to thank everyone who supported me to get here. Speaking of Oscars, Matthew McConaughey once said in his acceptance speech in 2014 that he needed three things each day: something to look up to, something to look forward to, and someone to chase. I will borrow the same three things here, with my own interpretation of what each of them means to me.

First, to those I look up to. I would like to thank my undergrad advisors, Mike and Lung-Pan, who brought me into the world of HCI, and who are probably the only reason I ended up doing a Ph.D. at all. Thank you two for showing me how we can be creative and brave enough to chase the ideas we just want to chase, simply because we want to do it, without any specific reason. Then it comes to my Ph.D. advisors, Jeff and Amy, who have been supporting me over the past many years. I remember my first exchange with Jeff was me emailing you about the Ph.D. application results, since you were the program director at the time. I did not really imagine that you would become my advisor for all the years after. I remember everything seemed exciting after the results were out, and then COVID hit. That’s almost how I got to know Amy, who was a postdoc in Jeff’s group at the time. Because everything was becoming remote, Jeff suggested that the three of us could work together online, and I did not know that one suggestion would end up lasting for the next six years. There were a lot of ups and downs over the years, but what I really appreciate is that both of you are always kind, patient, and reasonable, and always try to think about what could work out best for all of us in any scenario, especially while bearing with my questionable English and communication skills all these years. I will definitely miss Jeff’s wisdom as a high-up in the company, delivered while running during our meetings, and Amy spending her nights on 1-1s with me, even peer-reviewing some spaghetti code I wrote before coding agents were even a thing. I was always afraid of becoming a professor, because I don’t think I would ever get close to 0.00001% of what any of you did, and I am still not sure if I will become one. But I hope that wherever I end up, I can bring part of you and the things you taught me with me: be kind, be reliable, and be proactive, and maybe one day become someone that others can look up to (even though I know I am still quite far away from that).

Second, to those I am looking forward to. I guess if you are trying to read this acknowledgment, then you probably are one of them. You know who you are, and I know you know I am talking about you! I always felt weird about name-dropping in the acknowledgments for friends, as I am afraid I would accidentally leave someone out. So here I will not mention any specific names. I would say you all are the unexpected gifts I got through this journey, who I have always been looking forward to over the past few years: who I look forward to learning a bit about the new research you did, who I look forward to finding new restaurants, ice cream places, and boba shops with when we are free, who I look forward to talking with about anything random in our current and future life, and who I look forward to sometimes just doing nothing with but lying down

on the grass and watching the sky. When I came to the U.S. for the first time six years ago, I never imagined I would end up having a group of friends I look forward to meeting every day, which ended up truly making Pittsburgh my second home. I have both a good memory and a bad memory. I have a good memory in that I believe I remember the first moment I met you, and even some random yet maybe fun or important moments that happened between us. I have a bad memory in that I probably have forgotten any bad things that happened between us (even though I doubt there are many), but if so, I would want to say you probably know I did not mean it. I truly look forward to crossing paths with all of you in the future, and I am definitely looking forward to the moment that comes.

Finally, to those I would like to chase: my family, including my parents and my sister. By chasing, I mean I really hope I can be wherever my family is, just like the way they let me chase my dream without any worries. The effort I would want to put into chasing them is similar to the effort they have put in for me to chase my dream. To my parents, I know you probably won't read this, as you don't really know much English. But when I was growing up, you still pushed me to learn English, even though I was always naughty and did not even want to do any homework for the English class, to the point where you got called by the teacher to come to school and look at how I wrote "I don't want to write this homework" so many times in the assignment. Little did I know I would end up doing a Ph.D. in the U.S. You have definitely taught me lots of lessons beyond the English language, and even more, the language of love, such as how to be kind and caring to people. I hope I will be able to chase the level of the love language you have spoken to me, and speak it back to you, as well as to the many people I love. I would also like to thank my sister, who was five years older than me (our birthdays were only one day apart). Growing up, I was always the kid following you and chasing after you. You have always been the role model I tried to chase. We always went to the same schools and the same tutors growing up, and every teacher told me I should learn from my sister, to be more hardworking, to listen to what the teachers said, and so on. I know there were some hard times for you over the years, and I think you probably don't know this even today, but you have been one of the main reasons why I wanted to do accessibility research, as I really want to do something that can help people. I have been doubtful of myself, even to this day. If I cannot help my own sister, then what is the point of my research? I hope I will spend my whole life chasing the answer to this. I also hope you won't blame me on the other side of the world, and hopefully you will be at least a little bit proud of who I am and what I did. One day, I will hopefully chase you to the place where you are, and before that, I will keep chasing the answer to the purpose of what I am doing.

If you have read all the way to this part, I would like to say one more time, thank you! Thank you for giving me these amazing years of my Ph.D. You are all my collaborative and obviously accessible agents — who I can look up to, learning about your research and how you manage a big team; who I can look forward to sharing the ups and downs of life with; and who I would like to chase, from the past, to the present, and into the future, a future in which I hope I will be able to meet you again. I spent the past few years trying to build agents that can collaborate with people, and it turns out the best ones have been around me all along. No agent I will ever build could replace any of you. Thank you, and see you in some corner of the world. :)

Contents

Abstract	iv
Acknowledgments	vi
1 Introduction	1
1.1 Surfacing Choices When User Intent Is Unclear	2
1.2 Choosing Well After a Decision Is Delegated	3
1.3 Accessibility as a Lens on Agency	4
1.4 Research Approach	4
1.5 Thesis Statement and Contributions	5
1.6 Dissertation Organization	6
2 Background	7
2.1 Agency and the Allocation of Work	7
2.1.1 Agency, autonomy, and mixed initiative	7
2.1.2 From GUI automation to collaboration	8
2.2 Accessibility and Unequal Access to Agent State	9
2.2.1 Nonvisual access and assistive agents	9
2.2.2 Interdependence and mixed-ability authoring	9
2.3 Representing Screens and Ongoing Activity	10
2.3.1 Structured understanding of interfaces and documents	10
2.3.2 From a screen state to an activity over time	10
2.4 Delegated Visual Decisions	11
2.4.1 Shared models of design quality	11
2.4.2 Generative and editable interfaces	11
2.4.3 Personal judgment and efficient preference learning	12
2.5 Open Problem and Dissertation Scope	12
I Preserving User Agency During Agent Automation	14
3 Morae: Proactively Pausing UI Agents for User Choices	15
3.1 Introduction	16
3.2 Related Work	18
3.2.1 Accessible Language-based UI Assistants	18
3.2.2 Language-driven Interactive UI Agents	18

3.2.3	Proactive AI Agents that Solicit User Input	19
3.3	Understanding Non-Visual Use of UI Agents	20
3.3.1	Field Study	20
3.3.2	Dataset Analysis	21
3.3.3	Follow-up Interviews	23
3.4	Morae	24
3.4.1	Dynamic Verification of Ambiguous Choices	24
3.4.2	Generative UI for Capturing User Choices	25
3.4.3	In-situ Feedback for Screen Reader Users	25
3.5	Technical Evaluation	26
3.5.1	Methods	27
3.5.2	Results	28
3.6	User Evaluation	29
3.6.1	Method	29
3.6.2	Results	31
3.7	Discussion	33
3.7.1	Task Success Rate for Current UI Agents	33
3.7.2	Levels of Pause Necessity	33
3.7.3	Multi-round User Preference Elicitation	34
3.7.4	Extending the Interaction Scope of Morae	34
3.7.5	Human Agency During Automation	35
3.7.6	Beyond Accessible Automation: Broader Agent-driven Applications . .	35
3.8	Conclusion	35
4	DreamStruct: Understanding Slides and UIs via Synthetic Data Generation	37
4.1	Introduction	37
4.2	Related Work	38
4.2.1	Understanding Slides	38
4.2.2	Understanding UIs	39
4.2.3	Synthetic Data Generation for Representation Learning	39
4.3	Methods	39
4.3.1	Generation Principles	39
4.3.2	Generation Pipeline	40
4.3.3	Synthetic Dataset	43
4.4	Experiments	43
4.4.1	Element Recognition	43
4.4.2	Image Captioning	45
4.4.3	Image Classification	47
4.4.4	Summary	48
4.5	Limitations	49
4.6	From Screen Structure to Ongoing Activity	49
4.6.1	Recognizing Behavior, Intent, and Help Needs	50

4.6.2	Grounding Events and Transitions in Time	50
4.6.3	From Representation to Intervention	51
4.7	Conclusion	52
II	Preserving User Agency in Delegated Decisions	53
5	Diffscriber: Describing Visual Design Changes to Support Mixed-Ability Collaborative Presentation Authoring	54
5.1	Presentation Authoring as Mixed-Ability Collaboration	54
5.2	Formative Study	55
5.2.1	A Common Division of Labor	56
5.2.2	What Changed in the Slides	56
5.3	Diffscriber	57
5.3.1	Slide Content and Change Views	57
5.3.2	Element Correspondence	58
5.3.3	Edit Classification and Description	59
5.4	Technical Evaluation	60
5.5	User Study	61
5.5.1	Change Awareness and Feedback	61
5.5.2	Limits of Description	61
5.6	Limitations and Extensions	62
5.7	Diffscriber as a Probe for Collaborative Agents	63
5.8	Conclusion	63
6	DesignPref: Capturing Personal Preferences in Visual Design Generation	65
6.1	From Shared Quality to Personal Judgment	65
6.2	DesignPref Dataset	66
6.2.1	Dataset Construction	67
6.2.2	How Much Do Designers Agree?	67
6.3	Why Designers Disagree	68
6.3.1	Common Themes	68
6.3.2	Divergent Trade-offs	69
6.4	Personalized Design Preference Models	70
6.4.1	Data Representation and Splits	70
6.4.2	UIClip Fine-Tuning	71
6.4.3	Personalized Retrieval-Augmented Generation	71
6.5	Evaluation	71
6.5.1	Personalized Fine-Tuning	72
6.5.2	Personalized Retrieval	72
6.6	Shared Constraints and Personal Trade-offs	73
6.7	Limitations and Open Questions	73
6.8	Conclusion	74

7	Efficient Personalization of Generative User Interfaces	75
7.1	The Cold-Start Personalization Problem	76
7.2	Adaptive Preference Modeling	77
7.2.1	Preference Data Bank	77
7.2.2	A User Profile as a Distribution over Prior Designers	78
7.2.3	Expected Information Gain for Query Selection	79
7.2.4	Personalized Retrieval and Pairwise Scoring	80
7.3	Offline Technical Evaluation	80
7.3.1	Protocol and Baselines	80
7.3.2	Results	81
7.4	Online Study of Personalized Generation	82
7.4.1	Participants and Procedure	82
7.4.2	Free-Text Profiles and Pairwise Evidence	83
7.4.3	Arena Results	84
7.5	Applications Beyond the Study	84
7.5.1	Personalized Generative Widgets	84
7.5.2	Personalized UI Configuration	85
7.5.3	Personalized Design Suggestions	85
7.6	Discussion	86
7.6.1	Population Data as a Prior Rather Than an Average	86
7.6.2	Why Active Elicitation Matters	87
7.6.3	From Explicit Onboarding to Continual Learning	87
7.6.4	Uncertainty Within a Person	88
7.6.5	Reranking Versus Personalized Generation	88
7.7	Limitations and Future Work	88
7.8	Conclusion	89
8	Discussion and Future Directions	90
8.1	Agency as a Decision-by-Decision Policy	90
8.2	Collaboration Across Longer Workflows	92
8.3	Continual Personalization Across Tasks and Contexts	92
8.4	Accessible, Verifiable, and Shared Artifacts	93
8.5	Conclusion	94
	References	95

List of Figures

1.1	The dissertation’s argument as a decision loop: understand the state, surface a choice when intent is unclear, choose within the delegated scope, and keep the outcome revisable.	2
3.1	Morae is an accessible UI agent that proactively pauses automation at key decision points for BLV users to make choices. For example, when asked to “ <i>add the cheapest sweetened sparkling water to my cart</i> ”, existing agents would automatically choose a product even if multiple items have identical prices. Instead, Morae pauses automation and presents relevant product differences (e.g., flavors, ratings) and allows users to choose based on their preferences. By proactively pausing when user input is necessary, Morae allows users to actively express preferences during UI automation.	15
3.2	TaxyAI’s interface features a task command input, start/pause automation button, and action history section displaying step-by-step agent reasoning. Here, TaxyAI autonomously executes the task “ <i>add the cheapest sweetened sparkling water to my cart</i> ” on Target but does not ask users for their preferences even though several choices exist.	21
3.3	Morae’s interface features a task command input field, a start/pause automation button, an action history panel that displays the agent’s step-by-step reasoning, and dynamically generated UI elements for preference selection. In this scenario, Morae autonomously executes the command “ <i>add the cheapest sweetened sparkling water to my cart</i> ” on Target. At a decision point that needs user input, Morae pauses, presents relevant information, and offers accessible interactive controls so BLV users can state their choices in a structured manner.	26
3.4	The UI of OpenAI’s Operator Agent. Users assign tasks via chat, monitor browser automation through live streaming and text updates, and manually intervene at any time. The example illustrates Operator autonomously executing the command “ <i>add the cheapest sweetened sparkling water to my cart</i> ” on Target without pausing to ask for the user’s choice.	27
3.5	The average ratings for each of three agent conditions (TaxyAI, Morae, Operator) across eight subjective evaluation criteria (Satisfaction with Choices, Awareness of Choices, Control over Choices, Ease of Choice-making, Awareness of Actions, Awareness of Results, Perceived Usefulness, Confidence in Use) for our user evaluation.	31

4.1	Overview of the DreamStruct pipeline. From left to right: In the <i>Ideation Phase</i> , screen descriptions for slides and mobile UIs are generated. In the <i>Generation Phase</i> , corresponding code is produced based on these descriptions, guided by the design principles and targeted labels. Finally, in the <i>Production Phase</i> , visual placeholders are replaced with actual data to render the final slides and UIs. . . .	40
4.2	An example of a generated slide description with the corresponding generated code and generated slide (top) with example CSS, HTML, and JavaScript excerpts from the generated code (bottom).	41
4.3	An example of a generated UI description with the corresponding generated code and generated UI. In the <i>Production</i> phase, our pipeline extracts alternative text and placeholder image links and uses Bing Search API or DALL·E to get a link or image to replace the placeholder.	42
4.4	Normalized Log Distribution of Slide and UI Elements	43
4.5	Performance of slide element recognition when adding training samples	44
4.6	Performance of UI element recognition when adding training samples	45
4.7	Pairwise win-rate matrices across captioning models	46
4.8	Performance of Slide and UI Image Classification when scaling up synthetic training samples	48
4.9	GUIDE separates user understanding into behavior state, immediate intent, and help prediction. The benchmark uses open-ended software activity rather than a fixed expert demonstration.	50
4.10	TEMPURA first learns event continuity through masked-event prediction, then grounds the learned event structure through timestamped segmentation and dense captioning.	51
5.1	Diffscriber aligns an author’s initial slide content with a collaborator’s revised presentation, then describes content, layout, and style changes. The descriptions let BLV authors review and respond to visual decisions that ordinary slide accessibility metadata does not expose.	55
5.2	The formative study found a recurring asynchronous workflow. BLV presenters authored content, sighted collaborators revised the visual design, and only some workflows included a structured review and feedback round.	56
5.3	Diffscriber provides two linked views of the same revision. Authors can begin from familiar slide content or from a summary of changes, then move between the views for more detail.	58
5.4	Participants rated Diffscriber more positively than accessible slides alone for understanding changes, reviewing the revision, and preparing feedback. . . .	62
6.1	UIClip performs best on design-choice prediction after UI-specific training. Synthetic JitterWeb pairs provide a clear quality signal, while the professional Better-App comparisons remain much harder.	66
6.2	DesignPref’s annotation interface presents a shared functional prompt and two generated variants. Each designer records both preference direction and strength.	68

6.3	A t-SNE projection of rationale embeddings reveals six recurring themes. Designers share a vocabulary of design concerns even when they choose different candidates.	69
6.4	Contested pairs reveal recurring trade-offs rather than arbitrary disagreement. Designers differ on density, palette, imagery, and the amount of functionality that should remain visible.	70
7.1	The personalization pipeline. A shared preference corpus provides repeated labels from prior designers. Adaptive onboarding estimates a distribution over those designers for a new user. The learned distribution then guides retrieval and reranking over newly generated UI candidates.	76
7.2	Adaptive onboarding represents a new user as a distribution over prior designers. After each pairwise answer, probability shifts toward designers whose prior labels are consistent with the answer. The system selects the next comparison to reduce uncertainty in the distribution.	78
7.3	Pairwise prediction accuracy as the onboarding budget k increases. The retrieval model performs best at every budget and improves to 0.620 at eight examples. GPT-5.2-Chat declines as more preference examples enter the prompt. The dashed line marks the pretrained UIClip baseline.	81
7.4	Arena ratings and aggregate win rates in the online study. The adaptive preference model achieves the highest rating and a 60.35% win rate, ahead of profile-conditioned reranking, profile-conditioned generation, and zero-shot generation.	83
7.5	A personalized widget editor. A learned profile proposes the initial style, while direct controls let the user override individual attributes. Each accepted or rejected edit can become another preference signal.	85
7.6	A reading application before and after preference-based auto-configuration. The system chooses among supported settings rather than generating arbitrary code, and later manual overrides refine the profile.	86
7.7	Preference-based reranking in a presentation editor. A generic gallery places one style first; the personalized gallery promotes the style that better matches the designer's learned taste.	87

List of Tables

3.1	Task completion rate and pausing performance for each method we benchmark in our technical evaluation.	29
4.1	Statistics of DreamStruct dataset.	42
4.2	The classification accuracy of each model for both slides and UIs (training samples: synthetic $\approx 5K$; human-annotated $\approx 0.5K$).	47
5.1	The change taxonomy derived from participant examples and 37 before-and-after slides.	57
5.2	Overall element-correspondence performance across the four before-and-after presentation sets.	60
6.1	Personalized and pooled models on each designer’s held-out comparisons. N denotes the per-designer training size; $20N$ denotes the full pooled set.	72
7.1	The four end-to-end generation conditions in the online study. The two reranking conditions share the same text profile and candidate pool, which isolates the quality of the final selector.	83
8.1	A decision policy for preserving human agency. The response depends on what the agent knows, what the person can inspect, whether the choice is personal, and how costly a mistake would be.	91

Chapter 1

Introduction

A capable agent still has to decide when not to decide.

Through an accessibility lens, this dissertation asks how collaborative agents should surface choices when user intent is unclear and choose well when users delegate decisions. The answer depends on what the agent knows, what the person can inspect, whose judgment the choice requires, and how hard an error would be to reverse.

AI agents have begun to move from conversation into action. A person can ask an agent to search a website, schedule an appointment, edit a document, compare products, or generate an interface. The agent then observes a digital environment, chooses a sequence of actions, and produces an outcome. Greater capability can remove substantial effort. Capability alone does not tell an agent when to surface a choice or how to choose well on a person's behalf.

The problem becomes especially clear for blind and low-vision (BLV) people. Many BLV people use screen readers that expose an interface through a sequential stream of speech or braille. Visual grouping, spatial relationships, hidden defaults, and changes elsewhere on the screen can remain difficult to inspect [9, 10]. An agent may see a screenshot, a DOM tree, or an accessibility structure that the person cannot review at the same speed or level of detail. The agent can therefore choose among options the person never knew existed, make an error the person cannot verify, or settle on a visually plausible result that does not match the person's taste.

The central concern of this dissertation is *human agency*. I use the term to mean a person's practical ability to understand the state that matters, shape consequential decisions, delegate work on their own terms, and revise the outcome. Agency does not require a person to perform every action or approve every click. A person can retain agency while an agent handles routine work. A person can also lose agency while still holding a nominal approval button if the system hides the relevant alternatives or gives too little information for an informed choice.

I use *decision authority* for the narrower question of who decides at a particular moment. Authority can move during a task. The agent may handle a routine and reversible step, return a preference-sensitive choice to the person, and later act again after the person supplies the missing judgment. A person may also hand a recurring choice to the agent after the system has learned enough about the person's preferences. The thesis is about how to make each shift deliberate rather than treating one natural-language request as permission for the agent to settle every later decision.

The dissertation develops the argument in two parts. The first asks how an agent can surface

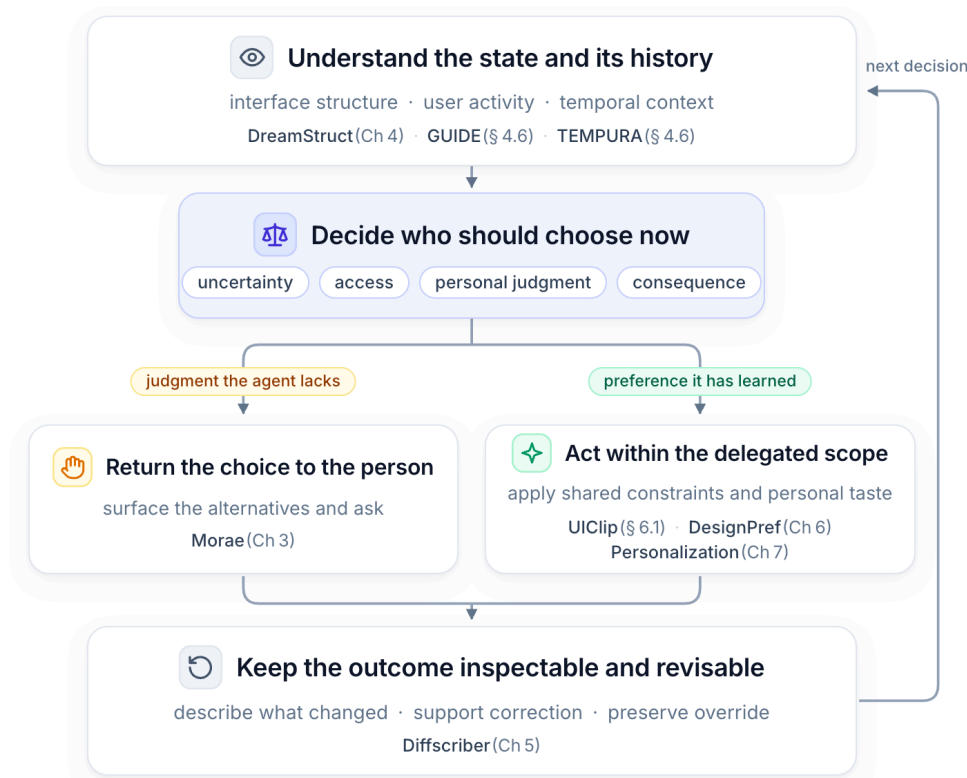


FIGURE 1.1: The dissertation’s argument as a decision loop. The agent maintains a grounded account of the interface and the activity around it (Chapter 4). At each consequential step, the agent surfaces an unresolved choice in an accessible form when the user’s intent does not determine the next action (Chapter 3) or chooses within the delegated scope under shared constraints and learned personal preferences (Chapters 6 and 7). The agent keeps the outcome inspectable and revisable (Chapter 5), and each completed decision changes the state that the next decision starts from.

a choice when the user’s intent no longer determines the next action. The second asks how an agent can choose well after a person delegates a decision, especially a visual one. Figure 1.1 shows how the projects fit together.

1.1 Surfacing Choices When User Intent Is Unclear

A natural-language request rarely specifies every decision that an agent will encounter. Consider the instruction “add the cheapest sweetened sparkling water to my cart.” The request names a product and a price rule. The interface may later reveal three products at the same lowest price, with different flavors, ratings, quantities, or delivery options. A fully autonomous agent can choose the first item and report success. The task may be complete according to a benchmark, while the interaction has removed a choice the person would have cared about.

The example exposes a timing problem. The relevant ambiguity may not exist at the start of the task. A long questionnaire before execution would ask for details that may never matter, and the person may not know which distinctions the interface will reveal. The agent has to move far enough to see the real options, recognize that the instruction no longer determines the next action, and ask only then.

Morae, presented in Chapter 3, follows that approach [1]. The agent checks the user's instruction against the current UI and action history at each step. When several valid actions depend on a missing preference, Morae gathers the information needed for comparison, pauses, and presents accessible controls through which the user can choose. The pause is tied to a concrete decision rather than to a generic request for confirmation.

A useful pause depends on accurate perception. An agent cannot expose a choice that the model failed to see, and it cannot explain an outcome when the model lacks a structured account of the interface. DreamStruct, presented in Chapter 4, generates slides and UIs from code and keeps the element structure, relationships, captions, and classes as labels [2]. The resulting synthetic data improves models across several levels of visual understanding without requiring a hand-annotated dataset for each task.

Longer workflows add another difficulty. An assistant may need to distinguish exploration from confusion, infer an immediate goal from a partial sequence, and decide whether help would support the user or interrupt useful work. GUIDE studies behavior state, intent, and help prediction in open-ended GUI activity [3]. TEMPURA studies the temporal event representations needed to place actions and transitions in time [4]. Chapter 4 discusses the two projects after the DreamStruct results because reliable intervention depends on both the structure of a screen and the history that led to it.

1.2 Choosing Well After a Decision Is Delegated

An agent cannot ask the user about every detail. Repeated questions can become disruptive, and many requests explicitly ask the system to choose. A design assistant may generate several interface variants and present one as a starting point. A coding agent may pick a widget style. A presentation tool may rank suggested layouts. Once the person hands the choice over, the agent still has to decide what a good result means for that person.

The problem first appears in this dissertation through mixed-ability presentation authoring. BLV authors often wrote the content of a presentation and relied on a sighted collaborator for visual design. The authors valued the collaborator's skill, but they could not always tell what had changed or give useful feedback on the result. Diffscriber, presented in Chapter 5, identifies and describes content, layout, and style changes between revisions [5]. The system shows why access to a final artifact is not enough. A person also needs access to the decisions embedded in the artifact if they are to review, challenge, or revise it.

Some design judgments can be learned across people. UIClip uses synthetic defects and professional ratings to learn a shared score for interface relevance and quality [6]. A shared score can reject broken layouts, unreadable text, or candidates that do not fit the prompt. The harder cases begin after several candidates satisfy the basic constraints.

DesignPref, presented in Chapter 6, keeps the identity behind each judgment instead of reducing all labels to a majority vote [7]. Twenty professional designers judged the same generated interfaces and disagreed substantially. The disagreements followed recognizable trade-offs in density, hierarchy, visual tone, imagery, and action prominence. Models trained for one designer predicted that designer's choices better than models trained on much larger pooled datasets.

The final chapter of this part asks how a system can learn a new person's preferences without requiring hundreds of labels. The method in Chapter 7 uses prior designers as a starting point, selects a small set of pairwise questions that distinguish among them, and updates a user-specific preference model after each answer [8]. The model then reranks new interfaces produced for new prompts. Shared data supplies a prior, while the person's own choices decide how the prior should change.

1.3 Accessibility as a Lens on Agency

Accessibility is more than an application domain in this dissertation. BLV interaction makes assumptions in agent design easier to see. Many systems presume that a user can watch the cursor, scan the alternatives, notice a default value, or compare a before-and-after state. When visual monitoring is unavailable, the system must decide what state to expose and how to make correction possible. The same problem appears for sighted users who face unfamiliar software, divided attention, small screens, remote execution, or a high cost of error.

Accessibility research also cautions against treating independence as the only measure of success. People often create access through interdependence, where collaborators contribute different skills and perspectives [11]. A BLV author may contribute domain knowledge and narrative structure while a sighted collaborator contributes visual design. An AI system can support the relationship by communicating changes and by giving each person influence over the decisions where their knowledge matters. The goal is not to remove collaboration. The goal is to make the distribution of work, information, and control understandable and revisable.

Ability-based design makes a related point: systems should account for what a person can do in context rather than assume a fixed average user [12]. For agents, the relevant question extends beyond whether a person can operate a widget. The agent must also consider whether the person can reach the evidence behind a decision, verify the outcome, and communicate a preference through an available modality.

1.4 Research Approach

The dissertation combines HCI studies, interactive systems, and machine learning. Formative studies identify where current workflows remove choices or hide state. System prototypes test concrete interaction mechanisms in real interfaces. Technical evaluations isolate pause detection, structured visual understanding, element correspondence, and preference prediction. User studies then ask whether the technical change improves task outcomes, awareness, control, feedback, or preference alignment. Three core ideas recur across the projects:

Represent the state that matters. Agents need more than a screenshot caption. They need interface elements, relationships, action history, and differences among candidate or revised artifacts. DreamStruct uses executable code as a source of structured labels. Diffscriber aligns elements across versions and turns the alignment into navigable change descriptions. The representations support both model inference and human inspection.

Ask for human input selectively. A question has value when the answer can change a meaningful decision. Morae waits until the interface reveals the relevant alternatives. The final personalization method chooses onboarding comparisons according to expected information gain. Both systems treat attention as limited and tie each question to uncertainty that matters for the outcome.

Use shared data without erasing the person. A new user rarely supplies enough data to learn a model from scratch. DreamStruct supplies broad perceptual supervision. DesignPref and the final personalization method use population data as a prior while preserving individual evidence. Shared regularities can help a system begin, but they should not define one universal objective for every user.

1.5 Thesis Statement and Contributions

The central statement of the dissertation is:

Through an accessibility lens, I build collaborative AI agents that surface choices when user intent is unclear and choose well when users delegate decisions.

The dissertation supports the claim through a series of contributions.

1. **An interaction mechanism that surfaces unresolved choices during UI automation.** Morae detects preference-sensitive decisions from the instruction, current interface, and action history. It gathers enough context for a useful comparison, pauses at the decision point, and gives BLV users an accessible way to direct the next action.
2. **Structured representations that make visual state and change available for action and review.** DreamStruct generates labeled slides and interfaces for element recognition, captioning, and classification. Diffscriber aligns presentation revisions and describes content, layout, and style changes so BLV authors can understand and respond to visual decisions.
3. **Preference-learning methods that help agents choose well after users delegate decisions.** DesignPref shows that identity-linked labels predict a person's design choices better than much larger pooled datasets. The final personalization method learns a new user through adaptive pairwise onboarding and applies the learned preferences to choose among new generative UI candidates.

The projects form one progression. Morae surfaces choices when the person's intent no longer determines the next action. DreamStruct and Diffscriber make the relevant visual state available. DesignPref and the final personalization method help agents choose well when people delegate decisions. Across the progression, agency depends on whether the person can reach, shape, and revise the decisions that matter.

1.6 Dissertation Organization

Chapter 2 reviews prior work on human agency and mixed initiative, assistive and collaborative agents, nonvisual access and mixed-ability authoring, multimodal understanding of interfaces and workflows, computational design assessment, and personalized preference learning.

Part I asks how agents can preserve agency while they act. Chapter 3 presents *Morae* through a field study, system design, technical evaluation, and BLV user study. Chapter 4 presents *DreamStruct*'s synthetic data pipeline and evaluations across slide and UI understanding, then considers how *GUIDE* and *TEMPURA* extend the perception problem to open-ended activity over time.

Part II asks how agents should handle visual choices that people delegate. Chapter 5 presents *Diffscriber*'s formative study, change representation, system, technical evaluation, and BLV user study. Chapter 6 begins with shared UI quality assessment through *UIClip*, then presents *DesignPref*'s identity-linked dataset, disagreement analyses, rationales, personalized fine-tuning, and retrieval experiments. Chapter 7 develops an efficient cold-start preference model and evaluates the model in an end-to-end generative UI study.

Chapter 8 returns to the thesis claim and develops three future directions: collaboration across longer workflows, continual personalization across tasks and contexts, and accessible shared artifacts that people and agents can inspect and revise together.

Chapter 2

Background

Agency depends on whether a person can understand, shape, and revise the work.

An agent can perform most of a task while the person still retains meaningful control over the decisions that matter. The literature in this chapter explains how collaborative agents can surface choices when user intent is unclear, choose well when users delegate decisions, and keep the relevant state accessible.

AI agents combine questions that earlier fields studied separately. Automation research asks which functions a machine should perform. Mixed-initiative interaction asks when the person or the system should take the next step. Accessibility research asks who can reach the information and controls that the interface exposes. Multimodal learning asks whether the agent can represent the screen and the activity around it. Computational design and preference learning ask how a system should assess several valid outcomes without erasing what one person wants.

The fields converge on two problems. During execution, an agent must recognize when the user's intent no longer determines the next action and surface the relevant choice in an accessible form. After a user delegates a decision, the agent must distinguish shared constraints from personal judgment so it can choose well for that person. Both problems bear on the same question: how can an agent take on useful work without weakening the person's practical ability to understand, shape, and revise the outcome?

2.1 Agency and the Allocation of Work

2.1.1 Agency, autonomy, and mixed initiative

HCI uses *agency* and *autonomy* to describe related but distinct properties. The terms can refer to control, independence, goal pursuit, or the ability to act according to personal values. A review of three decades of HCI research shows that formal control alone does not guarantee practical agency [13]. A person may hold an approval button yet lack the information needed for an informed choice. A person may also delegate substantial work and retain agency when the delegation remains understandable, bounded, and revisable.

Automation research offers a useful decomposition. Parasuraman, Sheridan, and Wickens separate information acquisition, information analysis, decision selection, and action implementation [14]. Recent work adapts the same concern to AI agents and argues for levels that distinguish

suggestion, shared execution, monitored autonomy, and independent action [15]. The decomposition matters because one natural-language command can shift several functions at once. An agent may observe the screen, infer the user’s goal, choose among alternatives, and commit the action before the user learns that a choice arose.

Mixed-initiative interaction treats control as something that can move during the task. Horvitz frames the handoff as a decision under uncertainty and interruption cost, while Hearst describes mixed initiative as a fluid exchange in which either party can contribute the next step [16, 17]. Human-centered AI later separated high automation from low human control and argued that strong systems should support both [18]. The resulting design question is more precise than asking whether an agent is autonomous: which decision should sit with which party at this moment, and what information makes the allocation meaningful?

Interactive machine learning and adaptive interfaces add two requirements. Users need an intelligible view of current system behavior, and their feedback should have a visible effect [19, 20]. Systems such as SUPPLE and mixed-initiative customization tools learned abilities or preferences while users accepted, rejected, or refined proposed changes [21–26]. The work also exposed a cost: an accurate adaptation can still harm predictability or spatial memory when the interface changes without a clear rationale [27]. Human-AI interaction guidelines and hybrid-intelligence frameworks reach the same conclusion at a broader level. Systems should communicate capabilities, surface relevant context, support correction, and coordinate human and machine strengths instead of treating model confidence as permission to act [28, 29].

2.1.2 From GUI automation to collaboration

Large language and multimodal models have made general-purpose GUI agents feasible. Mind2Web, WebArena, VisualWebArena, and WebVoyager evaluate web navigation from natural-language goals, while AppAgent, AssistGUI, UI-Vision, and VideoGUI extend the setting to mobile, desktop, and video-grounded tasks [30–37]. Most benchmarks reward task completion or action grounding. The agent receives a goal, executes a trace, and succeeds when the environment reaches a target state.

Task completion cannot settle every interaction question. Real requests omit constraints, admit several valid outcomes, and change as users learn about the interface. Creative and analytical work also uses exploration to form the goal. Studies of feature-rich software find that people often prefer full automation for routine work but guided or shared-control interaction for learning, exploration, and creative judgment [38]. CowPilot lets users pause, override, and resume web navigation; Magentic-UI exposes plans and intervention points; Cocoa supports co-planning and co-execution; and Plover makes the plan itself available for steering [39–42]. The systems shift the focus from whether a person can interrupt to whether the agent can coordinate responsibility without requiring constant supervision.

Proactive assistance sharpens the timing problem. Early work models intervention as a trade-off among expected benefit, interruption cost, and the risk of a wrong suggestion [16, 43]. Recent programming assistants show that a suggestion can improve productivity or disrupt the workflow depending on timing, salience, and relevance [44, 45]. ProMemAssist estimates assistance timing from a model of working memory, while CollabLLM and Proactive Agent train

language models to contribute before an explicit request [46–48]. ContextAgent broadens the observed context to open-world sensory input [49]. New benchmarks separate the decision chain further: ProAgentBench evaluates when and what to assist from long real-world work sessions, while KnowU-Bench evaluates preference elicitation, consent, and restraint in a live mobile GUI [50, 51]. GUIDE brings the problem back to open-ended software work and separates behavior-state detection, intent prediction, and help prediction [3]. The separation matters. Repeated clicks may signal confusion, deliberate comparison, or ordinary execution. An agent needs more than the current screen before it can judge whether a pause or suggestion will help.

2.2 Accessibility and Unequal Access to Agent State

2.2.1 Nonvisual access and assistive agents

Screen readers expose interfaces through speech or braille, often in a sequence that omits or weakens visual grouping, spatial relationships, and off-screen change. Studies of web use show persistent accessibility failures and a broader problem of “not knowing what you do not know”: users cannot request or verify information that the interface never makes discoverable [9, 10, 52]. The problem becomes more consequential when an agent sees screenshots, DOM nodes, accessibility trees, and remote browser state that the user cannot inspect at the same speed.

Natural-language access can reduce the cost of operating complex software. Capti-Speak and JustSpeak support speech-based control, conversational access work explores dialogue with screen readers, and Savant maps natural-language requests to screen-reader actions across desktop applications [53–56]. Such systems help users issue commands, but a convenient command channel does not automatically expose the state behind agent decisions. Morae addresses the gap by surfacing alternatives and pausing when the current instruction does not determine one outcome [1].

Recent evidence shows that computer-use agents also inherit a sighted interaction model. A11y-CUA compares blind and low-vision users, sighted users, and agents on the same desktop and web tasks, and finds large differences in interaction style and agent performance under assistive-technology conditions [57]. Studies of mainstream generative AI tools likewise report that blind users value new access to visual information but face uncertainty, inaccessible interfaces, and difficulty verifying model output [58, 59]. Work on accessibility alignment argues that an assistive agent must align its perception, action space, feedback, and evaluation with the user’s access method rather than place a conventional agent behind a voice interface [60]. For this dissertation, accessibility therefore means access to consequential agent state: the options the agent found, the action the agent plans, the changes the agent made, and the path to correction.

2.2.2 Interdependence and mixed-ability authoring

Accessibility research also challenges the assumption that independence requires one person to perform every part of a task. Interdependence treats access as a property that people and technologies construct together [11]. Studies of collaborative writing show that blind and sighted

collaborators value shared work but struggle with document state, change awareness, and inaccessible collaboration tools [61–64]. The relevant question is not whether help occurred. The question is whether each collaborator can understand the shared artifact and influence the decisions that shape it.

Visual authoring makes the gap concrete. Blind creators face barriers when they inspect artboards, arrange spatial content, or verify how a change affects the whole composition [65–67]. Diffscriber supports mixed-ability presentation authoring by identifying and describing content, layout, and style changes across slide revisions [5]. GenAssist gives blind creators structured ways to direct and inspect image generation [68]. Across the systems, access to the final artifact is only the starting point. Agency also depends on access to the edits, alternatives, and design reasoning that remain open for negotiation.

2.3 Representing Screens and Ongoing Activity

2.3.1 Structured understanding of interfaces and documents

An agent can coordinate a decision only when it represents the state that the decision depends on. UI datasets such as RICO and WebUI pair screenshots with element structure and metadata, while Spotlight and Ferret-UI connect screen regions to language [69–72]. CogAgent, OmniParser, and related GUI models improve visual grounding for action selection [73, 74]. UI-Vision extends evaluation to detailed desktop perception and interaction [36]. The line of work establishes that a screen is not an ordinary photograph. Text, controls, grouping, hierarchy, and interaction state all carry task-relevant meaning.

Structured visual documents raise similar problems. Slide-understanding systems recover layout, reading order, and element roles; accessible presentation systems use the structure to support nonvisual exploration and feedback [75–79]. A model that only captions the whole screen can miss the relations that matter for action, such as which label belongs to a field or which image replaced a text instruction.

Large labeled corpora are expensive because each screenshot may require boxes, classes, captions, and relationships. Synthetic data offers another route. Diffusion-generated images can improve representation learning at scale, and StableRep shows that synthetic images can support strong visual encoders [80, 81]. DreamStruct uses a more controlled source for slides and interfaces: executable code generates the pixels and also exposes element geometry, hierarchy, captions, and classes [2]. The pipeline supports several tasks from the same generation process and reduces dependence on a separate hand-annotation effort for each label type.

2.3.2 From a screen state to an activity over time

A static representation does not explain why the user reached the current screen. Screen-recording datasets and models begin to capture action sequences, narration, and workflow structure. GUI Narrator detects and captions GUI actions; VideoGUI and VideoAgentTrek learn from instructional or unlabeled computer-use videos; grounding work maps observed human demonstrations to agent actions [37, 82–84]. VideoWebArena evaluates long-context web tasks that require video

understanding [85]. GUIDE differs from automation benchmarks because the recorded users work through open-ended tasks, revise plans, explore features, and sometimes need help [3].

Temporal modeling remains difficult for multimodal models. TemporalBench tests fine-grained order and event relations; Momentor, LITA, TRACE, and VideoLLaMA2 add timestamp grounding, event modeling, or temporal instruction tuning [86–90]. TEMPURA trains a model first to infer missing events and causal links, then to segment and caption an untrimmed video with aligned timestamps [4]. The distinction between state and history is central to assistance. A menu that opens and closes repeatedly may indicate exploration, confusion, or deliberate comparison. Temporal context does not remove the ambiguity, but it gives the agent evidence that a single screenshot cannot provide.

2.4 Delegated Visual Decisions

2.4.1 Shared models of design quality

When a person delegates a visual choice, the agent needs some basis for selecting among candidates. Traditional evaluation begins with guidelines and expert inspection [91]. Computational measures estimate clutter, visual complexity, balance, or attention, and the Aalto Interface Metrics service packages several measures into a reusable toolkit [92–94]. Automated systems also detect specific usability or accessibility problems [95, 96]. Design tools use related models to search or organize alternatives, as in Sketchplore, Scout, GUIComp, and VINS [97–100].

Large multimodal models can produce open-ended critiques, but studies find instability, generic suggestions, and weak grounding in the artifact [101]. UICrit adds designer-authored region-level critiques and quality ratings, while UIClip learns a compact scoring model from synthetic defects and professional pairwise judgments [6, 102]. Recent work compares feedback modalities that designers can use to improve UI generators and finds value in comments, sketches, and direct manipulation beyond a single score [103]. The literature supports a layered interpretation of quality. Shared models can enforce prompt fit, readability, basic hierarchy, and functional constraints; they do not determine every valid trade-off among density, tone, imagery, and visual emphasis.

2.4.2 Generative and editable interfaces

Generative systems now produce interface code, presentation layouts, and interactive artifacts from screenshots or language. Pix2Code established screenshot-to-code generation, and StarVector and Design2Code extend the approach with modern multimodal models [104–106]. AutoPresent applies structured generation to presentation slides [107]. Malleable UI work explores interfaces that change their data model, layout, or level of detail at runtime, while Generative Interfaces studies when generated interaction can outperform a conversational response [108–110]. Other work cautions that unconstrained generation can weaken consistency and designer control [111].

Generation changes the role of assessment. A system may create many plausible artifacts for the same request and show only one. The selector therefore determines the user’s starting point and can hide alternatives that would require less later editing. Gradual Generation makes

intermediate customization choices visible during generation, and AlignUI uses preference data to guide generated controls for new tasks [112, 113]. Design Theater shows a complementary verification problem: a generator can offer a convincing design rationale while the rendered interface fails to implement the stated choice [114]. The work reinforces two requirements that recur in this dissertation. The result should remain editable, and the system should connect the visible artifact to the decisions and constraints that produced it.

2.4.3 Personal judgment and efficient preference learning

A shared quality model cannot represent every user. Personalized image-aesthetics research improves prediction when models retain rater identity, user attributes, or user-specific adaptation [115–118]. Goree et al. ask “correct for whom?” and show why aggregate accuracy can hide failure for particular people [119]. Bias analyses reach a similar concern for multimodal judges that claim to predict personal aesthetic preference [120].

Pairwise comparison provides a practical signal for subjective choice. Thurstone and Bradley-Terry models infer latent utility from comparisons, and modern reward learning and preference optimization use the same basic supervision for language and image generation [121–125]. Conventional pipelines often pool labels across raters or model annotator reliability [126, 127]. Pooling works when disagreement reflects random error around one target. Pooling loses information when people apply different but internally coherent objectives.

Personalized alignment methods condition reward models or policies on users, latent preference modes, or collaborative representations [128–131]. Hidden-context work also argues that a distribution over preferences can be more faithful than one modal answer [132]. The practical obstacle is cold start. A new user supplies only a few labels, so the system has to share statistical strength without treating the population average as the final objective. Recent methods study inference-time adaptation from a small number of preference queries [133, 134]. Active elicitation selects comparisons that reduce uncertainty or improve downstream decisions, rather than asking a random set of questions [135–137]. Research on interface personalization also shows that people need to understand and correct how interaction data influences adaptation [138].

2.5 Open Problem and Dissertation Scope

The literature leaves three linked problems.

The agent has to surface choices when user intent is unclear. Task success does not reveal whether an agent silently resolved an underspecified or preference-sensitive choice. A useful handoff requires enough progress to expose the real alternatives, enough context to ask a focused question, and a policy that weighs uncertainty against interruption cost.

The relevant state has to be available to both parties. An agent cannot ask about an option that the model failed to perceive. A person cannot meaningfully answer when the interface hides the evidence, the action history, or the consequences. Accessibility makes the asymmetry visible, but the same requirement applies whenever an agent acts quickly across remote tools or generated artifacts.

The agent has to choose well after a decision is delegated. Shared constraints can reject broken or irrelevant outputs. Several valid outputs can still encode different tastes. Preference data should retain whose judgment produced each label, and a deployed system should learn from sparse personal evidence without erasing the useful prior supplied by other users.

Part I addresses the first two problems. *Morae* surfaces unresolved choices for blind and low-vision users. *DreamStruct* develops structured supervision for the perception that such coordination requires. *GUIDE* and *TEMPURA* extend the representation from one screen to the user’s evolving activity and its temporal structure. Part II addresses delegated visual decisions. *Diffscriber* makes the choices embedded in a revised artifact available for review. *UIClip* supplies a shared design-quality signal. *DesignPref* and the final personalization method retain individual judgments and learn a new user’s preferences so the agent can choose well after delegation.

Part I

Preserving User Agency During Agent Automation

An agent that automates work on a person's behalf makes many small decisions along the way, and most of them never surface. The two chapters in this part ask how a collaborative agent can surface a choice when the user's intent no longer determines the next action. Chapter 3 presents a UI agent that recognizes when an instruction leaves a choice open and returns the choice in a form that blind and low-vision users can act on. Chapter 4 builds the structured visual understanding required to recognize and explain the available choices, then follows the perception problem from a single screen out to open-ended activity over time.

Chapter 3

Morae: Proactively Pausing UI Agents for User Choices

A person cannot choose among alternatives the agent never reveals.

Morae lets a UI agent advance until the interface makes an unresolved choice concrete. The agent then gathers the details needed for comparison, pauses before commitment, and presents the options in an accessible form. The chapter tests whether such pauses improve task success and help blind and low-vision users reach outcomes that match their own preferences.

🗣️: “Add the cheapest sweetened sparkling water to my cart”

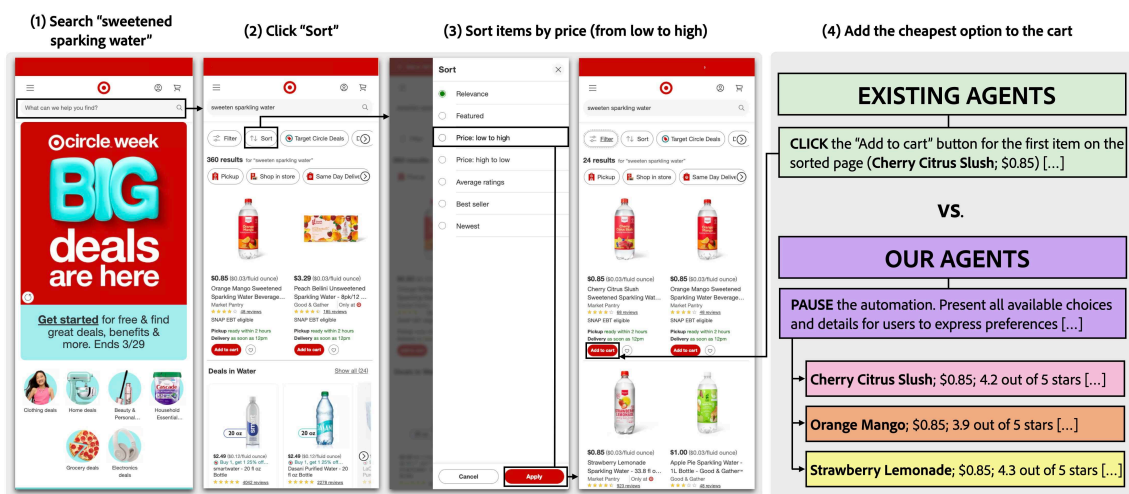


FIGURE 3.1: Morae is an accessible UI agent that proactively pauses automation at key decision points for BLV users to make choices. For example, when asked to “add the cheapest sweetened sparkling water to my cart”, existing agents would automatically choose a product even if multiple items have identical prices. Instead, Morae pauses automation and presents relevant product differences (e.g., flavors, ratings) and allows users to choose based on their preferences. By proactively pausing when user input is necessary, Morae allows users to actively express preferences during UI automation.

User interface (UI) agents promise to make inaccessible or complex UIs easier to access for

This chapter is based on work published at the 38th Annual ACM Symposium on User Interface Software and Technology (UIST 2025) [1], with Dingzeyu Li, Jeffrey P. Bigham, and Amy Pavel.

blind and low-vision (BLV) users. However, current UI agents typically perform tasks end-to-end without involving users in critical choices or making them aware of important contextual information, thus reducing user agency. For example, in our field study, a BLV participant asked to buy the cheapest available sparkling water, and the agent automatically chose one from several equally priced options, without mentioning alternative products with different flavors or better ratings. To address this problem, we introduce *Morae*, a UI agent that automatically identifies decision points during task execution and pauses so that users can make choices. *Morae* uses large multimodal models to interpret user queries alongside UI code and screenshots, and it prompts users for clarification when a choice requires their input. In a study over real-world web tasks with BLV participants, *Morae* helped users complete more tasks and select options that better matched their preferences, as compared to baseline agents, including OpenAI Operator. More broadly, this work exemplifies a mixed-initiative approach in which users benefit from the automation of UI agents while being able to express their preferences.

3.1 Introduction

A grand challenge in accessible computing is to make user interfaces (UIs) both accessible and usable for people who are blind or low-vision (BLV) [9, 52]. Accessibility errors can make a UI impossible to access or severely reduce its usability [9]. Many BLV people use screen readers, which convey UIs in a linear form that can become slow and confusing as interfaces grow more complex [9, 10]. Agents that help users navigate UIs offer one promising response to these challenges [139, 140], although such agents had limited capabilities until recently.

Recent advances in large language and multimodal models (LLMs/LMMs) have resulted in much more capable UI agents [141–144] that execute complex automation tasks based on natural language instructions. For example, on a site like Target [145], a user can instruct an agent to “*add the cheapest sweetened sparkling water to my cart.*” The agent then finds the search bar, inputs the query, sorts the results by price, and selects the first option (Figure 3.1). However, existing UI agents assume that users can visually follow the agent’s actions and understand the surrounding context. This assumption does not hold for BLV users. Consequently, even if the agent successfully adds the cheapest sweetened sparkling water, a BLV user may remain unaware that three equally priced but differently rated flavor options were available, some of which the user might have preferred (Figure 3.1).

To understand the challenges and opportunities that blind users face with UI agents, we conducted a field study with four BLV people who used an open-source agent for real-world automation tasks. The system recorded their queries, task outcomes, and interaction traces. Our interviews with first-time users revealed enthusiasm about automating tasks such as online shopping and scheduling meetings by simply providing a few words, yet participants also noted several challenges. Participants sometimes had difficulty identifying the actions available in unfamiliar interfaces. Five percent of user queries requested tasks that the current interface could not support, which reflects the “not knowing what you do not know” problem [10]. Among executable tasks, 19% were underspecified because required fields or constraints remained unclear, and 13% presented several valid options, such as equally cheap hotels or equally fast flights. In

both cases, the agent could choose an arbitrary option without giving the user a meaningful choice. Users also found it difficult to track the agent’s actions and wanted stronger verification of potential errors.

To address these challenges, we introduce *Morae*¹, an accessible UI agent that helps BLV users express preferences during UI automation. *Morae* proactively pauses at decision points whenever the user’s preference is unclear. *Morae* leverages a large multimodal model (LMM) to interpret natural language commands and analyze UI representation (e.g., the web DOM) and screenshots. To balance effective task execution with user control, *Morae* introduces a mechanism called *Dynamic Verification of Ambiguous Choices*, which internally verifies potential ambiguities while considering the progress of task execution. At each automation step, *Morae* generates and answers internal clarification questions based on relevant metadata such as the user’s queries, current UI state, and previous action history. When *Morae* identifies a critical ambiguity that requires user input, the agent pauses and asks for clarification (Figure 3.1). *Morae* also dynamically generates accessible, interactive controls that let users specify their choices before automation continues (Figure 3.3). To further support screen reader users, *Morae* provides real-time audio feedback synchronized with each agent action. Users can explicitly verify task outcomes after each automation. *Morae* also supports users by describing which tasks are available and how users can complete these tasks, including step-by-step instructions for screen reader-based interactions (e.g., relevant keyboard shortcuts).

We evaluated *Morae* through a technical assessment of its core component (i.e., the pause detection) and a user study examining its interaction experience. In our technical evaluation, we tested *Morae* across 256 tasks covering 8 UI types. Results indicated that *Morae*’s approach, based on dynamic verification of ambiguity, significantly outperformed direct prompting, OpenAI Operator, and other baseline methods. *Morae* achieved higher task success rates and demonstrated superior performance in correctly detecting necessary pauses. In our user study involving 10 BLV participants, *Morae* enabled participants to make decisions that better aligned with their preferences while maintaining more diverse choices. Users reported a stronger sense of control over the choices they made and achieved higher automation success rates compared to both the fully automated baseline agent and OpenAI Operator.

In summary, our contributions are threefold: (i) We release the first dataset of real-world interactions between blind users and UI agents; the dataset supports research on human-agent interaction beyond task automation. (ii) We introduce *Morae*, an accessible UI agent that proactively pauses at unclear decision points, prompts users to express their preferences through generative UI, and provides in-situ feedback for informed decision-making. (iii) A comparative study shows that *Morae* enables BLV users to express preferences more clearly and achieve higher task-completion rates than off-the-shelf UI agents.

¹*Morae*, the Latin plural of *mora*, means pauses or intervals. In linguistics, a *mora* marks the beat that measures syllable duration. The notion of incremental timing parallels the discrete pauses in our agent’s action sequence during task automation.

3.2 Related Work

Morae builds on three research strands: (i) accessible language-based UI assistants, (ii) language-driven interactive UI agents, and (iii) proactive AI systems that solicit user input.

3.2.1 Accessible Language-based UI Assistants

Language-based assistants transform the way BLV users interact with digital devices. Mainstream assistants such as Siri, Alexa, and Google Assistant now perform daily tasks for users such as setting reminders or initiating phone calls via voice commands. However, these assistants primarily manage general system-level functions and rarely handle complex, application-specific interactions. Application-specific macros [146] help automate repetitive actions within specific programs but generally lack flexibility for natural language interaction. Recent research explores conversational UI agents for BLV users to bridge the gap between natural language commands and application-level interactions [53–56, 147]. For example, JustSpeak [54] enables voice-based control of mobile applications, allowing users to operate apps through speech. Similarly, CaptiSpeak [53] and ConWeb [147] support web browsing through natural language commands, providing more accessible online interactions. Phutane et al. [55] explored conversational agents that encourage UI exploration through dialogue-based interactions. More recently, researchers introduced Savant [56], an LLM-powered desktop agent for BLV users. Savant automates desktop tasks by translating natural language commands into a series of predefined screen reader actions. This approach significantly improves efficiency and usability compared to traditional screen readers. However, like most user interface agents, Savant does not proactively seek clarification when user choices are unclear. Our study with BLV users identifies a critical limitation: users frequently give ambiguous commands without awareness, and existing agents rarely prompt users to clarify their preferences. This lack of proactive clarification leads to missed opportunities for users to express their unique preferences. We thus introduce an agent that proactively pauses at critical choice moments and allows users to specify their preferences.

3.2.2 Language-driven Interactive UI Agents

The idea of interacting with computers through natural language dates to SHRDLU, a pioneering conversational system introduced in 1968 [148]. SHRDLU allowed users to manipulate virtual objects in a simplified “block world” through English commands and shaped later research on natural-language interfaces in human-computer interaction and machine learning [149–156]. Today, language-driven agents like TaxyAI [157] and Browser Use [158] extend natural language interfaces to automate real-world web tasks. Recent research has also focused on using LLMs and LMMs to improve the performance and reliability of these autonomous UI agents. Kim et al. [142] demonstrated that LLMs efficiently complete tasks in simplified UI benchmarks like Mini-WoB++ [141], even from minimal demonstrations. Similarly, AutoWebGLM [143] further showed specialized language agents surpass GPT-4 on realistic web automation tasks in WebArena [31] and Mind2Web [30].

Recent research on UI automation explores richer interactions between users and agents. MoTIF [159] introduced a dataset for interactive vision-language navigation that captures ambiguous or infeasible commands. META-GUI [160] developed multimodal conversational agents that interact directly with mobile interfaces through natural conversation without a backend API. LLM4UI [150] used large language models to perform mobile UI tasks such as screen summarization, question answering, and action execution through prompting. WebLINX [161] enabled conversational guidance for complex web navigation tasks through multiple dialogue turns. Most recently, CowPilot [39] introduced explicit buffer periods during automation. These buffer periods let users intervene, stop, or correct agent actions and support task completion through human-agent collaboration. Despite significant progress, existing research generally assumes that users can visually track and verify agent actions. Thus, UI automation designed for blind or low-vision (BLV) users remains largely unexplored. Our research addresses this critical gap by creating UI agents that proactively pause at unclear decision points, explicitly prompt BLV users for clarification, and provide clear audio and screen-reader-friendly feedback. Deliberate pauses and tailored feedback help BLV users participate actively and make informed, personal choices during UI automation.

3.2.3 Proactive AI Agents that Solicit User Input

Early research on mixed-initiative interfaces established continuous dialogue as a method to clarify user intent and enhance system effectiveness [16]. For example, LookOut proactively extracts scheduling tasks from user emails [16], and SearchBot anticipates conversational breakdowns by proactively suggesting relevant search queries [43]. These foundational works inspired subsequent research on conversational agents that proactively initiate dialogue to enhance task performance and communication efficiency [162, 163]. Recent research extends proactive conversation into multimodal and situated contexts. In vision-language navigation tasks, agents proactively request user clarifications to resolve ambiguous instructions, which directly improves navigation accuracy [164–166]. Embodied interaction environments such as TEACH [167] and collaborative dialogue scenarios in Minecraft [168, 169] illustrate proactive strategies for three-dimensional space. These agents actively clarify task-related ambiguities and align mutual understanding among users, environments, and agent goals. Similarly, in human-robot interaction, proactive techniques enable robots to detect and resolve misunderstandings during collaborative tasks. Examples include inverse semantics frameworks that request human assistance [170], navigation strategies enhanced by retrospective curiosity-driven clarification [171, 172], and explicit frameworks addressing communication breakdown and recovery [173, 174]. Recent work has also explored context-aware, domain-specific proactive agents. For instance, researchers have developed creation agents [175] that pose clarification questions whenever multi-turn text-to-image generation encounters uncertainty, and programming agents that examine code context to propose improvements [44] or flag underspecified issues [176]. Our research builds on earlier work and introduces an agent that proactively asks BLV users to confirm choices whenever it cannot infer their preferences during UI automation. By offering explicit clarification steps through an agent-driven self-verification process, our approach preserves user agency and maintains task effectiveness during automated workflows.

3.3 Understanding Non-Visual Use of UI Agents

Prior research showed that desktop UI agents can simplify interactions for BLV users compared with traditional screen readers [56]. However, little work has examined the limitations that BLV users encounter when they adopt UI agents in the wild. We conducted a one-week field study in which four BLV participants used a UI agent for everyday web tasks. We recorded the tasks they attempted, the agent’s actions, and the resulting outcomes. We then interviewed each participant about the agent’s benefits and shortcomings. All studies were approved by the institution’s IRB.

3.3.1 Field Study

Participants. We recruited four BLV participants (three female and one male; the original paper reports full demographic details [1]) through mailing lists. Two participants were totally blind, and two were legally blind. All participants regularly performed web tasks in professional roles that included accounting, software engineering, and education. None had prior experience with UI agents.

Study Apparatus. For data collection, we built on TaxyAI [157], an open-source browser agent powered by an LLM, with GPT-4 as the default model. Like many recent UI agents from research [142–144, 155, 156] and industry [158, 177, 178], TaxyAI translates a user command and a representation of the web interface into actions such as clicks and text entry. To fit the DOM within the model context, the agent recursively removes invisible elements, redundant text nodes directly under body, and empty or non-interactive elements. It retains interactive elements identified by accessibility attributes such as `aria-label`, `role`, and `name`, assigns each a unique identifier, and merges single-child nodes when appropriate. The resulting DOM contains the information needed for task execution without much of the original page complexity.

The agent then matches the command to relevant elements and executes a sequence of actions. For example, given “add the cheapest sparkling water to my cart,” the agent searches, sorts the results, and adds an item (Figure 3.1). A numbered action history lets users follow the agent’s reasoning and interactions (Figure 3.2). We modified TaxyAI to accept a screenshot alongside the DOM, log each model response, and capture the interface after every step. We also added the safety-check prompt from prior work [179] to prevent full automation of high-risk actions such as checking out an order or deleting files from cloud storage. We created a written guide to help BLV users install and use browser-based UI agents. The guide explains UI automation, shows how to read an agent’s action history, and describes potential outcomes and risks through concrete examples.

Data Processing and Annotations. We collected 638 unique user queries from BLV participants along with corresponding automation steps and results. Three participants used Windows (two primarily used NVDA and one used JAWS), and one participant used macOS with VoiceOver. Two annotators independently reviewed each collected query using a structured annotation process with three stages. First, annotators classified each query as (i) valid and feasible, when the UI supported a meaningful request; (ii) valid but infeasible, when the request was meaningful but unsupported by the UI; or (iii) invalid, when the query did not express a meaningful automation

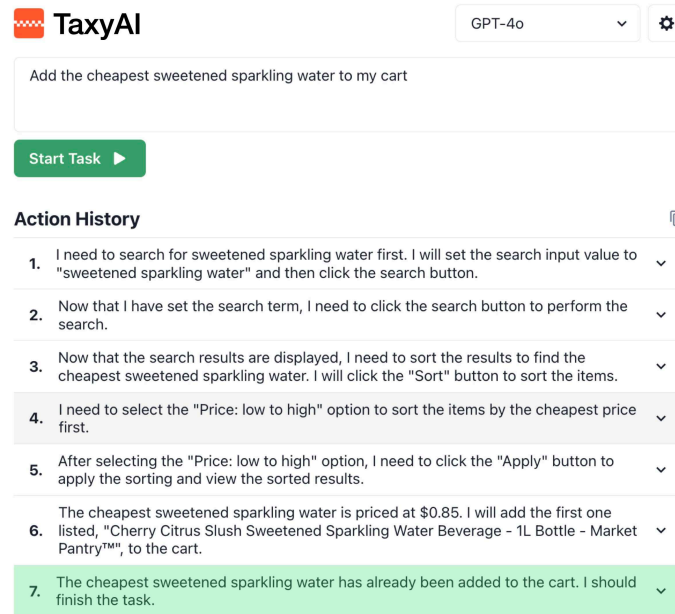


FIGURE 3.2: TaxyAI’s interface features a task command input, start/pause automation button, and action history section displaying step-by-step agent reasoning. Here, TaxyAI autonomously executes the task “*add the cheapest sweetened sparkling water to my cart*” on Target but does not ask users for their preferences even though several choices exist.

request. Second, for valid and feasible queries, annotators examined screenshots, reasoning logs, and user queries to determine whether the agent’s reported “completed” state reflected actual task completion. They labeled each task as accurately completed, inaccurately completed, or incomplete. Finally, for inaccurately completed tasks, annotators marked cases in which the instruction left a user preference unspecified and recorded the exact step where clarification was needed.

The annotators labeled the first 100 queries together and reached Cohen’s kappa values of 0.96 for query validity, 0.94 for task completion, and 0.82 for ambiguity and user-input requirements. After they resolved disagreements and refined the criteria, the annotators divided and independently labeled the remaining queries.

3.3.2 Dataset Analysis

Valid and invalid user queries. Our dataset contained 616 valid user queries and 22 invalid queries. The invalid queries typically included greetings (e.g., “How are you doing today?”) or compliments (e.g., “You have done a great job! Thank you!”) directed toward the UI agent. After excluding the invalid cases, we analyzed the remaining queries to characterize request types, automation outcomes, and situations that required additional user input. The original paper reports additional dataset statistics and analyses [1].

Types of user queries and task automation. Among the valid queries, 12 queries (2%) did not directly request task automation but instead asked about specific UI capabilities. For instance, one participant asked on Google Slides, “Can I insert a video from my local computer onto the slide?”

Another participant queried Google Calendar capabilities by asking, “Is it possible to automatically include a Zoom meeting link instead of Google Meet in my calendar event?” The remaining 604 queries explicitly requested task automation. From the 604 automation-oriented queries, the annotators categorized 28 (5%) as impossible automation requests. These queries resulted in agent failure either because participants requested tasks on unrelated or incorrect websites (e.g., booking a hotel on Target’s website) or requested actions unsupported by the platform (e.g., generating images within Google Docs). Participants submitted the remaining 576 feasible automation queries across 40 different platforms, representing diverse UI categories. These platforms included e-commerce websites, travel booking services (e.g., flights, hotels, restaurants), calendar and scheduling applications, productivity tools (document and slide editors), communication tools (email and messaging platforms), social media (X, Reddit), online content services (video streaming, e-learning), and cloud storage (Google Drive, Dropbox). Within the 576 feasible queries, the agent reported reaching a “completed” state for 484 (84%) tasks. The remaining 92 (16%) tasks were “incomplete” due to challenges in navigating complex, multi-level interfaces, performing granular UI interactions (such as hovering or drag-and-drop), or satisfying complicated user constraints. In summary, participants sometimes issued commands that the agent could not execute because they referred to features they had not yet encountered. We view those “impossible” commands as genuine instances of users discovering previously unknown interface capabilities. We also received non-task-related inputs (such as congratulatory messages praising the agent), which we excluded from our performance analysis because they did not affect task execution.

States of task completion and necessity of user input. For queries marked as “completed” by the agent, the average query length was 10.3 words, and each involved an average of 4.5 automation steps with pauses occurring at step 3.1. Among these, only 189 tasks were correctly completed. These tasks mapped directly to one available UI option or interaction, which eliminated any ambiguity regarding user choices and preferences. The remaining 295 queries were inaccurately labeled as completed (i.e., “inaccurately completed”). Out of these inaccurately completed queries, 113 involved clear execution failures. For instance, the agent mistakenly indicated task completion after adding the cheapest item to the cart without first sorting the results. The other 182 inaccurately completed queries involved ambiguities due to incomplete user instructions or unspecified preferences. Specifically, 107 queries involved underspecified user commands. In these scenarios, essential UI fields either remained empty or kept default values because users did not specify their preferences clearly. Examples include cases where users indicated departure and arrival airports but left out critical information such as travel dates, ticket types (round-trip or one-way), or travel class (economy or business). Additionally, underspecified queries included ambiguous constraints, such as requesting the “best chocolate” on the website without defining criteria for “best”. The remaining 75 inaccurately completed queries occurred when multiple valid options or UI actions matched the user’s query (e.g., the example presented in Figure 3.1), but users did not explicitly specify their preference in the original commands. For example, the agent faced multiple flavored waters with identical lowest prices when tasked with selecting the cheapest one, or encountered several available layout styles for adding page numbers when the user had not indicated a preferred style.

3.3.3 Follow-up Interviews

We interviewed every participant from the field study. For each participant, we selected six tasks (three that involved ambiguous choices and three that did not) from two applications the participant tried to automate. Participants compared the UI agent with a screen reader alone. They reported persistent navigation difficulties with traditional screen-reader workflows and agreed that the agent improved task efficiency. However, they also identified gaps for existing agents, especially when ambiguous choices arose and the agent failed to ask clarifying questions.

How Do UI Agents Improve Task Execution? None of the participants had used a UI agent before the study. After hands-on experience, every participant stated that language-based automation boosted both accessibility and efficiency. P2 remarked, “*The agent automates repetitive operations and lets me decide what I want; it handles the rest.*” Participants completed 40% of the tasks with the agent versus 25% with only a screen reader and worked roughly five times faster (42s per task versus 217s). They valued the agent’s ability to process complex or repetitive actions, which allowed them to focus on key decisions.

What Challenges Remain? During the study, the agent often acted without clarifying user intent. Interviews revealed that in 95% of these situations, participants never realized that multiple valid options existed. P4 noted, “*Without this interview I wouldn’t know there were several choices at the same price. I need prompts that surface differences and invite my input.*” Participants also struggled to specify complex preferences and to monitor progress. P3 said, “*When many fields appear, I lose track of the required information. A clear scaffold that lists pending items and allows review would help.*” Participants further highlighted limited real-time feedback. The visual action-history log offered some traceability, yet it seldom conveyed success or failure accurately. Several tasks appeared complete even when the agent had failed. Participants asked for audible or textual cues during execution and explicit confirmation at the end. Finally, participants wanted guidance on manual task completion to ensure independence when automation falls short.

Design Opportunities for Accessible UI Agents. Insights from the interviews suggest five design directions that address the reported challenges and keep BLV users in control:

- **D1: Active choice.** Pause at decision points and describe key differences among alternatives. For example, list the cheapest items side by side so users can select before the agent proceeds.
- **D2: Clear preference input.** Provide structured widgets such as drop-downs or number pickers that capture details text commands might miss, so users can refine a request without typing the full query again.
- **D3: Real-time feedback.** Announce ongoing actions through audio or concise status text, rather than wait until completion, so users remain oriented during lengthy or multi-step tasks.
- **D4: Result verification.** Prompt users to confirm outcomes, such as “Page numbers added. Accept or undo?”, and offer quick links for inspection to prevent silent failures.
- **D5: Task literacy.** Explain the UI’s available actions and outline manual workflows so users can complete tasks with a screen reader when automation tools are unavailable.

3.4 Morae

Informed by field studies and interviews, we developed Morae, a UI agent supporting BLV users' active engagement in UI automation. Morae addresses identified needs by providing: (i) dynamic ambiguity verification for user clarification of choices (D1), (ii) interactive preference input methods (D2), (iii) in-situ feedback on agent actions and outcomes (D3, D4), and (iv) a query mechanism for available actions and screen reader task guidance (D5). To build Morae, we extended TaxyAI, the UI agent used in the field study. TaxyAI runs as a Chrome extension, requires few system dependencies, and is straightforward to deploy. At each interaction step, Morae captures the UI state as a reduced DOM tree and a screenshot. GPT-4o then translates the user's command, current observations, and action history into executable actions.

3.4.1 Dynamic Verification of Ambiguous Choices

We aimed to ensure that Morae pauses at appropriate times. To identify suitable pause conditions, we analyzed our validation dataset. The validation set contains 54 of the 182 tasks (30%) annotated as cases of "unclear user choice or preference." From these examples, we developed an ambiguity-aware framework that determines when to pause through a self-ask-then-answer verification strategy. The complete prompts and model parameters appear in the original paper [1].

At each interaction step i , given user command Q , observed UI state $V(i)$, and executed action history $H(i)$, the agent follows three stages to determine the next actions:

Stage 1: Critical Actions. Our agent aims to balance autonomous task execution with timely intervention. We define *critical actions* as actions that either involve a user preference or reveal UI details required for a meaningful ambiguity check. Actions at planned step i are divided into critical ($P_c(i)$) and non-critical ($P_n(i)$):

$$P(i) = P_c(i) \cup P_n(i), \quad P_c(i) \cap P_n(i) = \emptyset.$$

The agent prioritizes critical actions so that it can gather enough information before it decides whether to pause.

Stage 2: Ambiguity Verification. At each step, the agent formulates prioritized questions from the command Q , observed UI state $V(i)$, and action history $H(i)$. The questions test two common cases: *selection ambiguity*, in which several UI options or actions satisfy the user's criteria, and *specification ambiguity*, in which the command omits required details or uses an unclear criterion. The agent answers each verification question with one of four responses (yes, no, unanswerable and proceed, not important and proceed). An ambiguity indicator $A(i)$ is defined:

$$A(i) = \begin{cases} 1, & \text{if any question returns "yes",} \\ 0, & \text{otherwise.} \end{cases}$$

The agent also evaluates whether the observed UI details are sufficient for an informed choice, represented by indicator $I(i)$:

$$I(i) = \begin{cases} 1, & \text{if agent's recorded details enable informed user decisions,} \\ 0, & \text{otherwise.} \end{cases}$$

Stage 3: Decision Function. Finally, the decision function $D(i)$ combines task execution and ambiguity verification, explicitly determining preference-elicited pauses:

$$D(i) = \begin{cases} \text{Execute critical actions,} & \text{if critical actions remain,} \\ \text{Pause for clarification,} & \text{if } A(i) = 1, I(i) = 1, \\ \text{Gather more UI details,} & \text{if } A(i) = 1, I(i) = 0, \\ \text{Proceed with next planned actions,} & \text{if } A(i) = 0. \end{cases}$$

Our algorithmic approach balances effective task execution with the need to detect and clarify unclear user choices or preferences. The agent dynamically verifies whether pausing is necessary at each interaction step. Our prompt design strategy builds general templates from field-study observations. The templates adapt to varied scenarios and require only a few task-specific changes. The approach ensures broad applicability and enables straightforward reuse across diverse tasks.

3.4.2 Generative UI for Capturing User Choices

When *Morae* detects ambiguity in user choices during task automation, the agent proactively pauses the process and prompts users to clarify their preferences. Users can respond by text, but managing multiple decisions simultaneously can be challenging. For example, if a user asks the agent to find the fastest flight between two cities but omits travel dates, ticket type (one-way or round-trip), and travel class (economy or business), the agent pauses. It explicitly lists missing details and informs users of preset defaults. To simplify preference specification, the agent generates a UI that displays the available options and lets users state their preferences (Figure 3.3). To achieve dynamic UI generation, the agent first identifies required decision properties and fields by analyzing the user's command Q alongside the current UI state $V(i)$. Based on the identified elements, the agent then creates appropriate UI components on the agent interface, such as radio buttons for single selections or text fields for numeric or text inputs (e.g., meeting time or topics). Each decision aspect is clearly labeled with proper header levels for screen reader navigation. Users may interact with the generated controls or respond in text. Their input determines the agent's subsequent actions.

3.4.3 In-situ Feedback for Screen Reader Users

To support active decision-making, we incorporate multiple forms of in-situ feedback that help BLV users remain involved throughout UI automation. Drawing from feedback collected in our prior studies, the agent provides auditory feedback that corresponds to its actions and states. A clicking sound plays when the agent selects a button or link, while a typing sound marks

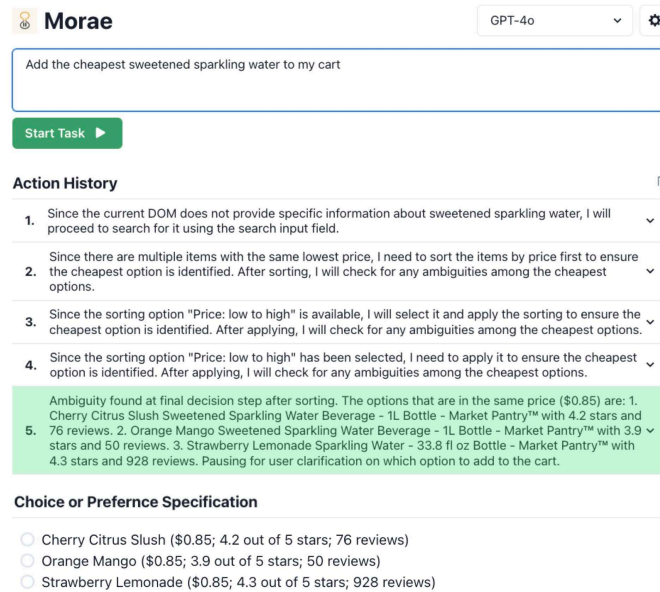


FIGURE 3.3: Morae’s interface features a task command input field, a start/pause automation button, an action history panel that displays the agent’s step-by-step reasoning, and dynamically generated UI elements for preference selection. In this scenario, Morae autonomously executes the command “*add the cheapest sweetened sparkling water to my cart*” on Target. At a decision point that needs user input, Morae pauses, presents relevant information, and offers accessible interactive controls so BLV users can state their choices in a structured manner.

text input. A distinct tone signals a request for clarification, and a confirmation sound marks a completed action. Users can also request visual verification after a task. GPT-4o drives the primary automation, while Gemini-2.0-Flash performs the supplementary visual check. The agent also supports blind users who want to explore and learn an interface. It classifies each query as either a question about the UI or a command for automation. We evaluated the classifier on interface questions from the field study and an equal number of randomly selected automation commands. The model distinguished the two categories with 96% accuracy. When responding to interface-specific questions (i.e., what tasks I can do with the UI and how to do specific tasks), the agent combines external knowledge encoded in the LMM with the current UI state observations. To provide personalized actionable guidance, we include the user’s screen reader choice gathered from the survey directly within the model’s prompt. For example, when a user asks, “*How do I find my recent emails in Gmail?*”, the agent provides detailed, screen-reader-specific instructions. The response can explain how to press NVDA + F7 to open the links list, select “Inbox,” and navigate messages with the arrow keys.

3.5 Technical Evaluation

We evaluated the core computational component of our agent, which determines whether and when the agent should pause to request user input to make choices or specify preferences. Our evaluation benchmarks our method against several ablations and OpenAI Operator. We conducted these comparisons on a subset of annotated tasks from our field study.

3.5.1 Methods

We constructed the test set from two groups of tasks across 20 different application platforms. The first group included tasks identified during our field study as having ambiguous user choices or unclear preferences. From this group, we selected 128 tasks (70%), sampling evenly from each of the eight application categories. The second group consisted of tasks that users marked as completed without any ambiguous preferences. Similarly, we randomly selected another 128 tasks from this group, ensuring equal representation from the same eight categories. We ensured the reproducibility of all selected tasks without needing access to personal user accounts or data. For instance, tasks conducted on platforms like Google Drive were recreated using our own accounts to generate identical interaction traces. This approach allowed us to accurately produce ground-truth labels and evaluate the agent in our online setup. For each task in the test set, we manually recorded and verified two interaction paths: (i) a complete sequence of steps required for successful task completion without pausing, and (ii) a sequence of steps leading up to the point when the agent paused to request user input. On average, the agent required 5.4 interaction steps per task, with pauses typically occurring at step 3.8.

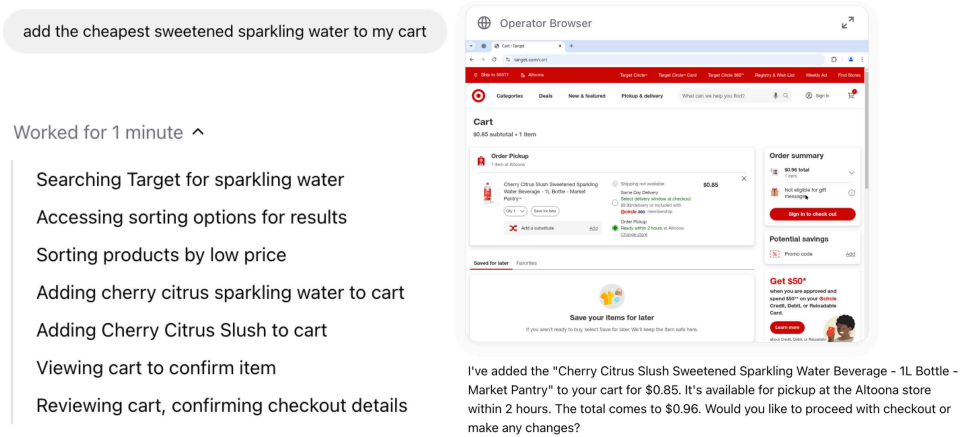


FIGURE 3.4: The UI of OpenAI’s Operator Agent. Users assign tasks via chat, monitor browser automation through live streaming and text updates, and manually intervene at any time. The example illustrates Operator autonomously executing the command “add the cheapest sweetened sparkling water to my cart” on Target without pausing to ask for the user’s choice.

Unlike conventional offline evaluations, which statically assess agents based on fixed queries, UI states, and interaction histories, we employed an online evaluation approach [180, 181]. Our online evaluation captures the interaction between execution and pausing decisions, stochastic model behavior, and variation in the observed UI state. We evaluated each agent condition based on two performance criteria: (i) **Task success rate**: The agent must either fully execute all annotated ground-truth steps for tasks requiring no pauses or correctly pause exactly at the annotated step for tasks needing user input. (ii) **Pausing performance**: We categorized agent pausing decisions at the task level into four possible outcomes:

- **True Positive (TP)**: The agent correctly pauses exactly at the annotated step in tasks requiring a pause.

- **False Positive (FP):** The agent pauses incorrectly or unnecessarily. False positives occur when the agent pauses prematurely (before the annotated step) in pause-required tasks, or at any step in tasks requiring no pause.
- **False Negative (FN):** The agent fails to pause at the explicitly annotated pause step. This scenario includes cases when the agent continues without any pausing or pauses at a later step after missing the correct annotated step.
- **True Negative (TN):** The agent successfully completes all annotated steps without pausing in tasks requiring no pauses. We exclude tasks from TN calculations if the agent neither pauses nor successfully completes the task.

We compared several variants of our method as baselines. Every variant, including our method, was built on TaxyAI and used GPT-4o as its LMM backbone. Their primary differences were the instructions or mechanisms used to trigger pauses; the original paper lists the full prompts. Additionally, we included OpenAI Operator as a strong baseline. OpenAI Operator [177] was explicitly trained for UI operations and optimized for user interactivity and safety. The methods we compared were:

- **Prompting:** The agent received explicit instruction to pause whenever user choices or preferences appeared unclear, supplemented by three descriptive examples.
- **Verifying-First-Step:** The agent generated ambiguity verification questions at the start of task execution. Questions were sampled three times, and the system retained the top five as the verification set. Questions relied solely on the initial user query and UI observation and remained constant throughout task execution steps.
- **Verifying-Per-Step:** At every execution step, the agent generated new ambiguity verification questions based on the current user query, observed UI states, and prior action history. These verification questions differed at each step.
- **Verifying-Per-Step-with-Planning (Our Method):** At each step, our method dynamically generated ambiguity-verification questions from the user’s query, observed UI, and prior actions. The agent also considered task progress so that it could request input when a preference appeared unclear without pausing too early.
- **OpenAI Operator** [177]: A state-of-the-art UI agent [181], fine-tuned explicitly on UI operations. We used the official client interface (Figure 3.4) to execute all tasks.

After each run, we closed and reopened the browser before the next task. Because the agent remained nondeterministic even at a model temperature of zero, we ran every task three times per method and compared each outcome with the ground-truth annotation. We report the mean performance across runs.

3.5.2 Results

Our approach achieved the highest average task success rate of 55.2%, outperforming Operator by 2.1 percentage points. The main improvement came from tasks that required pauses,

Method	Task Success Rate			Pausing Performance		
	Pause Required	Pause Not Required	Overall	Precision	Recall	F1-score
Prompting	18.0%	33.1%	25.5%	60.0%	18.6%	28.4%
Verifying-First-Step	32.3%	25.0%	28.6%	30.2%	40.8%	34.8%
Verifying-Per-Step	41.1%	27.9%	34.5%	32.6%	57.9%	41.7%
OpenAI Operator	50.8%	55.5%	53.1%	58.7%	59.1%	59.0%
Verifying-Per-Step-with-Planning (Ours)	65.6%	44.8%	55.2%	59.7%	69.8%	64.4%

TABLE 3.1: Task completion rate and pausing performance for each method we benchmark in our technical evaluation.

where our approach increased success rates from 50.8% (Operator) to 65.6%. We also observed a significant recall improvement of 10.7 percentage points over Operator. The self-verification process primarily contributed to this gain by improving sensitivity to unclear user choices, such as distinguishing multiple calendar entries labeled “Tuesday” for scheduling a meeting. Compared to other variants of our method, our approach consistently demonstrated higher success rates across both pause-required and non-pause tasks. For tasks without pauses, the agent effectively planned and prioritized critical actions, continuously monitored task progress, and anticipated potential needs for user input. For tasks requiring pauses, the self-verification method more accurately determined the correct moment to request user input than simpler prompting methods. Additionally, our method balanced proactive detection of pauses with continuous task execution to avoid unnecessary pauses. Alternative variants relying only on verification at initial or intermediate stages frequently paused prematurely to resolve ambiguities without fulfilling task requirements or obtaining sufficient context for an informed user decision. While our approach still paused unnecessarily (precision errors) or missed opportunities to pause (recall errors), it demonstrated greater overall robustness than the other methods in automating tasks while identifying the pauses needed to clarify user preferences. The original paper describes additional failure cases [1].

3.6 User Evaluation

We conducted a user evaluation to assess the usability of Morae by comparing it with two baseline agents: TaxyAI with GPT-4o as its LMM backbone and OpenAI Operator. Each participant performed nine tasks across three applications with all three agents.

3.6.1 Method

Participants. We recruited 10 BLV participants aged between 28 and 55 (U1–U10; four women and six men) through online mailing lists. The evaluation involved a 2-hour remote Zoom session. Participants had diverse professional backgrounds including students, customer support specialists, accessibility consultants, and software engineers. All participants had extensive experience with screen readers, general web browsing, and AI-based accessibility tools such as ChatGPT, BeMyAI, and SeeingAI. None had previously participated in our formative study nor used the UI agent before. Participants received compensation of \$100 per hour, and the institution’s IRB approved the study.

Procedure. The study began with demographic data collection and an overview highlighting core features of each agent. Before the study, participants reviewed documentation for Morae, TaxyAI, and Operator and learned how to use each agent independently. Participants completed three tasks on each of three websites: (i) Target [145], where they added the highest-rated beer to the cart, found details about the latest video-game deal for a specified brand, and bought the best-selling shorts; (ii) Google Calendar [182], where they turned off a notification, added an event, and modified an existing event; and (iii) Google Docs [183], where they reviewed recent edits, inserted page numbers, and added a code block. All tasks were sampled from the test set we used in our technical evaluation to further validate the results when users are actually involved. We matched task complexity between websites to ensure balanced comparison across agents. Participants experienced each agent in a counterbalanced order.

After using each agent, participants rated their experiences subjectively on a 7-point Likert scale (7 = highly positive) for:

- **Satisfaction with Choices:** Satisfaction with choices made during the task.
- **Awareness of Choices:** Awareness of available choices provided by the agent.
- **Control over Choices:** Sense of active control during choice-making.
- **Ease of Choice-making:** Ease of providing input when making choices.
- **Awareness of Actions:** Awareness of actions performed by the agent.
- **Awareness of Results:** Awareness of the outcomes resulting from agent actions.
- **Perceived Usefulness:** Overall perception of the agent’s usefulness for task automation.
- **Confidence in Use:** Confidence in independently using the agent for task automation.

In our study, we measured user satisfaction with the task outcome through a composite of related metrics: “Satisfaction with Choices”, “Control over Choices”, and “Perceived Usefulness”. We concluded with follow-up interviews about participants’ interaction strategies and their perceptions of each agent’s strengths and limitations.

Analysis. All sessions were recorded, and qualitative data from interviews and user comments were transcribed. We categorized qualitative feedback into two main areas: (i) strategies participants adopted during interactions, and (ii) perceived benefits and limitations of each agent. Subjective Likert ratings across three conditions (TaxyAI, Morae, Operator) were analyzed using Kruskal-Wallis H tests, followed by pairwise Mann-Whitney U tests with Bonferroni correction. Objective measures, including task completion time, task success rate, and number of decisions aligned with user preferences, were compared using repeated-measures ANOVA with paired post-hoc t-tests. To quantify the *diversity* of user decisions, we computed decision entropy D_e , which measures uncertainty or randomness in users’ choices. For a task with up to N options, entropy was defined as:

$$D_e = - \sum_{i=1}^N p(x_i) \log(p(x_i))$$

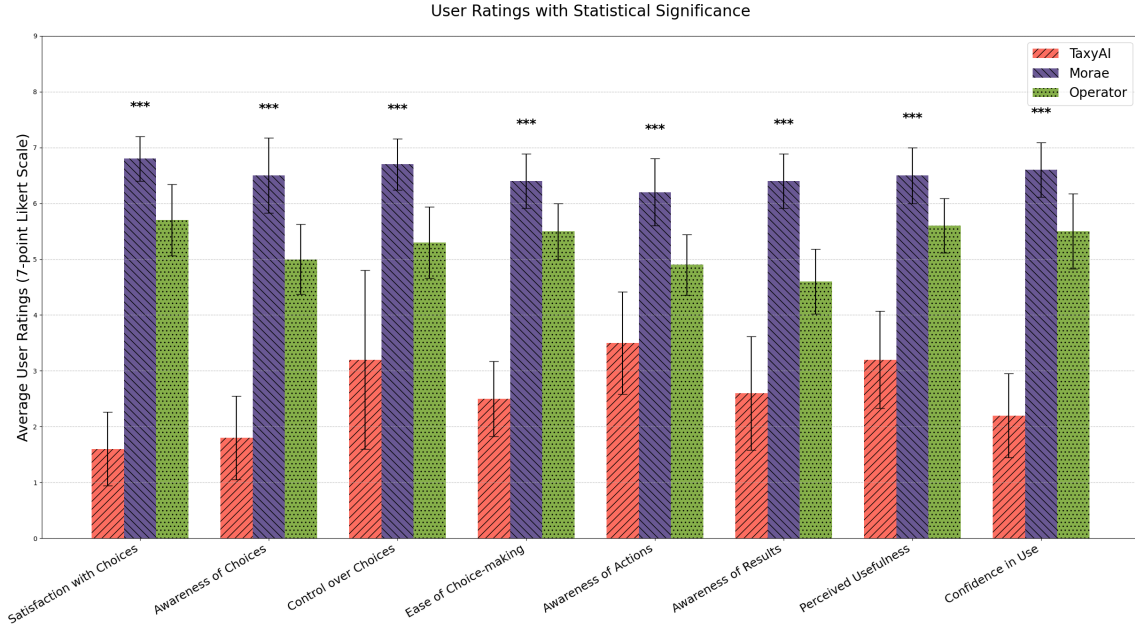


FIGURE 3.5: The average ratings for each of three agent conditions (TaxyAI, Morae, Operator) across eight subjective evaluation criteria (Satisfaction with Choices, Awareness of Choices, Control over Choices, Ease of Choice-making, Awareness of Actions, Awareness of Results, Perceived Usefulness, Confidence in Use) for our user evaluation.

where $p(x_i)$ represents the probability of selecting option x_i . Higher entropy indicates greater diversity and autonomy in selections, while lower values suggest uniformity.

3.6.2 Results

All participants (U1–U10) preferred Morae to TaxyAI and Operator. They described Morae as more accessible and easier to use, and they valued the additional opportunities to express preferences. Morae received significantly higher usefulness ratings ($\mu = 6.50$, $\sigma = 0.50$) compared to TaxyAI ($\mu = 3.20$, $\sigma = 0.87$; $U = 0.0$, $p < 0.001$) and Operator ($\mu = 5.60$, $\sigma = 0.49$; $U = 55.0$, $p = 0.015$). Participants reported significantly greater confidence in using Morae independently ($\mu = 6.60$, $\sigma = 0.49$) versus TaxyAI ($\mu = 2.20$, $\sigma = 0.75$; $U = 0.0$, $p < 0.001$) and Operator ($\mu = 5.50$, $\sigma = 0.67$; $U = 64.0$, $p < 0.001$).

Active User Involvement in Decision-Making. Participants spent significantly more time completing tasks with Morae ($\mu = 129.40$ sec, $\sigma = 13.75$) compared to TaxyAI ($\mu = 55.70$ sec, $\sigma = 8.51$; $t(9) = 12.56$, $p < 0.01$) and Operator ($\mu = 86.60$ sec, $\sigma = 9.11$; $t(9) = 10.87$, $p < 0.01$). Morae required more time because it paused to present alternatives and solicit informed choices when several options were available. Correspondingly, participants made significantly more preference-aligned choices on average with Morae ($\mu = 4.03$, $\sigma = 0.75$) compared to Operator ($\mu = 2.98$, $\sigma = 0.69$; $t(9) = 3.45$, $p < 0.01$) and TaxyAI ($\mu = 1.92$, $\sigma = 0.84$; $t(9) = 6.87$, $p < 0.001$). Decision-making entropy was also notably higher with Morae ($D_e = 1.58$) compared to Operator ($D_e = 0.86$) and TaxyAI ($D_e = 0.22$), indicating greater diversity and autonomy in participant choices. Participants also successfully completed significantly more tasks on average using Morae ($\mu = 5.50$, $\sigma = 0.71$) compared to Operator ($\mu = 3.90$, $\sigma = 0.57$; $t(9) = 5.38$, $p < 0.001$) and

TaxyAI ($\mu = 2.60$, $\sigma = 0.52$; $t(9) = 10.19$, $p < 0.001$). Participants consistently rated Morae significantly higher on critical aspects related to choice-making, including *Satisfaction with Choices* ($\mu = 6.80$, $\sigma = 0.40$), *Awareness of Choices* ($\mu = 6.50$, $\sigma = 0.67$), *Control over Choices* ($\mu = 6.70$, $\sigma = 0.46$), and *Ease of Choice-making* ($\mu = 6.40$, $\sigma = 0.49$), compared to both TaxyAI and Operator (all pairwise comparisons $p < 0.01$). Qualitative feedback further reinforced these findings: participants highlighted Morae’s ability to proactively pause and give users opportunities to express preferences. For instance, U4 stated, *“When selecting the best-rated beer or identifying product details at Target, Morae clearly described all available choices and allowed me to decide independently, unlike Operator or TaxyAI, which made selections on my behalf without detailed explanations and disclosure on potential multiple choices.”* Similarly, U7 remarked, *“With Morae, managing event details on Google Calendar was straightforward; I was always aware of the available fields I am missing or should choose from. With the interactive UIs that scaffold my decision process, I could more easily choose what I wanted while be aware of all default values that were filled.”*

Despite all the benefits Morae provided, participants indicated that Morae could improve further by providing more interpretable information on the agent’s decision confidence. For instance, U3 expressed, *“I appreciate Morae proactively pausing to let me make decisions, but I wish the agent could share how confident it feels about its suggested options. A confidence score or similar cue would help me decide when to interrupt and explore the UI myself.”* Additionally, participants recommended implementing customizable pause mechanisms to recognize individual differences in comfort and confidence with automation. U6 explained, *“Personally, I am comfortable allowing the agent to proceed independently, but other BLV users might prefer more frequent pauses due to monitoring challenges. A feature letting users define their preferred level of intervention would significantly enhance the experience.”*

Better In-Situ Feedback for Agent Actions and Outcomes. Participants reported significantly higher *Awareness of Actions* and *Awareness of Results* with Morae (actions: $\mu = 6.20$, $\sigma = 0.60$; results: $\mu = 6.40$, $\sigma = 0.49$) compared to TaxyAI (actions: $\mu = 3.50$, $\sigma = 0.92$; results: $\mu = 2.60$, $\sigma = 1.02$; $U = 0.0$, $p < 0.001$) and Operator (actions: $\mu = 4.90$, $\sigma = 0.54$; results: $\mu = 4.60$, $\sigma = 0.58$; $U = 64.0$, $p < 0.001$). Morae provided clear audio feedback at each step and explicit task completion confirmations. This design notably improved participants’ confidence in recognizing successful interactions. Participant U5 explained, *“Morae clearly informed me through audio cues whenever I successfully modified events in Google Calendar and confirmed each step explicitly. With Operator, I often felt uncertain about whether my requested changes took effect.”* Although Operator demonstrated state-of-the-art agent capabilities, its reliance on remotely streamed visual screenshots limited accessibility. Participants reported that Operator felt straightforward to instruct initially but verifying task outcomes independently became difficult without local screen reader access. Consequently, participants needed greater trust in Operator’s remotely executed actions compared to Morae’s immediate and verifiable feedback. Participants also found Morae’s screen reader-specific guidance clearer and more instructive. U2 stated, *“When I asked agents about how to insert page numbers in Google Docs, Morae explicitly guided me through each step with potential shortcut, unlike TaxyAI or Operator, which left me guessing about what actions I should do if I did the task by my own.”* Participants recommended improvements for Morae, including customizable audio feedback (U9) and additional modalities such as haptic signals (U1), to further

enhance overall accessibility.

3.7 Discussion

3.7.1 Task Success Rate for Current UI Agents

Our technical and user evaluations demonstrate promising results. Specifically, our agent effectively balances autonomous task execution and active user engagement in decision-making. Nevertheless, the sample size of our studies remains limited due to the specialized nature of our target user group and practical constraints during data collection. Larger studies could test the agent across a broader range of users and tasks and could supply training data for more reliable pause prediction. Rapid changes in open-source agents also make repeated evaluation valuable. In terms of task performance, current open-source alternatives and OpenAI Operator achieve success rates comparable to those reported in prior studies [180, 181], ranging between 30% and 60%. Operator currently establishes the highest benchmark for fully automated interactions without user intervention. One promising path to stronger robustness involves autonomous exploration. Prior work [155, 184, 185] generates interaction hypotheses or leverages demonstrations to guide agents through deep menu hierarchies. Morae complements these techniques with a lightweight self-verification loop that predicts when user input is required. The pausing module achieves 59.7% precision and 69.8% recall against human annotations, covering almost two-thirds of the 94% human upper bound while relying on no task-specific fine-tuning. Such recall already unlocks sizable gains in preference-critical scenarios; further exploration-based training should close the remaining gap and reduce unnecessary interruptions. While future improvements in accuracy for fully automated agents are likely, high accuracy alone is insufficient for practical adoption. Effective UI agents must actively involve users during crucial decision points, especially if user preferences are unclear. Our agent is the first to address the challenge by incorporating a self-verification loop within a balanced execution mechanism. The design explicitly prioritizes active user participation in UI automation tasks and sustains user engagement throughout the human-agent interaction.

3.7.2 Levels of Pause Necessity

Morae now relies on a binary pause policy such that the agent either halts or proceeds based on the user's command, the visible UI, and earlier actions. Our study revealed diverse preferences among blind and low-vision creators: some steps require an immediate stop, whereas routine operations can continue without interruption. These observations motivate a graded pause-necessity model that assigns a continuous score to each step, capturing both interface uncertainty and user-specific sensitivity. A calibrated threshold, learned from interaction data, can tune the score for every user and context. Payment dialogs, for instance, would receive a lower threshold than simple scrolling, triggering pauses more readily. The graded model naturally positions Morae within mixed-initiative control: the agent yields control when quantified uncertainty surpasses the threshold. At such moments, Morae presents ranked alternative actions side-by-side and invites the user to explain why one option best satisfies the goal. The contrastive prompt functions

as an implicit explanation, echoing contrastive techniques in explainable AI (XAI) [186–188] and unifying pausing, information disclosure, and user choice into a trust-building strategy. Future work will refine the calibration procedure so that *Morae*’s quantitative judgments align closely with individual preferences and the sensitivity of each task, thereby enhancing both personalization and automation effectiveness.

3.7.3 Multi-round User Preference Elicitation

Our dataset contains few cases of multi-round preference elicitation. In most annotated scenarios, once the agent resolves an ambiguity, the agent completes the task without further user clarifications. Such a pattern suggests users typically have simpler objectives in our dataset. However, real-world automation tasks often involve more complex user queries with multiple objectives. Multi-objective queries can introduce multiple ambiguous decision points during a long-horizon automation process. Each decision point may require continuous clarification and preference elicitation from users. Handling multi-objective tasks creates significant challenges. Agents need to balance immediate task execution and adapt to evolving user preferences or changes in the web context. Future research should study more interactive scenarios for conversational web navigation [161], including protocols that explicitly incorporate pauses by both the user and the agent. By actively managing interaction pauses, agents can better adapt to user preferences, handle complex objectives, and ensure coherent task automation.

3.7.4 Extending the Interaction Scope of *Morae*

Currently, *Morae* primarily operates as a web-based tool to enable broad deployment and rapid iteration across online applications. However, extending the agent’s capabilities to desktop and mobile platforms would significantly broaden its utility. Future work can integrate more precise visual recognition models [2, 70, 189] alongside existing LMM models. Recent work [74] has demonstrated that granular visual recognition can improve agents’ semantic understanding of user interfaces. Such visual models directly extract UI structures from pixel-level data and provide metadata similar to DOM trees, including detailed element positions and groupings. Leveraging granular visual parsing will enhance the agent’s capability to reason accurately about visual semantics and automate complex tasks effectively across multiple software environments. Additionally, expanding the agent’s native compatibility with various operating systems would support research into users’ long-term usage patterns and interactions with diverse digital tools. On the interaction side, *Morae* currently supports basic operations such as clicks and text entry. Future development should broaden the set of interaction methods. Direct calls to native UI control APIs provided by major operating systems [190–192] will unlock richer interactions and enable proactive agents to tackle more complex workflows in video editing [78, 79, 193], graphic-design creation [5, 6, 68, 107], map navigation [194–196] and beyond. Native integration also delivers deeper functionality and finer control across web, mobile, and desktop platforms. Although initially developed to enhance accessibility for blind and low-vision (BLV) users, expanding *Morae* to support additional disabilities requires further customization. Future development should include adaptations to feedback mechanisms and interface controls. For instance, simplifying text-based

feedback can better support individuals with cognitive disabilities, while integrating customizable UI controls can better accommodate users with motor impairments. Future enhancements can also focus on compatibility with existing assistive technologies, increasing accessibility for a broader range of users.

3.7.5 Human Agency During Automation

Our agent can act as a proxy for BLV users to interact with digital interfaces that are difficult or inaccessible to use. However, we strongly encourage developers to create user interfaces that are technically accessible as well as intuitive and genuinely usable, which would also benefit people who use assistive technologies such as screen readers. Our goal extends beyond offering basic accessibility overlays or automating simple tasks. We developed *Morae* in response to the accessibility barriers that users face in complex software. The agent actively supports users to express and apply unique choices and preferences throughout task execution. Instead of focusing solely on robust automation that completes tasks automatically, our approach emphasizes collaboration. Our findings indicate that the agent encourages users to actively explore available choices, express preferences clearly, and remain engaged and informed during the automation process. The agent proactively introduces pauses that allow BLV users to express their choices clearly, an opportunity that standard automation often neglects. The value of *Morae* extends beyond successful task completion. The agent lets users express meaningful choices, engage actively with applications, and learn from the automation process. Ultimately, our approach positions users as active collaborators who guide automation rather than passive observers.

3.7.6 Beyond Accessible Automation: Broader Agent-driven Applications

Our user study highlights the potential for UI agents to bridge language barriers. One participant (U5) successfully interacted with a website in Chinese by issuing commands in Polish: *“I don’t speak Chinese at all, but when I typed in Polish, the agent operated successfully and provided feedback in Polish! The agent not only made the website accessible non-visually but also bridged the language gap!”* Participants further recognized broader implications when agents interact with user interfaces. Another participant (U10) commented, *“Sometimes AI works like a screen reader. If you make an interface accessible to us [screen reader users], you are also likely making it easier for AI to navigate!”* Such insights highlight that agents can extend beyond accessibility applications. For example, agents may automate accessibility testing [197], perform usability analyses, and ensure digital interfaces support diverse human users and AI agents. Difficulties agents encounter when interacting with user interfaces often reflect similar usability challenges faced by humans. Therefore, improving interface designs for agent interactions can potentially enhance usability for human users.

3.8 Conclusion

We introduced *Morae*, an accessible UI agent that proactively pauses during task automation to allow blind and low-vision users to express their preferences and make choices. Driven by

insights from an in-depth field study and detailed interviews with four BLV individuals, Morae proactively identifies critical decision points requiring user input. Instead of assuming preferences, the agent pauses at these junctures and dynamically creates accessible interfaces to enable users to interactively specify their preferences. Morae uses large multimodal models to interpret commands, analyze UI structure, plan and execute tasks, detect ambiguity, and provide contextual information with in-situ feedback about actions and results. The technical evaluation and user study show that Morae gives users more opportunities to make informed, preference-aligned choices, increases the diversity of their decisions, and improves task completion. Morae therefore offers a foundation for agents that align automation with human judgment and preserve active user input in real-world interaction.

Chapter 4

DreamStruct: Understanding Slides and UIs via Synthetic Data Generation

A meaningful handoff begins with a faithful account of the interface.

DreamStruct uses executable code to generate slides and interfaces together with the structure needed to train multimodal models. The chapter then extends from one screen to longer activity: GUIDE asks what the user is doing and whether help would be useful, while TEMPURA develops temporal representations that place events and transitions in context.

Morae could pause at the right moments only because its models recognized that several interface options satisfied the user’s instruction. An agent cannot return a choice, describe an alternative, or explain a result that its models never registered, so the interaction mechanisms in this dissertation rest on how well models parse structured visual content. Building such models has typically required collecting and annotating data separately for each task and visual domain, which limits scale. DreamStruct instead generates structured visuals as code and embeds target labels in the generated artifacts. The resulting datasets support models trained with little human-annotated data. We evaluate the approach on three tasks: visual element recognition, screen description, and visual content classification.

4.1 Introduction

Computational models of visual structure can describe slides and user interfaces (UIs) for people who are blind [198–201], retarget layouts to new devices [202, 203], and personalize content according to user ability [192, 204–206]. However, building the underlying machine learning models that enable these capabilities requires labor-intensive data collection and annotation, which must be performed for each type of input.

We generate synthetic, structured visuals by producing and rendering code (Figure 4.1). The pipeline has three phases. First, a large language model (LLM) creates design concepts under task and design constraints. Second, the model expresses each concept in a labeled declarative language such as HTML. Third, the pipeline filters, post-processes, and renders the code to produce

This chapter is based on work published at the European Conference on Computer Vision (ECCV 2024) [2], with Faria Huq, Yue Jiang, Jason Wu, Xin Yue Li, Jeffrey P. Bigham, and Amy Pavel. Section 4.6 discusses coauthored work led by Saelyne Yang (GUIDE, CVPR 2026) [3] and Jen-Hao Cheng (TEMPURA) [4].

an annotated dataset. The method can apply to many structured visual domains; we evaluate presentation slides and UI screenshots because both lacked large, high-quality public datasets for the target tasks. In each domain, we study (i) element recognition, (ii) image captioning, and (iii) image classification. The pipeline produces synthetically annotated data on demand and supports later adaptation to real-world data. Compared to automated approaches such as crawling and metadata extraction [70, 207], our approach can apply to more visual domains (e.g., presentation slides) and produce more types of annotations (e.g., semantic descriptions, visual classification).

Using our method, we generate two synthetic datasets: one with 10,053 labeled slides (*DreamSlides*) and another with 9,774 labeled user interfaces (*DreamUI*). We trained machine learning models on the synthetic datasets and compared them with models fine-tuned on existing human-annotated data and models with strong zero-shot performance, including LLaVA [208] and Qwen-VL [209]. For element recognition, synthetic slide training improved mAP by 10.95 percentage points over training on an equal amount of human-annotated data. Pretraining on synthetic UIs improved mAP by 5.20 points after adaptation to real UI data. For image captioning, the synthetically fine-tuned models achieved average pairwise win rates of 68.9% for slides and 67.1% for UIs against the tested baselines. For image classification, the synthetically fine-tuned models surpassed counterparts trained on human-annotated data by 18.4 percentage points for slides and 11.3 points for UIs. The results show that generated code can provide useful supervision for structured visual semantics and representations.

4.2 Related Work

We review literature in three related areas: (i) understanding slides, (ii) understanding UIs, and (iii) synthetic data generation for representation learning.

4.2.1 Understanding Slides

Previous research developed human-annotated datasets and models for interpreting visual and textual content in presentation slides, enabling flexible presentation consumption and authoring experiences [5, 78, 79, 202, 210]. The SpaSe [76] and WiSe [211] datasets were early resources that provided manually curated object segmentation masks for 2,000 slides from a large online presentation platform [212]. Further advancements in slide modeling have involved annotating bounding boxes for elements on more than 50K slides, facilitating applications such as visual question answering [213] and sketch-based slide retrieval [214]. In addition to static slide documents, the Lecture Presentation Multimodal Dataset [215] includes annotations for approximately 8.6K graphic elements across 9K slide frames derived from lecture videos. The FitVid dataset [202] extends the data annotations by labeling both individual-level and semantic-group-level elements (e.g., headers, footers, bullet text boxes), totaling 26K element annotations across 5.5K slides. Unlike earlier methods that relied on human annotators, our approach generates synthetic slides with relevant metadata on demand, minimizing the need for manual annotations.

4.2.2 Understanding UIs

The introduction of large-scale UI datasets by prior works [69, 100, 189, 216] enabled the development of data-driven approaches for computational UI modeling – including UI element recognition and semantic grouping [189, 217–221], assessments of interactivity of UI elements [207, 222, 223], and evaluating agentic workflows for UIs [30, 31, 224]. However, collecting such large-scale datasets is often time-consuming and labor-intensive. Unlike these methods that rely on manual collection and human annotation for real UI data, our method uses synthesized UIs as training data, allowing people to construct datasets flexibly while reducing time and manual effort. We assess models trained on DreamStruct with existing UI benchmarks for semantic element and group recognition [100], screen captioning [225], and screen design-pattern classification [226].

4.2.3 Synthetic Data Generation for Representation Learning

The concept of using synthetic data to learn real-world distributions has been applied across various domains including robotics (sim-to-real) [80, 227, 228] and healthcare [229, 230]. Recent research has increasingly used generative models to create synthetic data for visual and language representation learning [81, 231–236]. One particularly relevant work is WebSight [237], which presents a synthetic dataset of approximately 820K HTML code examples generated from screen descriptions. Similar methods have been employed to generate synthetic data for other modalities, such as vector graphics [105, 238], charts [239, 240], animation [241], and 3D models [242, 243]. While these papers generate code to synthesize visuals, their objectives are fundamentally different from ours. They focus on translating between pixels and code abstractions (pix2code and code2pix) without explicitly generating task-specific underlying structures, which limits their applicability in understanding structured visuals (slides and UIs). In contrast, our method generates task-specific metadata alongside code, providing a fine-grained training source for structured visual representation learning.

4.3 Methods

Instead of relying on traditional methods for building visual understanding datasets, which require human annotation of inputs (e.g., screenshots), DreamStruct begins with an abstract specification of the desired annotations and constraints, then generates the corresponding visual inputs as renderable code. In this section, we describe our methodology for creating synthetically labeled slides and UIs. We first explain our generation principles, followed by the generation pipeline and the analysis of our synthetic datasets.

4.3.1 Generation Principles

To make the generated visuals useful for models evaluated on in-the-wild data, we define explicit *design principles* and *generation goals*. The rules make the desired output concrete and support more precise prompt instructions [244]. We use three principles (*P*):

- P1. Consistency with existing visual design guidelines:** The generated design and layout should follow established practitioner guidelines. For example, UI controls should measure

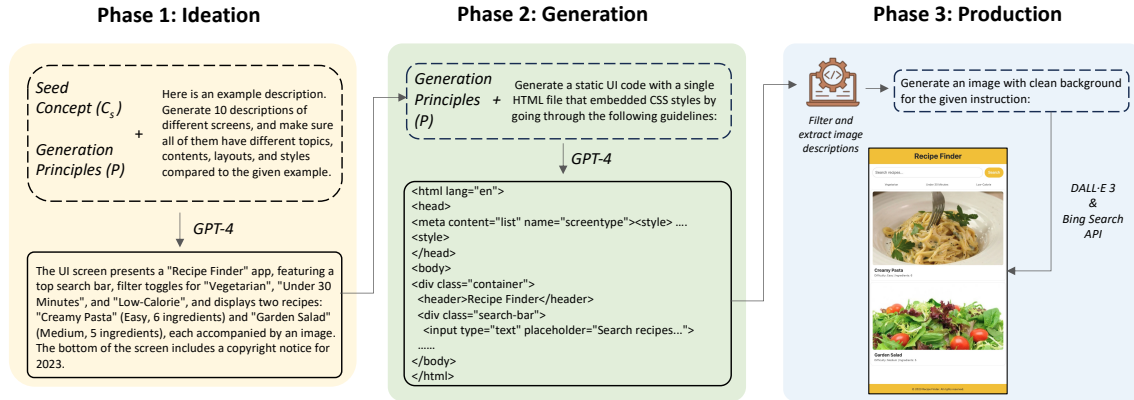


FIGURE 4.1: Overview of the DreamStruct pipeline. From left to right: In the *Ideation Phase*, screen descriptions for slides and mobile UIs are generated. In the *Generation Phase*, corresponding code is produced based on these descriptions, guided by the design principles and targeted labels. Finally, in the *Production Phase*, visual placeholders are replaced with actual data to render the final slides and UIs.

at least 44×44 pt [245], and the main text on a slide should use a font size larger than 24 pt [246].

- P2. Consistency with existing visual design patterns:** The generated visual design should reflect common design patterns (e.g., established themes or layouts). For example, the login and setting screens are common classes for mobile UI design [226, 247], and the title slides and section slides are common patterns for presentation design [248–250].
- P3. Consistency with target data labels:** The generated metadata should align and remain consistent with all specified data labels for slides and UI benchmarks. The principle applies to both element-wise [100, 202] and screen-wise semantics [76, 202, 226]. For instance, an interactive swipe feature in a UI should be categorized as “page indicator.”

4.3.2 Generation Pipeline

Our generation pipeline consists of three steps (Figure 4.1): (1) *Ideation*: Create design concepts for slides and UI. (2) *Generation*: Generate code with target labels using design concept descriptions. (3) *Production*: Filter, post-process, and render the code to produce the complete slides and UIs.

4.3.2.1 Phase 1: Ideation.

We generate a corpus of design concepts (C_d) for slides and UIs by prompting an LLM with our generation principles (P) and a set of seed design concept examples (C_s). To obtain our seed design concept examples (C_s), we examine real-world instances of slides and UIs [202, 215, 226]. From the examination, we select 20 diverse samples for both slides and UIs. We then manually write descriptions for all the samples based on prior visual description guidelines [5, 251]. Each human-authored description summarizes the visual content, style, and layout and therefore serves as a design concept. For example, *The slide titled “By the Numbers” presents three key statistics on disability representation, each in a separate column with a corresponding icon. The statistics*

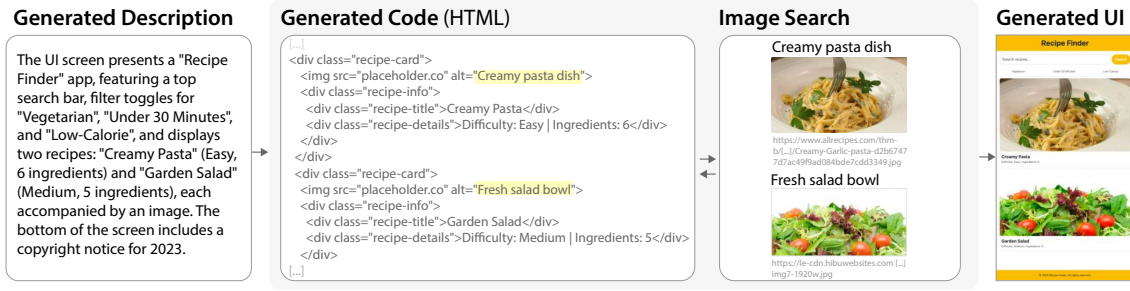


FIGURE 4.3: An example of a generated UI description with the corresponding generated code and generated UI. In the *Production* phase, our pipeline extracts alternative text and placeholder image links and uses Bing Search API or DALL·E to get a link or image to replace the placeholder.

pipeline uses `chart.js` [254], `chartist.js` [255], and `chart.css` [256]. At this step, we have an updated corpus of pairwise descriptions and annotated code that both follow our principles, $C_h = \{\mathcal{G}_{t=0.3}(c_d; P), \forall c_d \in C_d\}$.

4.3.2.3 Phase 3: Production.

The final phase post-processes, filters, and renders the generated code. The alternative text attached to each graphical placeholder serves as the query or prompt for its replacement. For simple elements such as images and icons, the pipeline queries the Bing Search API and favors sources with transparent backgrounds. For complex elements such as diagrams, the pipeline prompts DALL·E. It then downloads each result and replaces the placeholder source. Beyond rendering graphical elements, the pipeline applies three heuristic post-processing steps: (1) remove background fill colors when a background image is present; (2) add explicit width and height values to `<img>` elements so that assets fit their containers; and (3) keep sliding menus open on page load so that their contents remain visible. To reduce the low-quality samples caused by major element occlusion or invalid `<img>` references, we compute the CLIP score between the rendered screens and their design concepts, and we filter out examples with a CLIP score below 0.3. We choose the threshold based on the average image-text alignment for LLM-based text-to-code generation [257]. In total, we filtered out 215 (2.1%) slide and 336 (3.36%) UI samples. The final HTML is rendered within viewports of 1280×720 for slides and 628×1118 pixels for UIs. Our updated corpus can be represented as:

$C_{dhm} = \{\mathcal{R}(\mathcal{F}(c_h)) \mid \forall c_h \in C_h \text{ and } \forall c_d \in C_d, \text{CLIP}(\mathcal{R}(\mathcal{F}(c_h)), c_d) > 0.3\}$, where $\mathcal{R}$ and $\mathcal{F}$ are the rendering and filtering functions.

	Slides	UIs
# of Descriptions	10,268	10,000
# of Generated Code (after filtering)	10,053	9,774
# of Generated Images and Icons	15,593	30,518
# of Generated Charts	2128	-
Avg. # of Elements	6.46	11.08

TABLE 4.1: Statistics of DreamStruct dataset.

4.3.3 Synthetic Dataset

Table 4.1 presents the statistics for the data generated in DreamStruct¹. We created a total of 10,053 slide-code pairs (DreamSlides) and 9,774 UI-code pairs (DreamUI). We quantitatively compare DreamStruct with human-annotated datasets in terms of descriptions and element compositions. DreamStruct provides significantly more detailed descriptions for slides and UIs, averaging 57.23 words and 19.66 named entities per description, compared to 5.74 words and 3.37 named entities in human-annotated datasets. The differences in element compositions are marginal, including the number of images per sample (DreamStruct=2.32, human=2.48), charts per sample (DreamStruct=0.21, human=0.07), diagrams per sample (DreamStruct=0.24, human=0.12), and total elements per sample (DreamStruct=8.74, human=8.83). Figure 4.4 illustrates the distribution of elements in our synthetic slide and UI datasets compared to ground-truth datasets. Even without explicitly controlling the distribution of element classes during generation, the element distribution in our synthetic data aligns with real-world datasets [100, 202]. An exception is the ‘upper task bar’ in our synthetic UI dataset, which displays the time and internet signals and is not typically part of the UI content. Consequently, the upper task bar appears infrequently in our dataset, as shown in Figure 4.4b.

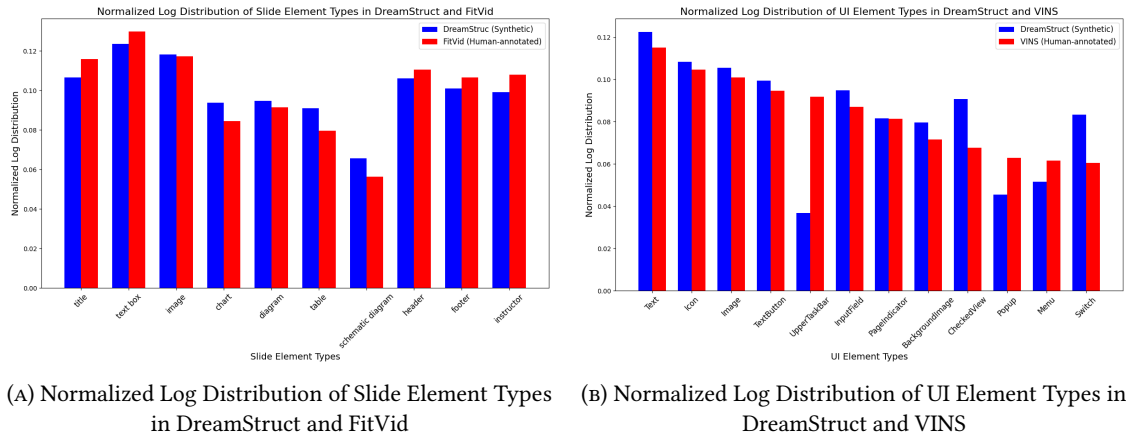


FIGURE 4.4: Normalized Log Distribution of Slide and UI Elements

4.4 Experiments

We evaluate DreamStruct by training or pretraining models on the generated slide and UI datasets and comparing them with strong baselines trained on human-annotated data or released for zero-shot use. The chosen models are designed to perform element recognition, image captioning, and image classification on screenshots, which are essential tasks for the computational understanding of slides and UIs.

4.4.1 Element Recognition

Element recognition identifies and outlines elements in slides and UIs from pixel input. The recognition results can be applied to domains such as accessibility [189], design [100], and device

¹<https://github.com/yihaop/dreamstruct>

retargeting [202]. For our base object recognition models, we selected several vision backbones, including CenterNet2 [258], Deformable-DETR [259], and EfficientDet-D0 [260].

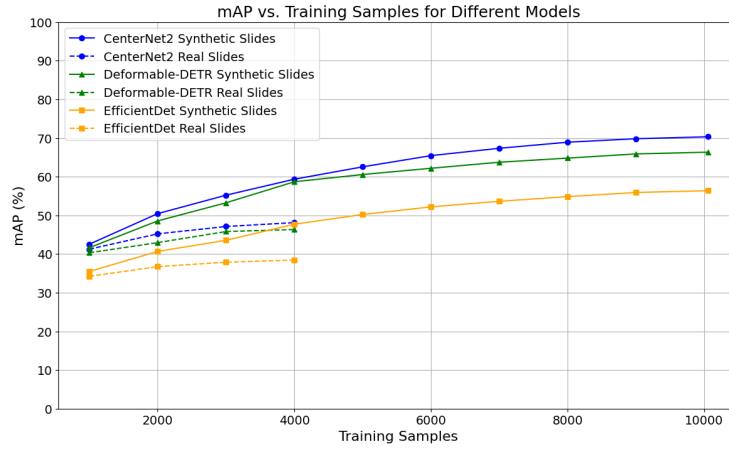


FIGURE 4.5: Performance of slide element recognition when adding training samples

4.4.1.1 Slide Element Recognition.

For slide element recognition, we use the FitVid lecture-design dataset [202] with 70% of the data for training, 15% for validation, and 15% for testing. Slides from the same presentation video remain in one split, which prevents near-duplicate frames from appearing in both training and testing. FitVid contains 5,500 slides from 14 domains and labels 12 semantic element types. We merge semantically close classes, such as “figure” and “picture” into “image” and “handwritten” into “text box,” which leaves 10 classes: title, text box, image, chart, diagram, table, schematic diagram, header, footer, and instructor. The dataset provides bounding boxes for both individual elements and container elements. Its “instructor” class is uncommon in related slide datasets [76, 211, 214, 215].

Our findings show that models trained on synthetic slides achieved higher mean Average Precision (mAP) than those trained on human-annotated data. Specifically, models trained on synthetic data achieved an average of 55.3% mAP compared to 44.3% mAP for models trained on an equivalent sample size of human-annotated data (full training set). The limited performance of models trained on human-annotated data was due to a lack of visual design variation, as many training samples were derived from the same video. In contrast, our synthetic data generation method ensures a diverse dataset, resulting in a more generalized data distribution. Figure 4.5 shows the performance curve as we gradually added more synthetic training samples. Performance gains plateaued and peaked at an average of 64.4% mAP when training each model with a synthetic dataset twice the size of the original human-annotated set. Our performance analysis of individual element types revealed that our models excelled in recognizing tables and charts due to the diverse data generated through code variations coupled with image retrieval and generation.

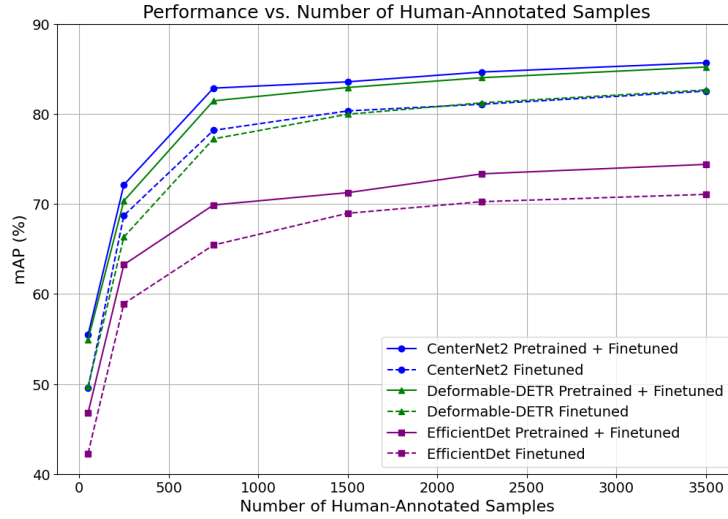


FIGURE 4.6: Performance of UI element recognition when adding training samples

4.4.1.2 UI Element Recognition.

For UI element recognition, we use the VINS mobile UI dataset [100] with the 70/15/15 training, validation, and test split from prior work [70]. The dataset contains 5K mobile UI screens along with 12 classes of elements (e.g., ‘text’, ‘image’, ‘text button’, ‘icon’, ‘input field’, ‘switch’, ‘checked view’, ‘background image’, ‘sliding menu’, ‘upper taskbar’, ‘page indicator’, ‘popup window’) across the screenshots of iOS and Android applications as well as UI mockups. The dataset provides bounding boxes for both individual elements (text, images, text buttons, icons, switches, checked views, background images, page indicators) and container-type elements (input fields, upper task bars, sliding menus, popup windows).

To address the element distribution gap between the synthetic and the real-world data, we adopted a two-stage training approach. Initially, we used our synthetic dataset to pretrain the models, which allowed us to establish a foundational set of weights. This pretraining step is crucial as it initializes the model’s weights with knowledge gained from the synthetic UI data, despite its distributional differences from real-world data. After pretraining, we then finetuned the models on human-annotated data. This finetuning phase adjusts the pretrained weights to align more closely with the real-world data distribution, thereby enhancing the model’s performance on tasks that reflect real-world scenarios. Our findings show that using synthetic UIs for pretraining, followed by finetuning with only 50 samples of human-annotated data, resulted in an average mAP improvement of 5.2% compared to training solely with the same set of human-annotated samples (from an average mAP of 47.2% to 52.4%). As we gradually added human-annotated samples, the pretraining step continued to show benefits, with an average mAP improvement from 78.8% to 81.8% when finetuning on the full human-annotated training set (Figure 4.6).

4.4.2 Image Captioning

Image captioning refers to the task of generating descriptions for the screenshots of slides and UIs. The descriptions can support blind and low-vision individuals in accessing information from visually structured images [5, 225, 261]. We evaluate the quality of descriptions generated

from four vision-language model (VLM) variations: our model that fine-tunes LLaVA-1.5-13B using the synthetic datasets; the original LLaVA model without fine-tuning; the vision language models including LLaVA or Pix2Struct fine-tuned with existing human-written captions [214, 225]; and GPT-4V (zero-shot). We conducted manual pairwise comparisons with six human annotators. All annotators first judged the same 50 slide and UI examples so that we could check preference consistency under procedures from prior caption-rating studies [251, 262]. Each rater then evaluated about 2,000 randomly sampled comparisons between two model outputs, for 10,392 judgments in total.

4.4.2.1 Slide Image Captioning.

For slide description, we randomly sampled 732 slides from existing lecture presentation slide datasets [202, 215] as our test set. Figure 4.7a presents the pairwise win rates between our methods and baseline conditions including LLaVA zero-shot, LLaVA fine-tuned with existing slide descriptions [214], and GPT-4V. The model fine-tuned with our synthetic image-description pairs surpasses those fine-tuned with current description datasets (77.9% win rate) and those derived from zero-shot inference (97.3% win rate). Our model, though smaller, still achieves a 31.6% win rate compared to GPT-4V.

4.4.2.2 UI Image Captioning.

For UI description, we randomly sampled 1,000 UIs from existing UI datasets [100] as our test set. Figure 4.7b presents the pairwise comparison results between our methods and baseline conditions (including LLaVA zero-shot, the vision-language model Pix2Struct [263] finetuned on existing screen2word dataset [225], and GPT-4V). The model fine-tuned with our synthetic image-description pairs surpasses those fine-tuned with existing description datasets (64.8% win rate) and those generated from zero-shot inference (99.2% win rate). Against GPT-4V, our model achieves a 37.3% win rate despite its smaller backbone.

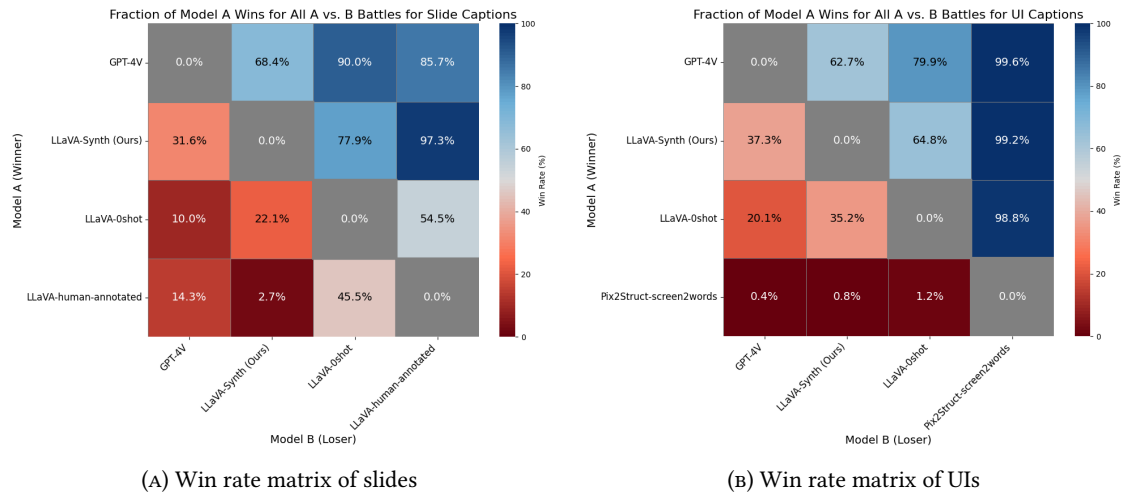


FIGURE 4.7: Pairwise win-rate matrices across captioning models

4.4.3 Image Classification

Image classification involves identifying the themes and designs of individual slides and UIs across subjects or visual patterns. By categorizing the themes, the machine can perform tasks like detecting lecture topics or chapters [210, 264, 265], and offering UI design recommendations [100, 226]. We chose two VLMs, LLaVA-1.5-13B [208] and Qwen-VL-Chat-7B [209], as our base classification models. We compared the classification performance across multiple conditions: base models fine-tuned on our synthetic datasets; the original base models under zero-shot inference; base models fine-tuned on existing human-annotated data; and two off-the-shelf VLMs: GPT-4V and Gemini-1.0-Pro. For the human-annotated dataset, we maintained an 80% training and 20% testing split for benchmarking. Notably, the current image classification datasets for either UIs or slides only comprise around 1,000 samples or fewer.

Model	Slides	UIs
GPT-4V	0.772	0.598
Gemini-1.0-Pro-Vision	0.700	0.533
LLaVA synth (ours)	0.686	0.545
Qwen-VL synth (ours)	0.672	0.558
LLaVA zero-shot	0.462	0.395
Qwen-VL zero-shot	0.473	0.382
LLaVA human-annotated	0.335	0.316
Qwen-VL human-annotated	0.341	0.303

TABLE 4.2: The classification accuracy of each model for both slides and UIs (training samples: synthetic $\approx$ 5K; human-annotated $\approx$ 0.5K).

4.4.3.1 Slide Classification.

For slide classification, we selected the top 10 topics from previous datasets [202, 215] and combined similar themes, resulting in six topics: psychology, communication, law, public health, computer science, and language learning. We initially sampled 960 slides, excluding irrelevant ones such as ‘Thank You’ slides, which led to a dataset of 732 slides as our ground truth set. From this human-annotated set, we randomly chose 25 slides per topic, creating a test sample of 150 slides. Table 4.2 displays the slide classification accuracy. Models trained on our synthetic slides achieved an average classification accuracy of 52.2%, surpassing the original base model (46.8%) and the base models fine-tuned with human-annotated data (33.8%). By expanding the training dataset to 5,033 synthetic slides, our method reached an average accuracy of 67.9%. However, further increases in sample size resulted in diminishing returns (Figure 4.8a). Despite this improvement, the optimal performance with the current synthetic sample sizes still lags behind GPT-4V by 9.3%.

4.4.3.2 UI Classification.

For UI classification, we chose the top-10 design topics from the previous mobile UI design dataset [226]. We merged the ‘News’ class into ‘Gallery’ because visual appearance did not



FIGURE 4.8: Performance of Slide and UI Image Classification when scaling up synthetic training samples

reliably distinguish the original label. This yielded categories including ‘List’, ‘Login’, ‘Settings’, ‘Menu’, ‘Media Player’, ‘Form’, ‘Profile’, ‘Tutorial’ and ‘Gallery’, resulting in a dataset of 1,020 unique UIs. For evaluation, we randomly selected 25 UIs per category from the dataset, forming a test set of 225 UIs. With a similar number of training samples, our synthetically-trained models demonstrated a 42.3% classification accuracy, outperforming the original base models (38.9%) and base models fine-tuned with human-annotated data (31.0%). With around a ten-fold increase in training samples to 5358 synthetic UIs, the accuracy rose to 55.2%, although further gains plateaued with additional training samples (Figure 4.8b). The current synthetic training sample sizes result in performance that is 4.7% below that of GPT-4V.

For both slide and UI classification, fine-tuning on human-annotated samples reduced performance relative to zero-shot inference. The available structured-visual datasets are small and class-imbalanced, which can limit fine-tuning, as shown in prior work [70, 266].

4.4.4 Summary

The results of our experiments reveal several key findings. First, in domains like slide understanding, where high-quality data is the primary limiting factor, DreamStruct achieves significant performance gains (+20% mAP). In cases like UI understanding, which benefit from larger quantities of human-annotated data, incorporating DreamStruct’s synthetic data still leads to substantial improvements in a fine-tuning setting (+5% mAP). Across all tasks and domains, training on DreamStruct’s synthetic data leads to performance gains and allows open-source models (*e.g.*, LLaVA, Qwen-VL) to approach the performance of larger proprietary baselines (*e.g.*, GPT, Gemini). The results show that code-oriented generation can supervise visual representations at several granularities.

4.5 Limitations

DreamStruct improves several downstream tasks for slide and UI understanding, but the current generation pipeline still has important limitations.

Quality of Code Generation. Although we specify guidelines for element positioning and spacing (§4.3.2.2) during our generation, the generated layouts sometimes fail to follow the specific guidelines, or the generated code adds the metadata into the wrong place. The errors can cause invalid CSS, overflow, and overlapping elements. For example, the generator placed the `data-type` attribute inside CSS rules 1,183 times across 354 screens, even though `data-type` is an HTML attribute rather than a CSS property. While we could fix these minor errors by parsing the HTML during post-processing (§4.3.2.3), future work should be mindful that such errors can occur in synthetic data-generation frameworks [244]. To improve the code generation quality, future work can incorporate additional domain-specific filtering models [6, 257] as reward functions to enhance the code-based visual generation results.

Limitation of Visual Content Generation. The current pipeline uses image alternative text to retrieve or generate visual content through Bing Image Search and DALL·E (§4.3.2.3). Because the retrieval or generation request lacks broader screen context, the resulting images may use inconsistent visual styles. Future work can extend the visual content generation process by including additional context [267] and visual asset libraries.

Potential Negative Impact of the Work. While our work makes it easier to improve understanding of structured visual content like slides and UIs, the DreamStruct pipeline could be misused to create false or misleading content [268]. We do not endorse the use of DreamStruct to create manipulative content. Future work can investigate the potential risk of synthetic data generation and design a more robust generation pipeline that handles copyright issues. Our work may be used for accessibility purposes (especially in *accessible creativity* domains [5, 68, 79, 193]). For example, blind people may use element recognition to traverse a slide screenshot from a lecture video using their screen reader [78] and may not be able to detect when our model produces errors (*e.g.*, by missing an element). Future deployments should warn users of potential errors when using such models, and future work should continue to improve the accuracy of such models.

4.6 From Screen Structure to Ongoing Activity

DreamStruct treats a slide or interface as a structured visual state. An assistant that follows a person through software also needs the history around that state. A repeated click may signal confusion, careful comparison, or a deliberate test. An undo may repair an error or reflect ordinary refinement. The same screen can therefore call for help in one workflow and no interruption in another. The system has to connect what appears on the screen to what the person has been doing over time.

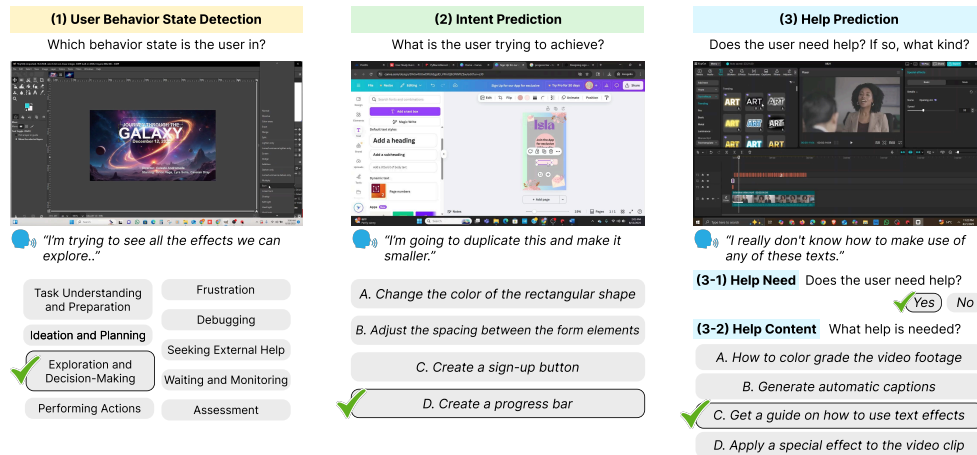


FIGURE 4.9: GUIDE separates user understanding into behavior state, immediate intent, and help prediction. The benchmark uses open-ended software activity rather than a fixed expert demonstration.

4.6.1 Recognizing Behavior, Intent, and Help Needs

GUIDE studies open-ended work in feature-rich software, where a user may still be learning both the task and the tool [3]. The benchmark separates three questions. *Behavior state detection* asks whether the person is planning, exploring, acting, debugging, frustrated, or assessing the result. *Intent prediction* asks what immediate goal explains the observed activity. *Help prediction* asks whether assistance would be useful and what form the assistance should take.

The dataset contains 67.5 hours of screen recordings from 120 novice demonstrations across ten applications for photo editing, graphic design, presentation design, video editing, and data analysis. Participants worked on open-ended tasks without a prescribed action sequence or target appearance. Think-aloud narration supported annotation, while the main evaluation asked models to infer behavior, intent, and help needs from the visual trace alone.

The evaluation shows why task automation and assistance require different models of success. The strongest behavior-state result remained below 45% accuracy, and models often read hesitation, repeated actions, or undo operations as ordinary execution. Intent prediction was easier, but models still confused nearby goals. Help prediction improved sharply when the model received structured information about behavior and intent. Longer prefixes of the workflow also helped. An assistant therefore needs more than a single screenshot or the next likely action; it needs a representation of how the person’s activity has developed.

4.6.2 Grounding Events and Transitions in Time

The GUIDE results make temporal representation a technical bottleneck. Video models often sample or compress frames to control computation, which can remove short events and transition cues. A menu that opens and closes, an edit that is immediately undone, or a cursor that returns to the same control may carry more information about the user’s state than the static frame that follows.

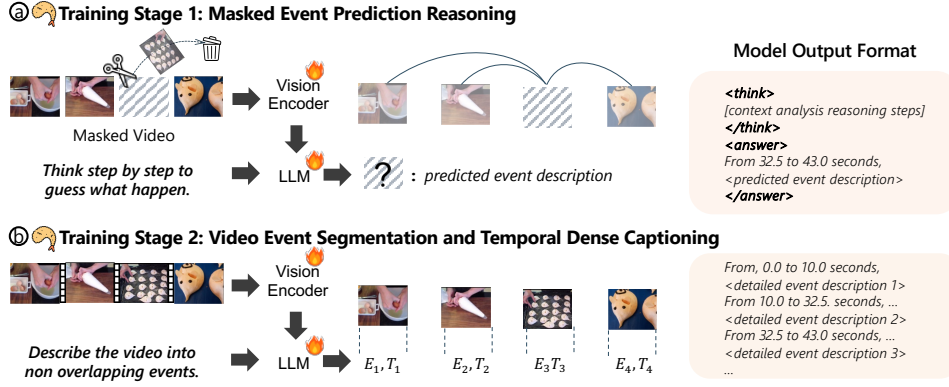


FIGURE 4.10: TEMPURA first learns event continuity through masked-event prediction, then grounds the learned event structure through timestamped segmentation and dense captioning.

TEMPURA introduces a two-stage training framework for event-level video understanding [4]. The first stage masks one event and trains the model to infer what happened from the visible events before and after it, together with a causal explanation. The second stage asks the model to divide an untrimmed video into non-overlapping events with timestamps and dense descriptions. Training uses VER, a corpus built for the framework that contains 500,000 videos, roughly 18,000 hours of footage, and one million instances across the two stages.

TEMPURA improves temporal grounding, highlight detection, and dense captioning across more than one base architecture. The stage-order ablation is especially relevant here: masked-event reasoning works best before dense grounding. A model first benefits from learning how events connect, then from learning exactly where each event begins and ends. TEMPURA does not decide whether a person needs help, but the representation can support that later judgment by showing whether the person has repeated a failed strategy, returned to an earlier state, or paused to assess a result.

4.6.3 From Representation to Intervention

The three projects address different parts of the same dependency. DreamStruct represents the structure within a screen. TEMPURA represents the events across a sequence. GUIDE asks what the sequence reveals about the person and whether help is appropriate. A complete assistant would connect the layers: perceive the interface and event history, infer the user's current goal and state, estimate whether an intervention would help, and present the suggestion or choice in a form the person can access and dismiss.

The separation also keeps different failures from collapsing into one confidence score. Weak element recognition calls for better perception. Ambiguous intent may call for a focused question. Uncertain benefit may call for a quiet, dismissible suggestion rather than a takeover. Predicting what a person is likely to do is not the same as having permission to do it for them.

The first part of the dissertation has focused on agency while an agent acts. Morae returns unresolved choices to the person, and DreamStruct supplies the structured visual understanding that such a handoff depends on. GUIDE and TEMPURA show how the same problem expands

once an assistant follows an open-ended workflow over time. The next part turns to visual work that a person has already handed to a collaborator or an agent, and asks how the person can still understand and shape the result.

4.7 Conclusion

DreamStruct uses code generation to create synthetic structured visuals with embedded labels. The method offers an efficient alternative to separate manual annotation campaigns for element recognition, captioning, and classification across presentation slides and user interfaces. The experiments show that the resulting data improves open-source models at local, screen-level, and category-level understanding.

DreamStruct supplies a perceptual foundation for collaborative agents. Code exposes element identities, geometry, hierarchy, and semantic descriptions, which makes the state easier to model and communicate. GUIDE and TEMPURA show how the problem expands across a workflow: an agent must connect interface structure to temporal activity before it can reason about assistance. Better perception cannot guarantee human agency, but an agent cannot return a meaningful choice or explain a result when it has misunderstood the environment.

Part II

Preserving User Agency in Delegated Decisions

People also delegate choices to agents on purpose, and this part asks how an agent can choose well once a decision has been delegated. Visual design is the running case. Chapter 5 makes the design decisions in a revised artifact readable to the blind and low-vision authors who delegated them. Chapter 6 measures how much of design judgment is shared across professional designers and how much is personal. Chapter 7 turns that measurement into a method that learns a new user's taste from a very few comparisons and applies it to interfaces generated for prompts the user has never seen.

Chapter 5

Diffscriber: Describing Visual Design Changes to Support Mixed-Ability Collaborative Presentation Authoring

Delegation preserves agency only when the result remains open to review.

Diffscriber makes visual revisions legible to blind presentation authors who rely on sighted collaborators for slide design. By exposing content, layout, and style changes, the system turns a finished artifact into something the author can inspect, question, and revise. The study also reveals that access to changes alone does not settle the design judgment behind them.

The previous chapters examined agency during agent action. This chapter turns to decisions embedded in the artifacts that agents and collaborators produce. A generated slide, interface, or document contains choices about content, hierarchy, layout, style, and emphasis. A user can delegate some of the work while still wanting to understand or revise the result. The problem is especially visible in mixed-ability visual authoring, where one collaborator can inspect the design directly and another cannot.

Diffscriber studies presentation authoring between blind and low-vision (BLV) presenters and sighted collaborators. The system identifies and describes changes between an initial presentation and a revised visual design. The work begins before current generative UI systems, but the workflow anticipates a central issue in agentic creation. A creator supplies goals and content; a collaborator or model produces a visual artifact; the creator then needs a representation of what changed in order to accept, reject, or refine the result.

5.1 Presentation Authoring as Mixed-Ability Collaboration

Slide presentations combine textual argument with visual structure. Titles, images, typography, color, alignment, and spatial grouping shape how an audience interprets the content. Mainstream presentation tools can expose text and alt text to a screen reader, but visual authoring remains difficult because creators need to assess the current state after each edit [65]. A screen reader

This chapter is based on work published at the 35th Annual ACM Symposium on User Interface Software and Technology (UIST 2022) [5], with Jason Wu, Jeffrey Bigham, and Amy Pavel.

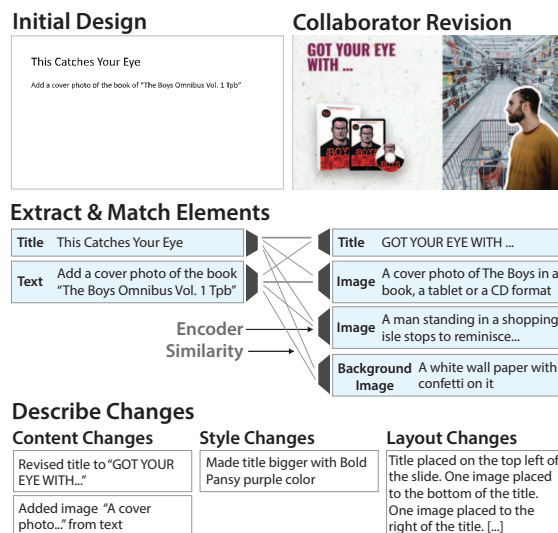


FIGURE 5.1: Diffscriber aligns an author's initial slide content with a collaborator's revised presentation, then describes content, layout, and style changes. The descriptions let BLV authors review and respond to visual decisions that ordinary slide accessibility metadata does not expose.

may announce a bounding box or element order without communicating the layout as a coherent design.

Prior accessibility work made slides easier to consume. Systems converted slides to accessible HTML, described diagrams, and extracted elements from presentation videos [77, 78, 269]. The systems primarily support an audience member who needs the content. Authoring poses a different requirement. A creator needs to know which visual decisions changed, how the changes relate to the original intent, and where feedback would affect the next revision.

Visual authoring research has explored tactile output, structured editors, and accessible tools for drawings, maps, websites, artboards, and video [65, 270–273]. Physical or tactile representations can communicate spatial structure, but each revision may require a new rendering. Mixed-ability collaboration offers a practical alternative. An author and collaborator contribute different expertise, which aligns with an interdependence view of accessibility [11]. The collaboration still depends on awareness. Document-editing studies show that blind and sighted coauthors can lose track of who changed what and where [61–63]. Slides add visual properties that text-oriented revision histories do not represent.

5.2 Formative Study

We conducted remote semi-structured interviews with nine BLV presenters who had experience creating and delivering slide-based presentations. Five participants were women and four were men, with ages from 21 to 58. Participants ranged from low vision to total blindness and presented from monthly to several times per week. The interviews covered recent authoring workflows, collaboration strategies, difficulties, and workarounds. Two presenters also shared before-and-after materials from recent presentations, which provided 37 slides for a change analysis.

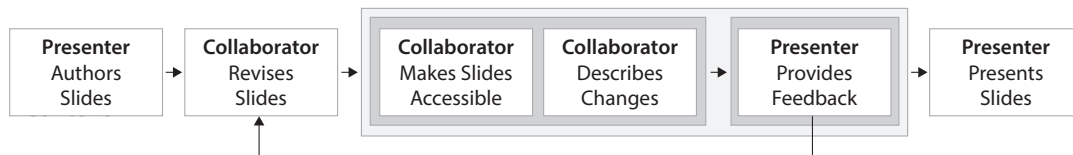


FIGURE 5.2: The formative study found a recurring asynchronous workflow. BLV presenters authored content, sighted collaborators revised the visual design, and only some workflows included a structured review and feedback round.

5.2.1 A Common Division of Labor

All nine presenters collaborated on slides rather than handling the complete visual design alone. The presenter usually wrote the content in a Word document or a minimally styled slide deck. The content included titles, bullet text, and requests for visuals. A sighted colleague, friend, or professional assistant then created the visual design.

The handoff often ended the authoring process. Six of nine presenters reported that they sometimes presented the revised slides without another review round. Time pressure, team norms, and the cost of making the slides accessible all contributed. When further communication occurred, the collaborator might describe the changes during a call, add accessibility metadata, or do both. The rare combination of a detailed verbal walkthrough and a fully accessible deck required substantial effort and usually occurred when an accessibility professional supported the workflow.

Participants valued the collaborators' speed and design expertise. The problem was not the existence of collaboration. The problem was that the presenter could participate only through the information that the collaborator remembered to communicate. Collaborators often prioritized content changes because visual style was harder to describe under limited time. A presenter could later learn from an audience member that the font was too small or that an unexpected visual element appeared, even though the issue never surfaced in the review.

Accessible slide files did not fully solve the problem. Text and image alt text exposed much of the content, but position, color, grouping, and emphasis remained difficult to interpret or compare. Presenters sometimes sought a second sighted opinion because they could not independently assess whether the collaborator's design choices served the presentation.

5.2.2 What Changed in the Slides

The 37-slide comparison and participant accounts produced a taxonomy with three high-level categories.

Content changes. Collaborators added, removed, replaced, or revised elements. Additions included titles, content-relevant images, decorative images, and shapes. Replacements could map one initial element to several revised elements, such as a bullet paragraph transformed into a row of cards. Text requests such as "insert a screenshot here" could become an image. Some text was shortened or removed to fit the layout.

Type	Operation	Examples
Content	Add	Text, relevant image, decorative image, shape
Content	Replace	Revise text; replace an instruction or text block with an image; split one element into several
Content	Remove	Remove text or an unfollowed instruction
Style	Change / define	Typeface, weight, size, color, fill, stroke, background
Layout	Reorganize	Title-to-body relation; list, row, column, or grid structure

TABLE 5.1: The change taxonomy derived from participant examples and 37 before-and-after slides.

Style changes. Collaborators defined or changed typeface, weight, size, color, fill, stroke, and background properties. Style changed the perceived hierarchy even when the textual content remained the same.

Layout changes. Collaborators changed the relation between title and body or reorganized groups of elements. A vertical bullet list could become a horizontal sequence, a grid, or a composition with an image on one side.

The taxonomy clarifies why a generic slide summary is insufficient. A summary of the revised slide answers “what is present now.” Collaboration also requires answers to “what decision did the collaborator make,” “how did the decision transform my input,” and “where should I respond.” Diffscriber organizes access around the difference rather than only the final state.

5.3 Diffscriber

Diffscriber augments Google Slides with two linked representations: a slide-content list and a change-description list. The system accepts an original and revised deck, or two snapshots of the same deck. It extracts slide elements, aligns elements across versions, classifies edits, and generates hierarchical descriptions.

5.3.1 Slide Content and Change Views

The slide-content list presents every element in an estimated reading order. The first level exposes text or image descriptions. A second level exposes properties such as position, size, and style. When an element changed, the interface links the element to the corresponding entry in the change list. The design lets an author begin with familiar content and request visual metadata only when needed.

The change-description list groups edits by layout, content, and style. Each group begins with a summary, followed by individual changes. A user can expand a summary such as “added an image above each item in the horizontal list” to inspect a specific image and its relative position. Each description links back to the affected slide element.

The hierarchy responds to a practical tension. Exhaustive visual descriptions can overwhelm a screen-reader user, while short summaries can omit the exact change that matters. Progressive disclosure lets the author choose the level of detail. The interface also supports text editing, which lets the presenter revise content without leaving the accessible view. Diffscriber augments the

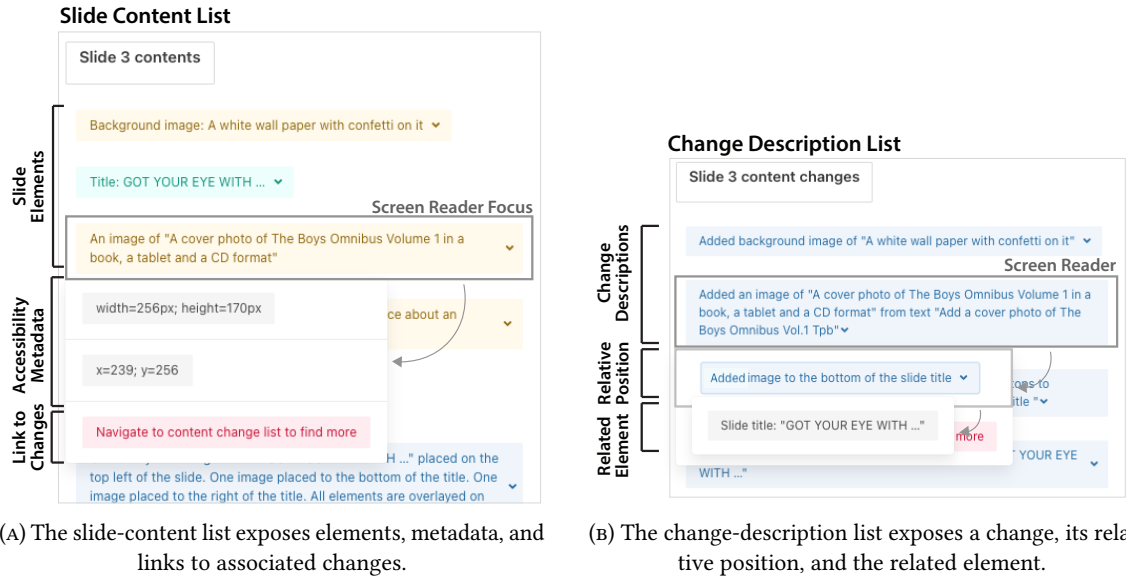


FIGURE 5.3: Diffscriber provides two linked views of the same revision. Authors can begin from familiar slide content or from a summary of changes, then move between the views for more detail.

existing editor rather than replacing it, so a user can move from a description to the corresponding object when direct editing is practical.

5.3.2 Element Correspondence

The central technical problem is correspondence. The system must determine which revised elements originated from which initial elements. A stable object ID is not enough because collaborators can copy an element, replace text with an image, split one paragraph into several cards, or merge several elements into one group.

For an original slide with M elements and a revised slide with N elements, Diffscriber constructs a similarity matrix

$$C \in \mathbb{R}^{M \times N}, \quad C_{ij} = \frac{e_i^\top e'_j}{\|e_i\| \|e'_j\|}, \quad (5.1)$$

where e_i and e'_j are semantic representations of source element i and revised element j . Text elements use their content. Image elements use human-authored alt text when available, an automatically generated caption, or a direct image embedding. A CLIP representation places text and images in a shared space [274], which allows an instruction such as “add a cover image” to align with the image that implements it.

A mutual argmax would produce a one-to-one alignment: a pair is retained only when each element is the other’s strongest match. Slide revision often creates one-to-many relationships, so Diffscriber uses an iterated maximum procedure related to word-alignment methods [275]. The algorithm extracts high-confidence pairs, updates the matrix, and repeats. The resulting graph can link one source element to several revised elements or several source elements to a shared target.

The correspondence graph drives later classification. An unmatched revised element indicates an addition. An unmatched source element indicates a removal or an instruction that the collaborator did not implement. A text-to-text match with different content indicates revision. A text-to-image match can indicate that a collaborator converted a written request into a visual. The graph also lets every description link back to the relevant elements in both versions.

5.3.3 Edit Classification and Description

Rule-based classifiers combine the correspondence graph with slide metadata. The design follows the change taxonomy from the formative study, which keeps the output close to decisions that participants wanted to review.

Content changes. A revised element without a source match is described as an addition. A source element without a revised match is described as a removal or an unfulfilled instruction. Text-to-text correspondences support sentence-level revision descriptions, while text-to-image correspondences support descriptions such as “added an image from the request.” The graph also supports one-to-many replacements, such as one paragraph becoming three horizontal cards.

Style changes. For matched elements, the system compares typeface, weight, size, color, background, shape type, fill, stroke, and stroke weight. The comparison reports changes to properties that affect visual emphasis. New elements receive a description of their defined style because no source style exists.

Layout changes. The system estimates reading order and identifies title and body groups. It compares the relation between the title and body and fits grouped elements to an approximate grid. The detector can therefore report that a vertical bullet list became a horizontal row or that the title moved from above the body to the left side. Relative descriptions avoid forcing users to interpret raw pixel coordinates.

Descriptions use controlled templates rather than open-ended generation. Templates provide predictable phrasing and reduce unsupported claims [276]. Location references anchor a new element to the nearest matched element, for example “added an image below the slide title.” Structural references replace a repeated set of element-level messages with one group-level summary. Instructional references connect a source request to the visual that implements the request.

The system orders summaries by estimated importance and supports two levels of detail. A high-level description compresses repeated edits or describes a group-level transformation. Fine-grained descriptions enumerate each detected change. The two levels let a user scan the revision before deciding which parts deserve closer review.

The pipeline produces a factual account of changes, not a complete evaluation of design quality. Diffscriber can report that the title became bold and purple, or that three images now appear in a row. The system does not claim that the choice is good for the presentation. The distinction motivates the personalized assessment work in the next chapters. An agent first needs a grounded representation of what changed; assessment adds another layer of judgment.

Representation	Precision	Recall	F1
Sentence transformer + human alt text	0.75	0.88	0.81
Sentence transformer + generated captions	0.73	0.87	0.79
CLIP + human alt text	0.78	0.86	0.81
CLIP + generated captions	0.76	0.84	0.80
CLIP image/text encoders	0.70	0.85	0.77

TABLE 5.2: Overall element-correspondence performance across the four before-and-after presentation sets.

5.4 Technical Evaluation

We evaluated correspondence because downstream change descriptions depend on the alignment. No public dataset contained before-and-after presentations authored by BLV presenters and revised by collaborators. We therefore contacted formative-study participants and collected four original-and-revised presentation pairs with different topics and authoring processes. The resulting dataset contained 61 slide pairs. We manually mapped source and revised elements and retained one-to-many relationships in the ground truth.

The evaluation compared five representations. Sentence Transformer encoded text directly and encoded images through either human alt text or generated captions. CLIP used human alt text, generated captions, or its native image encoder. Each configuration used the same matching procedure. Precision, recall, and F1 followed word-alignment evaluation so that one-to-many mappings received appropriate credit [277].

All variants achieved similar overall performance. The best configurations reached approximately 0.81 F1. Recall was generally higher than precision, which means the system found most true relationships but sometimes added an extra match. Human-authored alt text helped most on the presentation with the largest amount of visual content. Automated image representations still recovered useful correspondences when alt text was absent.

The evaluation also exposed a useful separation. Automated representations can support matching, but informative descriptions still benefit from meaningful alt text. A model can infer that two elements correspond without knowing the communicative purpose of the image. Accessibility metadata therefore serves both direct access and better machine interpretation.

The selected system used CLIP because one encoder handled both text and images and performed well across the presentation sets. The technical results also explain the main error modes. Rare proper nouns and URLs sometimes produced weak semantic matches. Stylized word art could fail to align with the source text because the image representation captured appearance more strongly than the characters. Correspondence errors affected more descriptions than template errors; once an alignment was correct, only a small fraction of descriptions contained a spatial phrasing error. The result justified a system design that keeps descriptions inspectable and linked to the underlying elements.

5.5 User Study

We conducted a remote study with six BLV presentation authors. Participants reviewed two professionally redesigned presentations under two conditions: Diffscriber and accessible slides without change descriptions. The original content came from BLV authors. A professional designer revised each deck, and five slides per deck were selected to cover additions, removals, replacements, layout changes, and style changes. Participants received the original materials before the session so they could assume the role of the author.

For each condition, participants reviewed the revised slides, identified changes, assessed the revision, and gave feedback or questions for the designer. Condition and presentation order were counterbalanced. Each slide pair had a five-minute limit. The study measured identified changes, feedback, navigation behavior, subjective ratings, and qualitative responses.

5.5.1 Change Awareness and Feedback

Diffscriber substantially increased change awareness. Participants identified an average of 28.83 changes with Diffscriber and 10.33 with accessible slides alone, a statistically significant difference ($t(10) = 12.11, p < 0.00001$). The additional descriptions made style and layout changes available, while the accessible-slide condition mainly exposed text and alt text. Participants also provided more feedback to the designer, 2.83 comments on average with Diffscriber compared with 0.83 without the change view ($t(10) = 3.96, p = 0.0027$).

Participants rated Diffscriber higher for understanding changes ($\mu = 6.33$ versus 3.33 on a seven-point scale) and for usefulness during authoring ($\mu = 6.33$ versus 3.33). Five of six participants began with the high-level change summaries and opened lower-level descriptions only for selected edits. One participant preferred to navigate from familiar slide elements and read the associated changes under each element. The difference supports the two-view design rather than a single prescribed navigation path.

The result matters because the system did more than increase recall of factual edits. Change awareness created a basis for negotiation. Participants could question an image choice, request a different layout, or ask why content had been removed. Without the descriptions, several participants reported that they did not know what feedback to give because they could not infer the design decisions.

Participants unanimously preferred Diffscriber to accessible slides alone and to their prior PowerPoint workflows. They rated content changes as the most useful category ($\mu = 6.50$), followed by style ($\mu = 6.00$) and layout ($\mu = 5.83$). Raw accessibility metadata such as coordinates and text properties remained useful ($\mu = 5.33$), but participants preferred relative and structural descriptions to absolute values. Some participants also saw an educational benefit. A creator could learn common layout patterns and style conventions from the collaborator's changes, which might support more independent authoring over time.

5.5.2 Limits of Description

Participants still encountered description errors when correspondence failed or a source image lacked meaningful alt text. A false match could produce an implausible replacement description.

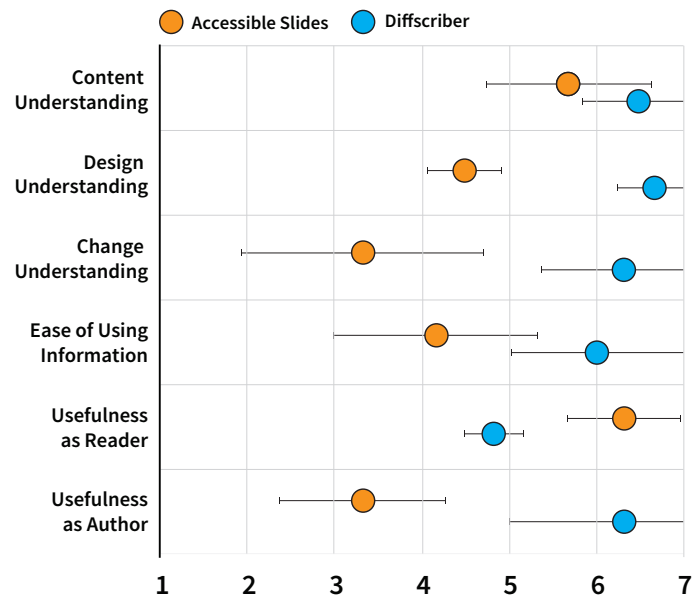


FIGURE 5.4: Participants rated Diffscriber more positively than accessible slides alone for understanding changes, reviewing the revision, and preparing feedback.

The system also focused on author-observed change types and did not represent every design property. Some participants wanted a higher-level statement about the quality or intent of the redesign, not only a list of edits.

The requests reveal the boundary between description and assessment. A description can say that a title moved to the left and became larger. A design rationale might say that the change establishes hierarchy. A preference model might say that one author usually favors the resulting asymmetric layout while another prefers a centered title. Diffscriber supplies the first layer and makes the need for the next two layers visible.

5.6 Limitations and Extensions

Diffscriber focuses on a common asynchronous workflow in which one deck preserves the slide order and a collaborator revises the visual design. The current system does not align reordered slides, detect when one slide becomes several, or summarize a theme change across the entire deck. Extending correspondence from elements to slide sequences would support larger revisions and presentation-level summaries.

The Google Slides implementation also relies on platform metadata and a browser-based extraction layer. A screenshot-only version could support other authoring tools but would introduce additional errors in element segmentation, reading order, and style extraction. DreamStruct's structured synthetic data suggests one route for training such models, while platform APIs remain preferable when they expose reliable semantics.

Description quality depends on the source material. Human-authored alt text improves both access and correspondence, especially for images with a communicative purpose that generic captioning cannot infer. Automatic captions keep the system usable when alt text is absent, but a

deployment should communicate the uncertainty rather than present every inferred relation as fact.

Finally, the detector covers changes observed in the formative study. Future systems could add overlap, clipping, contrast, and readability checks, attribute changes to individual collaborators, and stream descriptions during synchronous editing. The central design should remain stable: users need a concise overview, access to evidence, and a path from the description to the editable object.

5.7 Diffscriber as a Probe for Collaborative Agents

Diffscriber predates current general-purpose UI agents, yet the findings transfer directly to generative artifact workflows. A user may ask an agent to “make these slides look professional” or “create a settings page.” The model then makes many underdetermined decisions. A final screenshot does not tell the user which content was dropped, which defaults were introduced, or how the design changed the emphasis. A conversational rationale can help, but the rationale needs grounding in the artifact.

The study suggests four lessons for collaborative agents.

Artifacts carry decisions. The result of an agent action is not only a completed task. A visual artifact records choices about what to include, how to organize it, and what to emphasize. Collaboration requires a representation of the choices.

Comparison is often more useful than description alone. A user already knows the intended content. The important information lies in the delta between intent and result, or between two candidate results. Difference representations reduce the burden of reconstructing both states separately.

Progressive disclosure supports agency. A fixed summary omits details, while an exhaustive log overwhelms. Hierarchical descriptions let users choose how deeply to inspect a decision. The same pattern applies to agent plans, tool traces, and preference explanations.

Grounding precedes judgment. An agent should not claim that a design is clearer or more polished unless the claim corresponds to observable properties. Diffscriber’s structured change detection offers a basis for later assessment and verification.

The next chapter develops the assessment layer. UIClip shows how far a shared model of design quality can go; DesignPref then asks why trained designers still disagree among valid alternatives and whether a model can predict the judgment of a particular person. The shift moves from access to a collaborator’s decisions toward alignment when an agent evaluates or selects visual outcomes on the person’s behalf.

5.8 Conclusion

Diffscriber supports mixed-ability presentation authoring through structured descriptions of content, layout, and style changes. The formative study showed that BLV presenters commonly relied on sighted collaborators but lacked a reliable review stage. The system aligned elements

across slide versions, classified edits, and exposed summaries and details through a screen-reader-accessible interface. A technical evaluation showed that multimodal correspondence could recover most relationships across revisions, and a user study showed that change descriptions increased both awareness and feedback.

More broadly, the work reframes visual accessibility as participation in design decisions. Access to the final content is necessary, but a collaborator also needs access to how the artifact came to be. The next question is what should happen when an agent does more than expose the decision and begins to assess or select a visual outcome on the person's behalf.

Chapter 6

DesignPref: Capturing Personal Preferences in Visual Design Generation

Several designs can satisfy the same task and still reflect different people.

A shared evaluator can identify broad failures and common quality signals, as UIClip demonstrates. DesignPref begins where a shared score becomes insufficient: twenty professional designers judge the same generated interfaces, disagree in systematic ways, and support models that retain the identity behind each preference rather than average the differences away.

The previous chapter showed that access to a visual artifact includes access to the decisions embedded in its revision. A generative design agent faces a related problem before the artifact reaches the user. Given several plausible interfaces for the same request, the system has to reject broad defects, check whether each candidate fits the requested function, and decide how to rank the remaining options.

A single score can help with the first two tasks. The harder question begins after several candidates are all reasonable. The chapter first reviews UIClip as a model of broadly shared UI quality, then uses DesignPref to test whether one shared score can represent the judgments of particular people.

6.1 From Shared Quality to Personal Judgment

User interface design combines task fit, usability, accessibility, and visual communication. Expert review and usability testing provide rich evidence but require time and specialized labor. Hand-crafted metrics can estimate properties such as visual complexity, alignment, or color harmony, yet each metric captures only part of the design and may not reflect the purpose of the screen [92–94]. UIClip asks whether a learned evaluator can score a screenshot against a natural-language description while incorporating both relevance and design quality [6].

UIClip adapts the CLIP dual-encoder architecture to UI screenshots. JitterWeb supplies large-scale supervision by starting from crawled web pages and applying controllable changes to

This chapter is based on DesignPref [7], with Jeffrey P. Bigham and Jason Wu. Section 6.1 discusses UIClip, published at UIST 2024 and led by Jason Wu [6].

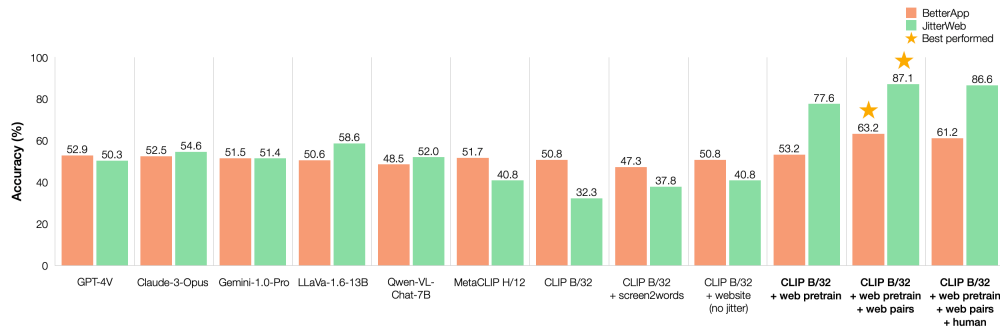


FIGURE 6.1: UIClip performs best on design-choice prediction after UI-specific training. Synthetic JitterWeb pairs provide a clear quality signal, while the professional BetterApp comparisons remain much harder.

color, contrast, font size, spacing, complexity, and layout. The process produces more than 2.3 million examples with known transformations and quality-oriented descriptions. BetterApp adds professional judgments on independent real and model-generated interfaces. Twelve designers compared screens, refined captions, selected a preferred design, and identified the principles that shaped the choice; after filtering, the collection contained 892 usable pairs. A pairwise training objective teaches the model to give the preferred screen a higher score for a description of a well-designed interface.

UIClip outperformed general CLIP models and large multimodal judges on design-choice prediction. The strongest pairwise-trained variant reached 63.2% accuracy on held-out BetterApp pairs and 87.1% on JitterWeb pairs. The representation also supported defect suggestions, relevance-aware retrieval, and reranking of interfaces generated from the same prompt. The results show that UI-specific supervision can provide a useful shared-quality prior.

The human data also exposed the limit of the prior. Agreement on the common BetterApp pairs was modest, with Krippendorff's $\alpha = 0.37$. For clearly degraded interfaces, treating disagreement as noise around a shared target is often reasonable. For valid interfaces that differ in density, tone, hierarchy, or decoration, the disagreement may reflect stable personal judgment. DesignPref changes the data collection protocol so that the same designers judge the same pool and every label keeps the identity of the person who made it.

6.2 DesignPref Dataset

Generative models can now produce several plausible interfaces from the same request. Evaluation and alignment pipelines usually aggregate human preferences through majority vote or a pooled reward model. Aggregation assumes that disagreement mainly reflects noise around one underlying utility. Visual design offers a harder case. People can agree that both candidates are usable and still prefer different information density, visual tone, hierarchy, or decorative style.

DesignPref studies the structure and modelability of the differences. The dataset contains 12,000 pairwise comparisons over generated mobile UIs. Twenty professional designers each labeled the same 600 pairs and selected one of four preference levels. A follow-up study collected natural-language rationales on the most contested pairs. The repeated shared pool supports analyses that ordinary crowdsourced benchmarks cannot provide: every pair has labels from

every designer, so the dataset can separate item difficulty from persistent differences among raters.

6.2.1 Dataset Construction

6.2.1.1 UI Generation

We sampled 100 human-written screen descriptions from an existing UI dataset [225] and expanded each description through the structured prompting method introduced in DreamStruct [2]. The prompts covered screen types from the Enrico taxonomy [226]. GPT-5 and Gemini-2.5-Pro generated executable HTML, CSS, and JavaScript at a high temperature to increase visual diversity. Each description produced four variants, and the variants yielded six within-prompt pairs. The final pool contained 400 generated designs and 600 unique comparisons.

The controlled design has two advantages. Candidate pairs share the same functional brief, so the judgment focuses on visual execution rather than different content. Every designer sees the same pool, which enables direct measurement of agreement and personalized prediction.

6.2.1.2 Annotation Protocol

We recruited 20 designers with at least one year of professional UI design experience. Each designer completed all 600 comparisons in approximately 90 minutes. The interface presented the prompt and two mobile screens side by side. Designers chose among four labels:

$$A \gg B, \quad A > B, \quad B > A, \quad B \gg A.$$

The even-numbered scale omitted a neutral option so designers would express slight preferences when neither screen clearly dominated. Each label remained linked to a persistent anonymized designer identifier. The study compensated participants at \$15.55 per hour and followed an institutionally approved protocol.

6.2.2 How Much Do Designers Agree?

Each judgment can be represented at two granularities. The binary label records which screen wins. The four-way label also records whether the preference is slight or strong. Agreement is modest under both views.

For binary preferences, mean pairwise agreement is 0.624. Mean Cohen’s κ and multi-rater Krippendorff’s α both equal 0.248. For four-way labels, raw agreement falls to 0.386, Cohen’s κ to 0.114, and Krippendorff’s α to 0.104. Designer pairs who agree on direction tend to agree more on strength, with a correlation of $r = 0.60$, but strength introduces another source of variation.

Part of the gap comes from scale use. Across all labels, 75.54% fall in the middle categories and 24.46% in the extreme categories. Individual calibration varies dramatically. Some designers almost never use a strong label, while others use strong labels for most comparisons. A four-way model must therefore learn both what a person prefers and how the person expresses confidence.

Agreement also depends on screen type. Form screens show the strongest consensus, with 72.11% pairwise agreement and the highest share of extreme choices. Lists, dialers, and search

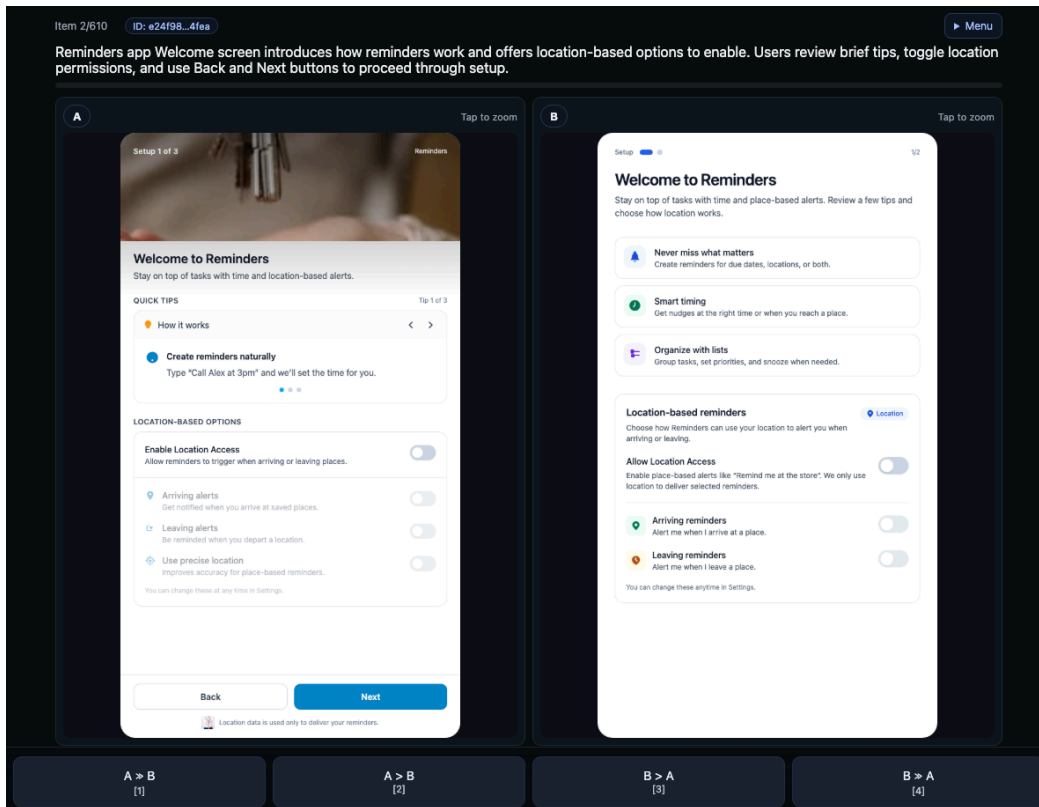


FIGURE 6.2: DesignPref’s annotation interface presents a shared functional prompt and two generated variants. Each designer records both preference direction and strength.

screens follow at roughly 66 to 68%. Terms pages, camera interfaces, media players, and login flows remain near 54 to 57%. Across screen types, agreement correlates strongly with the use of extreme labels ($r = 0.774$). Structured layouts with a clear primary action tend to produce a visible winner. Multifunctional or stylistically open screens leave more room for individual taste.

The findings argue against a simple label-noise interpretation. Some items are broadly agreed upon, especially when one option has a clear functional or readability advantage. Other items split the same trained designers in stable ways. A useful judge should preserve both patterns: a shared prior for common defects and a person-specific component for legitimate trade-offs.

6.3 Why Designers Disagree

Agreement statistics show the magnitude of divergence but not its content. We invited the original designers to a rationale study; 13 returned. We selected 106 highly contested pairs with a majority margin of at most ten percentage points, which corresponds to a 55/45 split or closer. Designers saw their earlier choice and wrote a brief explanation. The collection produced 1,378 rationales.

6.3.1 Common Themes

We analyzed the rationales through a two-stage clustering pipeline. Each sentence was embedded with Sentence-BERT [278]. Fine-grained clusters received a neutral label and summary, then

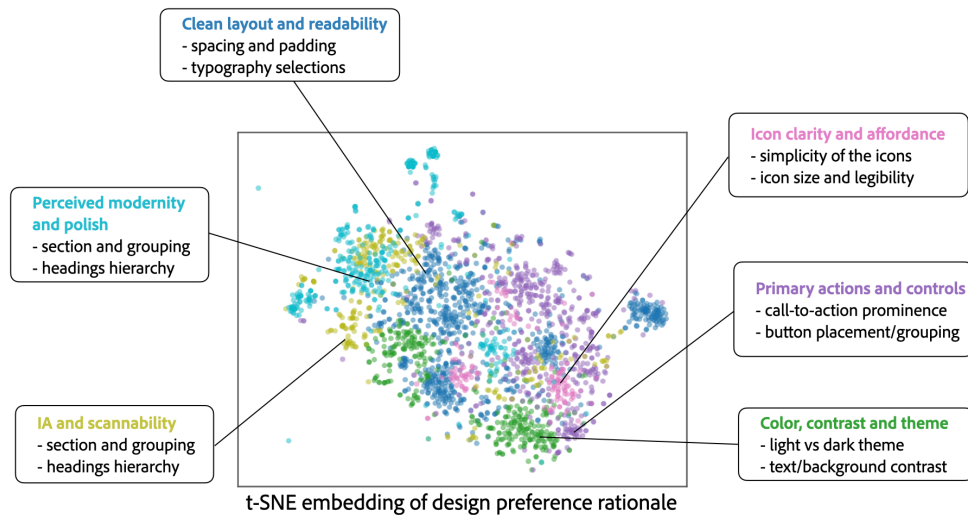


FIGURE 6.3: A t-SNE projection of rationale embeddings reveals six recurring themes. Designers share a vocabulary of design concerns even when they choose different candidates.

related clusters were merged into broader themes. Six recurring concerns appeared:

1. **Clean layout and readability:** spacing, alignment, type size, and visual calm.
2. **Primary actions and controls:** prominence, reachability, and separation from secondary actions.
3. **Perceived modernity and polish:** typography, spacing consistency, cohesive accents, and professional tone.
4. **Color, contrast, and theme:** background tone, palette cohesion, and emphasis through contrast.
5. **Information architecture and scannability:** sections, grouping, hierarchy, and comparison structure.
6. **Icon clarity and affordance:** size, contrast, labels, and the relation between icon form and action.

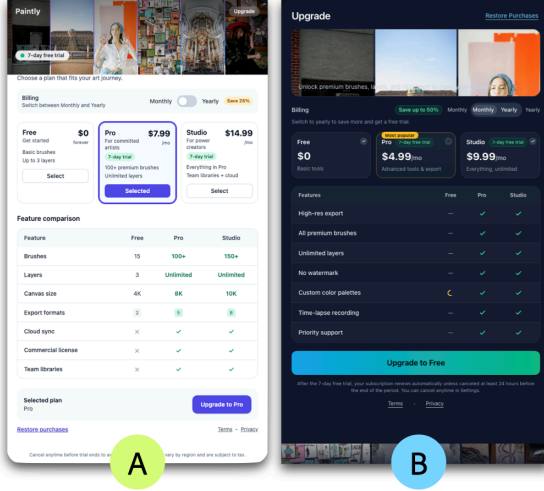
The themes resemble familiar design principles, which means disagreement does not arise because some designers ignore design knowledge. Designers often cite the same concern and reach opposite conclusions. The divergence lies in how they define, prioritize, or apply the concern to the current task.

6.3.2 Divergent Trade-offs

We analyzed each contested pair as a unit. GPT-5 received the rationales from both sides and summarized the main trade-off; a second pass grouped related trade-offs. Four patterns recurred.

Information density. Some designers preferred calm layouts with fewer visible choices. Others preferred denser screens that exposed more plans, support channels, or browsing options without another navigation step.

Prompt: Paintly art app. Upgrade screen displays Free, Pro, and Studio plans with feature comparison, monthly/yearly toggle, pricing, free trial badge, Upgrade and Restore buttons, plus terms and privacy links.



A >> B

I prefer that the image A has a light background, that the 3 options are separated on the same line, and that the differences between them are also visible in the lower area.

A > B

Both respond well to the prompt given, so it comes down to details. It was a matter of personal preference, since I tend to prefer the whites, and the whole aspect of the layout seemed more refined and organized.

B > A

Both designs used a visually appealing design for billing plans and feature comparisons, which help make a lot of information stand out quickly. However, I slightly preferred B, as I felt that the background color used provided more contrast and made the information easier to read.

B >> A

I like the layout here better. The list is very easy to read. The gradient button looks beautiful. The darker colors make it easier to read overall

FIGURE 6.4: Contested pairs reveal recurring trade-offs rather than arbitrary disagreement. Designers differ on density, palette, imagery, and the amount of functionality that should remain visible.

Visual style and tone. Light, neutral themes conveyed trust and legibility to one group. Darker or more saturated palettes conveyed premium quality, expressiveness, or seriousness to another.

Decorative imagery versus focused utility. Large photography, gradients, and illustration added warmth and brand identity for some designers. Other designers preferred restrained, text-led layouts that kept attention on the task.

Action prominence and task scope. Task-first designers favored one dominant action with minimal chrome. Feature-first designers valued visible modes, filters, alerts, and secondary controls and viewed sparse interfaces as underpowered.

The rationale analysis motivates personalization at the representation level. A fixed rubric can name the relevant dimensions, but the rubric does not specify a universal weighting. A profile such as “prefers clean interfaces” also remains underspecified because one person may interpret cleanliness as generous whitespace and another as compact, well-organized density. Concrete pairwise choices reveal how the person resolves a trade-off in context.

6.4 Personalized Design Preference Models

DesignPref evaluates two common assessment architectures: a CLIP-style encoder and a decoder-style multimodal model with retrieval-augmented generation. Both experiments compare personalized supervision against pooled supervision.

6.4.1 Data Representation and Splits

A labeled comparison is

$$\mathcal{D}_{\text{pref}} = \{(t, I^A, I^B, c) : c \in \{-2, -1, 1, 2\}\}, \quad (6.1)$$

where t is the screen description, I^A and I^B are the screenshots, and c is the signed strength label. We used 60/20/20 train, validation, and test splits. All pairs that contained a given screen ID remained in the same split to prevent visual leakage. Each designer contributed 360 training, 120 validation, and 120 test labels. The full dataset contained 7,200 training and 2,400 validation and test labels.

6.4.2 UIClip Fine-Tuning

The first model starts from the UIClip checkpoint trained on synthetic UI defects but not on BetterApp’s human preferences. A pairwise margin loss learns a higher score for the preferred screen. DesignPref extends the loss so a strong preference requires a larger margin:

$$\mathcal{L}(s^A, s^B; \hat{m}) = \max\{0, \hat{m} - (s^A - s^B)\}, \quad \hat{m} = \begin{cases} m_1 & \text{if } A > B, \\ m_2 & \text{if } A \gg B. \end{cases} \quad (6.2)$$

Negating the score difference covers preferences for B . At inference, the sign predicts direction and a threshold predicts strength.

A separate model is trained for each designer. Only the final UIClip layer is unfrozen because 360 examples are too few for full-backbone adaptation. Weight decay and early stopping reduce overfitting. Two pooled baselines use either a matched-size sample from all designers or the full combined dataset, which contains twenty times as many labels as one personal model.

6.4.3 Personalized Retrieval-Augmented Generation

The second setup adapts UICrit’s multimodal retrieval pipeline [102]. Sentence-BERT embeds the screen description, and UIClip embeds each screenshot. The pair index stores both screenshot orders so retrieval remains invariant to whether a candidate appears on the left or right. At inference, the system retrieves eight similar labeled examples and inserts them into the model prompt.

Personalized retrieval uses only the target designer’s training labels. Pooled retrieval draws from all designers. Because every designer labeled the same pairs, pooled retrieval aggregates the labels for a retrieved pair into one rounded score. The comparison tests whether a small, identity-matched context provides more useful evidence than a larger context that represents consensus.

6.5 Evaluation

We report binary preference accuracy, exact four-way accuracy, and Spearman rank correlation coefficient (SRCC) between predicted and ground-truth strength labels. Metrics are computed separately for each designer and macro-averaged.

Model group	Setup	Binary acc. (%)	Four-way acc. (%)	SRCC
Personalized models				
UIClip fine-tuning	Strength-aware margin	60.16	34.37	0.217
	Binary margin	57.24	28.89	0.180
LMM judge, personalized RAG	GPT-5, 8-shot	58.89	38.53	0.211
	Gemini-2.5-Pro, 8-shot	56.53	22.35	0.170
	Qwen3-VL-235B-Thinking, 8-shot	56.83	42.67	0.132
	Qwen3-VL-30B-Thinking, 8-shot	56.20	40.41	0.128
Non-personalized models				
UIClip pooled	Matched size, N	56.73	32.77	0.150
	Full pool, $20N$	57.45	41.23	0.196
LMM judge, pooled RAG	GPT-5, 8-shot	57.62	34.36	0.203
	Gemini-2.5-Pro, 8-shot	56.15	22.56	0.169
	Qwen3-VL-235B-Thinking, 8-shot	55.78	41.59	0.122
	Qwen3-VL-30B-Thinking, 8-shot	54.78	38.79	0.110
Baselines	UIClip pretrained	55.07	23.97	0.126
	OpenAI CLIP B/32	46.68	23.26	-0.009
	GPT-5 zero-shot	57.70	31.35	0.216
	Gemini-2.5-Pro zero-shot	53.65	17.76	0.135
	Qwen3-VL-235B-Thinking zero-shot	55.36	22.09	0.131
	Qwen3-VL-30B-Thinking zero-shot	54.82	16.17	0.117

TABLE 6.1: Personalized and pooled models on each designer’s held-out comparisons. N denotes the per-designer training size; $20N$ denotes the full pooled set.

6.5.1 Personalized Fine-Tuning

The per-designer UIClip with a strength-aware margin reaches 60.16% binary accuracy and 0.217 SRCC. The pretrained UIClip reaches 55.07% and 0.126, while the original CLIP model remains below chance on ranking. The personalized model also exceeds both pooled variants on binary accuracy and SRCC, even though the full pooled model sees twenty times more labels.

The full pooled model performs better on exact four-way accuracy, 41.23% compared with 34.37%. The extra data likely helps the model learn global class boundaries, while individual designers use the strong categories at very different rates. Binary accuracy and SRCC better capture whether the personalized model selects and orders the candidates in the same way as the person. Exact four-way prediction remains an important calibration problem.

The strength-aware loss outperforms a binary-margin variant across all metrics. Four-level labels therefore contain useful information when the objective exposes the magnitude. Collapsing the labels to winner and loser discards evidence about how decisively a designer distinguishes two screens.

6.5.2 Personalized Retrieval

Retrieval shows a similar but smaller pattern. Personalized GPT-5 retrieval improves binary accuracy from 57.70% zero-shot to 58.89% and four-way accuracy from 31.35 to 38.53%. Both Qwen models gain one to two binary points and more than twenty four-way points. Personalized retrieval also exceeds pooled retrieval for GPT-5 and both Qwen variants. Gemini remains nearly unchanged, which suggests that eight demonstrations do not override its consensus prior.

The gap between personalized and pooled retrieval is narrower than the fine-tuning gap. A decoder model must infer the person’s latent preference from a small set of examples inside a

long prompt. The model may treat the examples as task demonstrations rather than evidence about the user. The result motivates the explicit user representation developed in Chapter 7.

6.6 Shared Constraints and Personal Trade-offs

Shared design constraints and personal preferences address different parts of assessment. A model can learn a broad prior that recognizes relevance and many visual defects, while the remaining choice among valid candidates often depends on the person. A practical agent needs both kinds of evidence.

One way to formalize the distinction is to decompose a design score for candidate I and user u :

$$S(I \mid t, u) = S_{\text{shared}}(I \mid t) + S_{\text{personal}}(I \mid t, u), \quad (6.3)$$

where t is the design brief. The shared term can penalize broken layout, illegible contrast, missing content, or poor prompt relevance. The personal term can weight density, tone, hierarchy, imagery, and feature exposure for the user. The decomposition is conceptual rather than a claim that the two terms are statistically independent. Personal requirements can also concern accessibility, and a shared quality prior can reflect population biases. The distinction still gives an agent a useful decision rule: do not personalize away basic correctness, and do not treat every valid trade-off as if one universal answer exists.

DesignPref also changes how disagreement should be handled in evaluation. Majority vote is appropriate when the benchmark seeks a population preference. Majority vote is misleading when the deployment goal is to serve a known individual. A model can improve agreement with the majority while becoming less accurate for many users whose tastes differ from the majority. Identity-linked evaluation makes the target explicit.

6.7 Limitations and Open Questions

DesignPref includes 20 professional designers, which supports dense repeated judgments but limits analyses of larger preference communities. A broader population could reveal clusters associated with experience, domain, culture, ability, or design tradition. The current data also emphasizes disagreement across people. Repeated labels and behavior traces suggest that a person's judgment can change with context, attention, or uncertainty, but the dataset does not fully model intra-person variation.

The modeling experiments cover two representative architectures. Decoder-model fine-tuning, latent user embeddings, hierarchical Bayesian models, and rationale-conditioned preference models remain open. The best accuracy also leaves substantial room for improvement. Some comparisons may not be predictable from screenshots and a short prompt because the designer's reasoning depends on a larger product context.

A practical limitation is annotation burden. Six hundred labels took roughly 90 minutes. The result establishes that personal data can outperform much more pooled data, but it does not identify the smallest useful budget for a new user. Chapter 7 addresses the cold-start problem by

using prior designers as a structured prior and selecting a small set of informative onboarding comparisons.

6.8 Conclusion

DesignPref shows that shared quality does not determine every design choice. Twenty trained designers disagreed substantially on the same generated interfaces, and their rationales revealed coherent trade-offs in density, tone, imagery, and action scope. Personalized models predicted individual judgments more accurately than pooled models trained on much larger datasets.

The lesson is that an agent needs a layered assessment function. Shared supervision can protect relevance and basic quality, while identity-linked supervision aligns the remaining choices with a person. The next chapter turns the lesson into an interaction pipeline that learns a useful preference model from only eight pairwise answers and applies the model to generative UI selection.

Chapter 7

Efficient Personalization of Generative User Interfaces

Personalization has to begin before a system knows much about the person.

The final method treats prior users as a structured prior and selects a small set of comparisons that reveal how a new user resolves concrete design trade-offs. The learned profile then reranks newly generated interfaces. The chapter tests both preference prediction and end-to-end generation under an eight-comparison onboarding budget.

Chapter 6 established that a global design judge cannot serve every user equally well. The finding matters for human agency because a delegated choice can still reflect the wrong person's values. The result creates a practical problem. A system can train a separate model for a designer who has supplied hundreds of labels, but a deployed generative interface cannot ask every new user to complete a 90-minute annotation study. The system needs to infer useful preferences from a small interaction budget and then apply the inferred preferences to interfaces that did not exist during training.

This chapter develops a cold-start personalization pipeline for that setting. The method uses the repeated judgments in DesignPref as a structured prior. A new user does not begin as an unconstrained point in a large preference space. The method first represents the user as a distribution over designers in the prior data bank. It then selects a small number of pairwise questions that are expected to distinguish among the plausible designers. After each answer, a Bayesian update shifts probability toward prior designers whose choices resemble the new user's choices. A retrieval model combines the resulting user distribution with visually similar comparisons from the data bank. The model can then rank new candidate interfaces for the user.

The design addresses three constraints that arise in generative UI. First, only a few explicit labels are available for a new user. Second, each prompt creates a new candidate set, so personalization cannot rely on a fixed catalog. Third, users may describe broad tastes in language but still resolve concrete trade-offs differently from what their descriptions imply. Pairwise behavior provides evidence at the level where the final selection occurs.

The chapter makes two main empirical claims. In a leave-one-designer-out evaluation, the retrieval model outperforms both the pretrained UIClip evaluator and a much larger multimodal

This chapter is based on *Efficient Personalization of Generative User Interfaces* [8], with Samarth Das, Jeffrey P. Bigham, and Jason Wu.

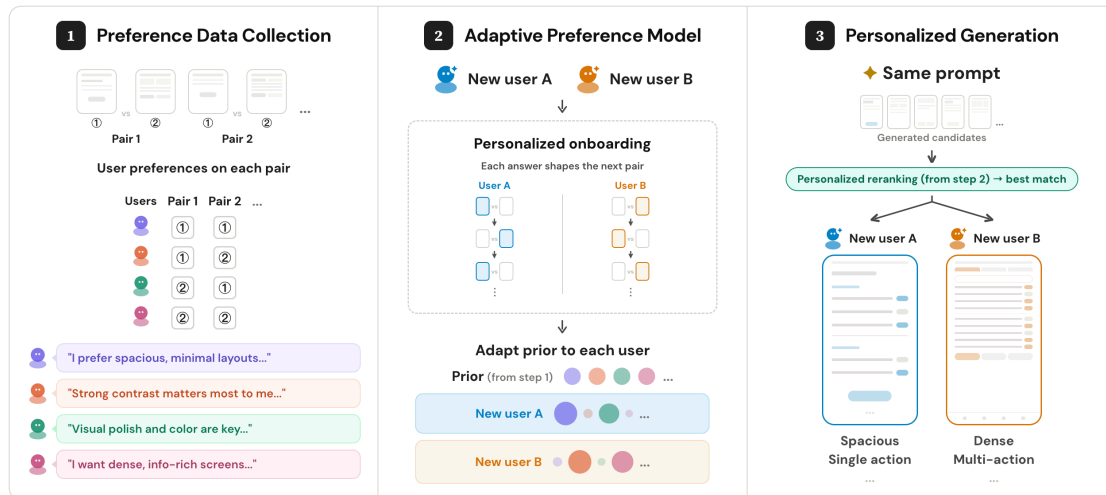


FIGURE 7.1: The personalization pipeline. A shared preference corpus provides repeated labels from prior designers. Adaptive onboarding estimates a distribution over those designers for a new user. The learned distribution then guides retrieval and reranking over newly generated UI candidates.

model. The retrieval model also improves as it receives more onboarding answers, while direct in-context personalization with the larger model does not. In an online study with 12 new designers, eight adaptive pairwise comparisons improve end-to-end generative UI selection beyond zero-shot generation, generation conditioned on a free-text profile, and reranking by a profile-conditioned large multimodal model.

7.1 The Cold-Start Personalization Problem

A generative UI system can sample several plausible implementations for the same request. The system may vary density, typography, palette, hierarchy, imagery, and the amount of visible functionality. A generic evaluator can reject candidates with broken layout or poor prompt relevance, but several remaining candidates may satisfy the task equally well. The system then needs a user-specific selection function.

A direct approach would ask the user to describe a preference profile. Natural language has clear advantages. A profile is easy to collect, can include high-level goals, and can transfer across tasks. Yet a profile also inherits the vocabulary problem observed in earlier adaptive systems: users may lack terms for latent preferences, and the same term may refer to different design choices. A person who asks for a “clean” interface may prefer sparse screens with generous whitespace, or compact screens with clear grouping and no decoration. A person who asks for “modern” design may still reject low-contrast gradients or unfamiliar controls. A broad description does not reveal how the person resolves a concrete conflict between two valid designs.

A second approach would fine-tune a model after a brief onboarding. Conventional fine-tuning is poorly suited to the eight-label regime studied here. The number of examples is too small to support stable adaptation of a vision-language model, and per-user training would add latency, storage, and deployment complexity. A decoder model can receive the eight examples in context, but the model must infer a latent preference policy from demonstrations that also

contain task content and visual details. The model may treat the examples as independent design judgments rather than as evidence about one person.

We instead frame cold-start personalization as inference over a prior population. DesignPref provides a particularly useful prior because all 20 designers judged the same 600 comparisons. For every candidate onboarding pair, the system knows how each prior designer responded. The pattern of agreement and disagreement gives the system a map of the preference space. A new user's answer can therefore rule in or rule out several preference profiles at once.

The framing leads to four design requirements:

1. **Use shared data as a prior, not as the final target.** Population data should support a new user before any personal evidence exists, but later evidence should move predictions away from the average.
2. **Spend the question budget selectively.** Onboarding pairs should distinguish plausible user profiles rather than repeat comparisons on which prior designers already agree.
3. **Generalize beyond the onboarding items.** The model must transfer from a few labeled pairs to newly generated candidates under different prompts.
4. **Preserve pair symmetry and task relevance.** A prediction should not depend on whether a candidate appears on the left or right, and retrieved evidence should concern visually comparable trade-offs.

7.2 Adaptive Preference Modeling

The method contains three components: a labeled data bank, an adaptive query policy, and a personalized retrieval model. Figure 7.2 summarizes the interaction. The user distribution is initialized uniformly, then becomes more concentrated as the user answers informative questions.

7.2.1 Preference Data Bank

Let

$$x = (t, I^A, I^B) \tag{7.1}$$

denote a comparison, where t is the functional UI prompt and I^A and I^B are two candidate screenshots. Each judgment in the data bank is

$$\mathcal{D} = \{(x, d, c) : c \in \{-2, -1, 1, 2\}\}, \tag{7.2}$$

where d is an anonymized designer identifier and c records preference direction and strength. Negative values favor B , positive values favor A , and magnitude distinguishes slight from strong preferences.

We retain the 60/20/20 split from Chapter 6. All pairs that contain the same screen ID remain in one split to avoid visual leakage. The data bank contains 7,200 training labels and 2,400 labels

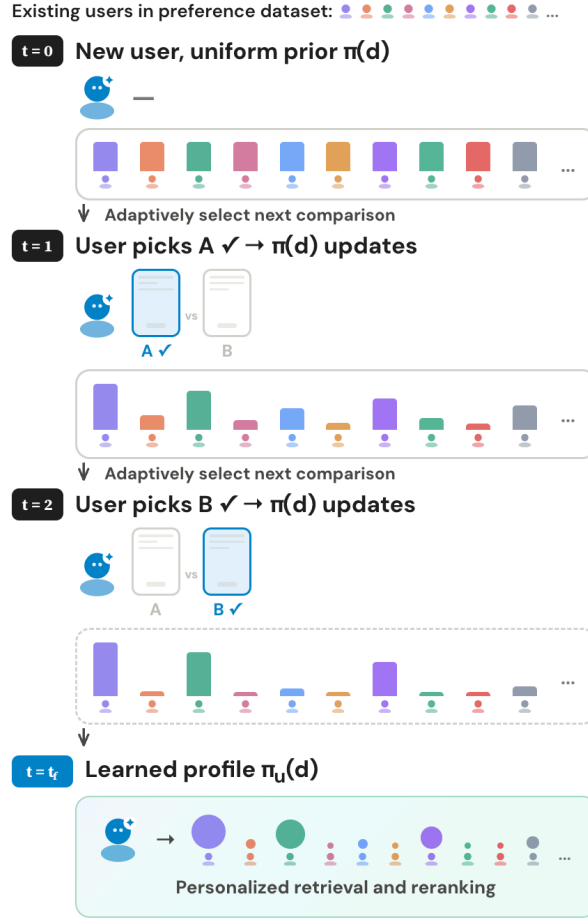


FIGURE 7.2: Adaptive onboarding represents a new user as a distribution over prior designers. After each pairwise answer, probability shifts toward designers whose prior labels are consistent with the answer. The system selects the next comparison to reduce uncertainty in the distribution.

in each of the validation and test splits. Each prior designer contributes 360 training labels. The repeated shared pool means that one bank pair has labels from every prior designer.

The data bank serves two roles. The labels provide the population structure used during onboarding. The screenshots provide examples from the target distribution used during retrieval. The second role matters because UIClip was trained largely on real web pages with synthetically introduced defects, while the current task compares fully generated interfaces that often contain no obvious defect. Access to in-domain comparisons helps the model represent subtler trade-offs among valid candidates.

7.2.2 A User Profile as a Distribution over Prior Designers

Suppose a new user has answered t onboarding questions:

$$\mathcal{O}_t = \{(x_1, c_1), \dots, (x_t, c_t)\}. \quad (7.3)$$

We represent the user's current profile by a categorical distribution π_t over the designers in the training bank. The value $\pi_t(d)$ is the weight assigned to designer d . At $t = 0$, the system uses

a uniform distribution. The initial prediction therefore averages across prior designers rather than privileging one arbitrary profile.

For candidate pair x , the current predictive distribution over possible answers is

$$p(c \mid x, \pi_t) = \sum_d \pi_t(d) p(c \mid x, d), \quad (7.4)$$

where $p(c \mid x, d)$ estimates the probability that designer d would provide label c on pair x . The repeated labels in the data bank make the designer-specific response pattern observable. In practice, smoothing prevents a single mismatch from assigning zero probability to a designer.

After the user provides answer c on pair x , the system applies Bayes' rule:

$$\pi_{t+1}(d) = \frac{\pi_t(d) p(c \mid x, d)}{\sum_{d'} \pi_t(d') p(c \mid x, d')}. \quad (7.5)$$

The update increases the weight of designers whose response pattern matches the observed answer. The representation is deliberately a mixture rather than a nearest-neighbor assignment. A new user need not replicate one historical designer. The distribution can combine designers who match different parts of the user's taste, and uncertainty can remain when the available answers do not distinguish them.

The profile also has an interpretable confidence measure. Entropy $H(\pi_t)$ is high when many designers remain plausible and low when probability concentrates on a small subset. The system uses the entropy to choose the next question.

7.2.3 Expected Information Gain for Query Selection

A random onboarding policy can waste a limited budget on pairs that reveal little. For example, if nearly every prior designer prefers a legible form over a broken form, either answer provides little evidence about the new user's distinctive taste. A better question presents a trade-off that separates the currently plausible designers.

For candidate pair x , the system estimates the expected posterior entropy:

$$\mathbb{E}[H(\pi_{t+1}) \mid x] = \sum_{c \in \{-2, -1, 1, 2\}} p(c \mid x, \pi_t) H(\pi_{t+1}). \quad (7.6)$$

The expected information gain is

$$\text{EIG}(x) = H(\pi_t) - \mathbb{E}[H(\pi_{t+1}) \mid x]. \quad (7.7)$$

At each step, the policy selects the unused pair with the largest EIG. Because π_t changes after every answer, two users can receive different later questions even if they begin with the same pool. The policy therefore adapts both to the population structure and to the user's particular path through the preference space.

The objective is not to discover a complete verbal theory of the user's taste. The question budget is too small for such a goal, and some preferences remain tacit. The objective is narrower: collect evidence that improves predictions over the candidate distinctions that the data bank can support.

7.2.4 Personalized Retrieval and Pairwise Scoring

After onboarding, the model predicts preferences on a new query pair $q = (I_{\text{left}}, I_{\text{right}})$. UIClip’s visual encoder maps each screenshot to an embedding. A learnable residual projection adapts the frozen embeddings to the generated-UI comparison domain without retraining the full encoder.

Let (z_L, z_R) be the embeddings of the query candidates, and let $(z_{j,L}, z_{j,R})$ be the embeddings for bank pair j . We score both the direct and swapped alignments:

$$s(q, j) = z_L^\top z_{j,L} + z_R^\top z_{j,R}, \quad (7.8)$$

$$s'(q, j) = z_L^\top z_{j,R} + z_R^\top z_{j,L}. \quad (7.9)$$

A soft maximum combines the alignments. The operation preserves pair symmetry: swapping both candidates should invert the predicted direction rather than change which visual comparison is considered similar. The model retrieves the top $n = 100$ bank pairs and assigns each one a similarity weight w_{qj} .

For user u , onboarding produces weights $\pi_u(d)$ over historical designers. The pairwise preference score is

$$s(q | u) = \sum_j w_{qj} \sum_d \pi_u(d) c_{d,j}, \quad (7.10)$$

where $c_{d,j}$ is designer d ’s label for bank pair j . The inner sum personalizes the label for each retrieved comparison. The outer sum transfers the personalized evidence from similar comparisons to the query. The sign of $s(q | u)$ predicts the preferred side; the magnitude expresses confidence or strength.

For a candidate set $C = \{I_1, \dots, I_K\}$, the system derives a pointwise score $f_u(I_i)$ by aggregating pairwise comparisons between I_i and the other candidates. The generation pipeline returns the highest-ranked candidate. This retrieve-and-rerank design works with a closed-source generator because personalization occurs after sampling. It also controls generation randomness during evaluation: several selection methods can rank the same candidate pool.

7.3 Offline Technical Evaluation

The offline evaluation asks whether the method predicts a new designer’s held-out choices and whether additional onboarding examples improve prediction.

7.3.1 Protocol and Baselines

We use leave-one-designer-out cross-validation. Each of 20 folds treats one designer as the new user and the remaining 19 as the prior data bank. The onboarding simulator selects k questions from the training split with the EIG policy and reveals the held-out designer’s recorded answers. The personalized model then predicts the held-out designer’s 120 test comparisons. Results are averaged across folds.

We compare three systems:

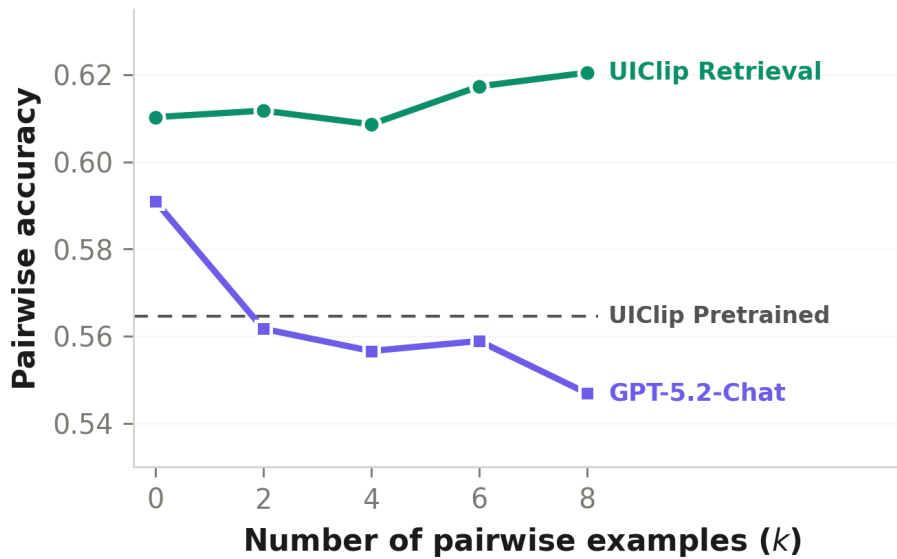


FIGURE 7.3: Pairwise prediction accuracy as the onboarding budget k increases. The retrieval model performs best at every budget and improves to 0.620 at eight examples. GPT-5.2-Chat declines as more preference examples enter the prompt. The dashed line marks the pretrained UIClip baseline.

1. **UIClip pretrained.** The released UIClip checkpoint scores candidates without access to DesignPref labels or personal evidence.
2. **GPT-5.2-Chat.** The multimodal model acts as a preference judge. At $k = 0$, it judges the pair zero-shot. For $k > 0$, the prompt includes the user's prior onboarding choices and asks the model to infer the user's preference on the new pair. The prompt includes at most eight examples to control latency and context length.
3. **Personalized UIClip retrieval.** The proposed model uses the in-domain data bank, the learned designer distribution, and the retrieval score in Equation 7.10.

The comparison isolates two questions. UIClip tests whether a generic UI quality metric is sufficient. GPT-5.2 tests whether model scale and direct in-context reasoning can replace an explicit preference representation.

7.3.2 Results

Figure 7.3 shows pairwise accuracy as a function of the onboarding budget. Before personalization, the retrieval model reaches 0.610 accuracy. UIClip reaches 0.565, and GPT-5.2 reaches 0.591. The retrieval model shares its base visual encoder with UIClip, so the 4.5-point difference does not come from a larger backbone. The result instead reflects access to labeled examples from the generated-UI distribution and a retrieval objective that preserves local decision boundaries.

With eight onboarding labels, the retrieval model reaches 0.620. The absolute gain over its unpersonalized condition is modest, but the direction is consistent for most budgets. The model starts from a strong population prior, so eight answers refine an already useful predictor rather than construct a preference function from scratch.

GPT-5.2 behaves differently. Accuracy declines as the prompt receives more onboarding examples. The result does not imply that in-context personalization must fail, but it shows that demonstrations alone do not guarantee a usable user model. The examples mix visual content, task requirements, and personal choices. Without an explicit latent representation, the model may overfit to superficial details, infer an unstable rule, or become distracted by examples that are only weakly relevant to the query. The larger model has more general visual and design knowledge, yet the smaller retrieval system uses personal evidence more reliably.

The evaluation also clarifies the role of adaptive query selection. EIG seeks questions that distinguish among currently plausible designers. The policy therefore spends the small budget on the part of the prior where a user answer can alter later predictions. Random examples can still help if the data bank is dense, but random selection offers no guarantee that an answer changes the user distribution.

7.4 Online Study of Personalized Generation

Offline prediction establishes that the model can reproduce held-out labels from the static corpus. A deployed system faces a harder test. It must personalize for people absent from the data bank, rank interfaces generated on demand, and improve the final output rather than only a benchmark score. We conducted an online study to test the full pipeline.

7.4.1 Participants and Procedure

We recruited 12 new designers, ages 25 to 32, with at least one year of professional UI or UX design experience. Five participants identified as male and seven as female. Every participant had used generative AI tools in a design workflow. The asynchronous study lasted approximately 45 to 60 minutes and compensated participants at the same hourly rate as the DesignPref collection.

The study had two phases. During onboarding, each participant first wrote a free-text description of the interfaces they preferred. The prompt remained open-ended and mentioned possible topics such as simplicity, density, color, visual tone, and interaction affordance without imposing a fixed rubric. The participant then answered eight pairwise comparisons selected by the adaptive EIG policy. Because selection depended on earlier answers, all 12 participants received an individualized query sequence.

During the arena phase, participants compared outputs from four generation pipelines:

GPT-5.2-Chat served as the generator and as the profile-conditioned judge. We created 20 prompts, each from a distinct screen type and with no overlap with the DesignPref prompts. For every profile-conditioned prompt, the system generated a common pool of 16 candidates. The profile judge and adaptive model each selected one candidate from that pool. When two conditions selected the same candidate, the comparison became an automatic tie and the duplicate trial was removed. Participants completed randomized arena-style comparisons among the four conditions, with at most 120 comparisons per participant.

We analyze the arena with Bradley-Terry ratings, aggregate win rates, direct head-to-head results, and participant-level bootstrap samples. Higher ratings indicate that a condition wins more pairwise comparisons.

Condition	Generation and selection procedure
Zero-shot generation	Generate one UI without any user information.
Profile-conditioned generation	Generate a UI with the participant’s free-text preference profile in the prompt.
Profile generation + profile judge	Generate a shared candidate pool with the profile, then ask a profile-conditioned multimodal judge to select the top candidate.
Profile generation + adaptive model	Generate the same shared candidate pool with the profile, then use eight pairwise onboarding labels and the proposed model to select the top candidate.

TABLE 7.1: The four end-to-end generation conditions in the online study. The two reranking conditions share the same text profile and candidate pool, which isolates the quality of the final selector.

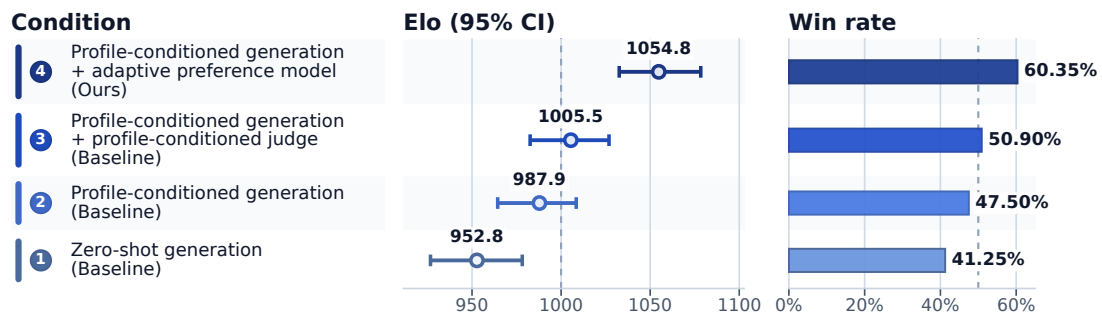


FIGURE 7.4: Arena ratings and aggregate win rates in the online study. The adaptive preference model achieves the highest rating and a 60.35% win rate, ahead of profile-conditioned reranking, profile-conditioned generation, and zero-shot generation.

7.4.2 Free-Text Profiles and Pairwise Evidence

The free-text profiles varied from 8 to 72 words, with a median of 28. Participants mentioned cleanliness, hierarchy, action discoverability, typography, spacing, color tone, platform conventions, density, and visual expressiveness. The profiles were multidimensional. A participant could want expressive styling but still prioritize usability, or prefer minimal layouts while valuing visible controls in information-heavy tasks.

The onboarding labels exposed distinctions that the text profiles did not predict. Designers who described a preference for expressive or modern screens still rejected variants with weak contrast. Designers who preferred minimal interfaces sometimes selected a denser option when the extra information improved browsing. One wallpaper-gallery comparison split participants almost evenly between a dense browsing layout with visible filters and a cleaner layout with stronger visual framing. A store-finder empty state produced unanimous preference for the variant with clearer navigation and contrast.

The pattern supports a complementary use of language and behavior. A text profile communicates high-level intent and can guide the generator toward a relevant region of the design space. Pairwise choices reveal how the person resolves a local trade-off inside that region. The two signals should not be collapsed into one prompt because each carries a different kind of evidence.

7.4.3 Arena Results

Figure 7.4 shows the main result. The adaptive pipeline achieves a 60.35% aggregate win rate and an Elo-style rating of 1054.8. The profile-conditioned judge reaches 50.90% and 1005.5. Profile-conditioned generation reaches 47.50% and 987.9. Zero-shot generation reaches 41.25% and 952.8. Across bootstrap samples of participants, the adaptive method ranks first in 96.8% of runs.

The comparison with the profile-conditioned judge is the strongest control. Both conditions receive the same profile text and rank the same set of 16 generated candidates. The only difference is the selection signal. The profile judge interprets the user’s prose; the adaptive model uses the user’s eight concrete choices. The selectors choose the same final UI in only 23 of 240 participant-prompt cases, and the adaptive choices win more often. The result indicates that the model learns more than a reformulation of the text profile.

The advantage also spans prompts. Aggregated across participants, the adaptive method beats zero-shot generation on 14 of 20 prompts, profile-conditioned generation on 16, and the profile judge on 12. The gain does not depend on one screen category or one participant. Eight labels are enough to alter candidate ranking in ways that users prefer, even though the interfaces are generated after onboarding and the users did not appear in the original data bank.

The profile-conditioned generation baseline also provides a useful result. Text profiles improve performance relative to zero-shot generation, but the output still depends on one stochastic sample. Reranking adds value because generation and evaluation are separate problems. A model may produce a candidate that reflects the profile but fail to return that candidate on the first sample. A selection stage can exploit diversity in the candidate pool. Personal preference modeling improves the selection further.

7.5 Applications Beyond the Study

The online study uses reranking as a controlled test, but the preference model can support several forms of adaptive interface behavior. The common pattern is to propose alternatives, score the alternatives with f_u , and treat later user edits as additional evidence rather than as isolated corrections.

7.5.1 Personalized Generative Widgets

An agent may generate a small task-specific widget inside a conversational or operating-system interface. The request can be unique, so a fixed template library cannot cover every output. The system samples K widgets and displays the candidate with the highest user score.

Figure 7.5 shows a prototype with two levels of control. A profile tab offers built-in presets and a personalized preset derived from f_u . A manual tab exposes attributes such as color, padding, and type treatment. The personalized result provides a strong starting point, while direct manipulation preserves local control. An override can be interpreted as an implicit comparison between the proposed state and the edited state, which lets the preference model improve through ordinary use.

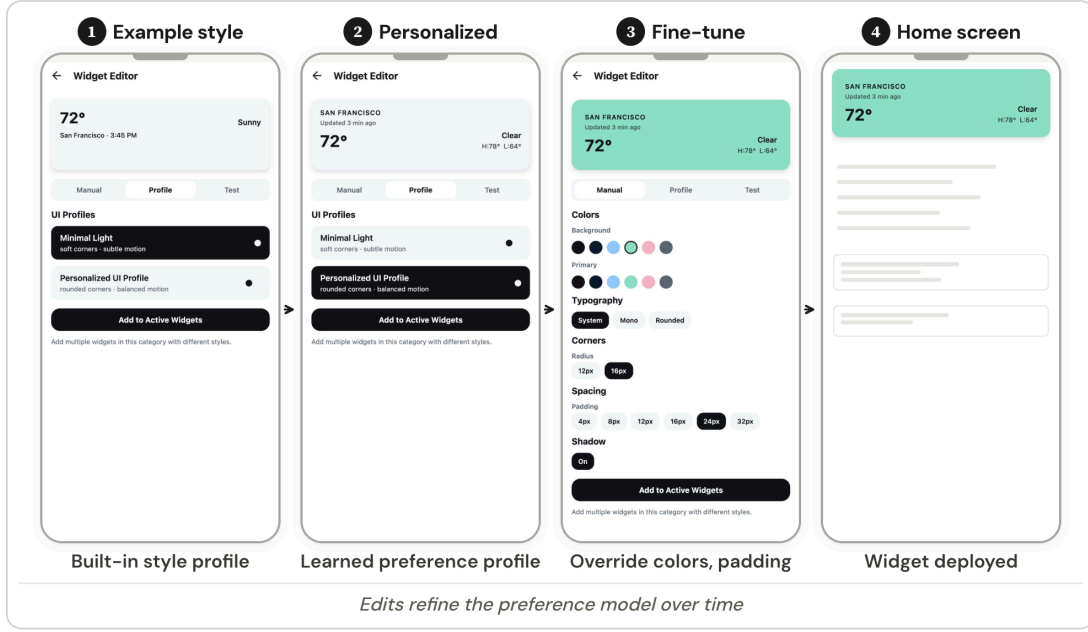


FIGURE 7.5: A personalized widget editor. A learned profile proposes the initial style, while direct controls let the user override individual attributes. Each accepted or rejected edit can become another preference signal.

7.5.2 Personalized UI Configuration

Many applications already expose a finite set of visual settings such as typeface, text size, theme, spacing, and content density. Users often configure the same preferences repeatedly across products. A preference model can enumerate possible configurations, render a representative screen under each configuration, and choose

$$v_u^* = \arg \max_{v_i \in \mathcal{V}} f_u(\text{render}(p, v_i)), \quad (7.11)$$

where p is a representative page and $\mathcal{V}$ is the set of allowed configurations.

The reading prototype in Figure 7.6 applies the learned profile with one action. The mechanism can also support accessibility preferences if the candidate space includes contrast, type size, motion, target size, or information presentation. The system should preserve hard accessibility constraints separately from aesthetic ranking so that personalization does not select a visually preferred but unusable configuration.

7.5.3 Personalized Design Suggestions

Design tools increasingly generate layout, palette, and component suggestions. A generic quality model can filter broken outputs, but the default ordering often reflects an average user or arbitrary generation order. The preference model can rerank the suggestion gallery:

$$\text{rank}(g_i) \propto f_u(g_i). \quad (7.12)$$

The presentation-editor prototype in Figure 7.7 illustrates the connection to Diffscriber. A BLV author may want suggestions that reflect the author's own style, the style of a collaborator, or a

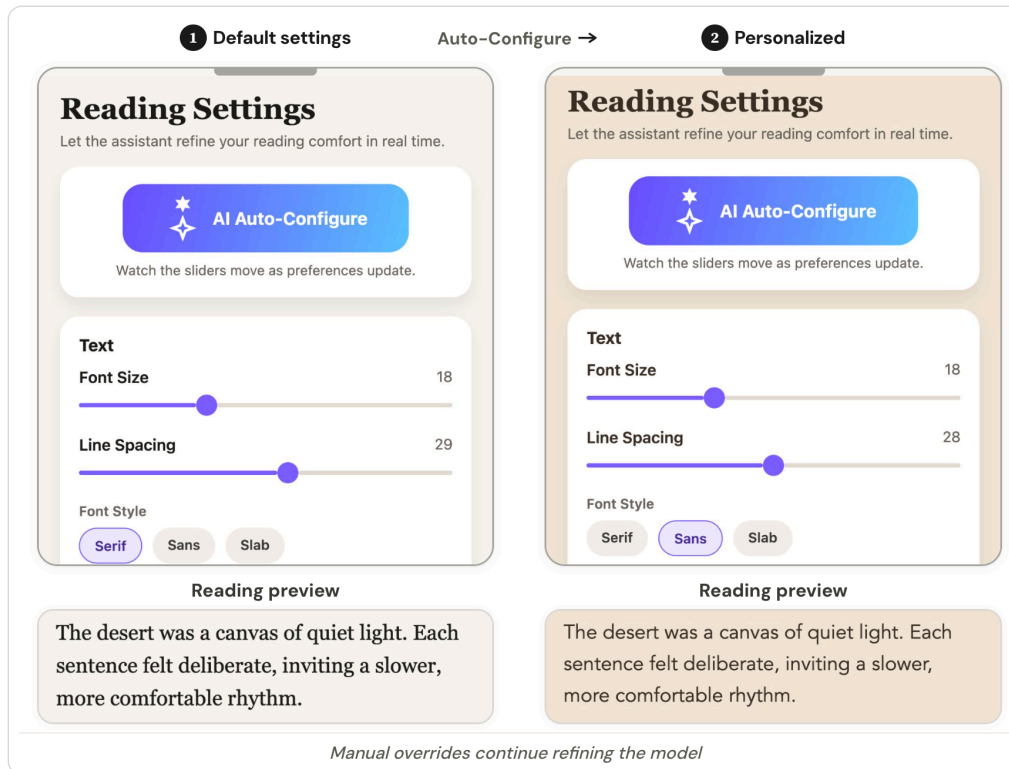


FIGURE 7.6: A reading application before and after preference-based auto-configuration. The system chooses among supported settings rather than generating arbitrary code, and later manual overrides refine the profile.

jointly negotiated project identity. Reranking can place more relevant options first, while change descriptions can explain what differs before the user accepts a suggestion. Personalization and accessible artifact understanding therefore solve complementary parts of the same collaborative workflow.

7.6 Discussion

7.6.1 Population Data as a Prior Rather Than an Average

The central modeling choice is to preserve the identities behind preference labels. Pooled data remains valuable. The unpersonalized retrieval model already reaches 0.610 accuracy because prior designers provide a strong account of the generated-UI distribution. The online method also needs the prior to generalize from eight answers. The problem arises when aggregation becomes the final objective. A majority label discards the structure needed to adapt to a user who consistently falls outside the majority on certain trade-offs.

A distribution over prior designers offers one practical compromise between a universal model and a model trained from scratch. The representation shares evidence across users, supports Bayesian updates, and exposes uncertainty. The representation also has limits. Historical designers may not span the new user's preferences, and the mixture assumes that useful similarity can be expressed through combinations of their observed choices. A larger and more diverse prior

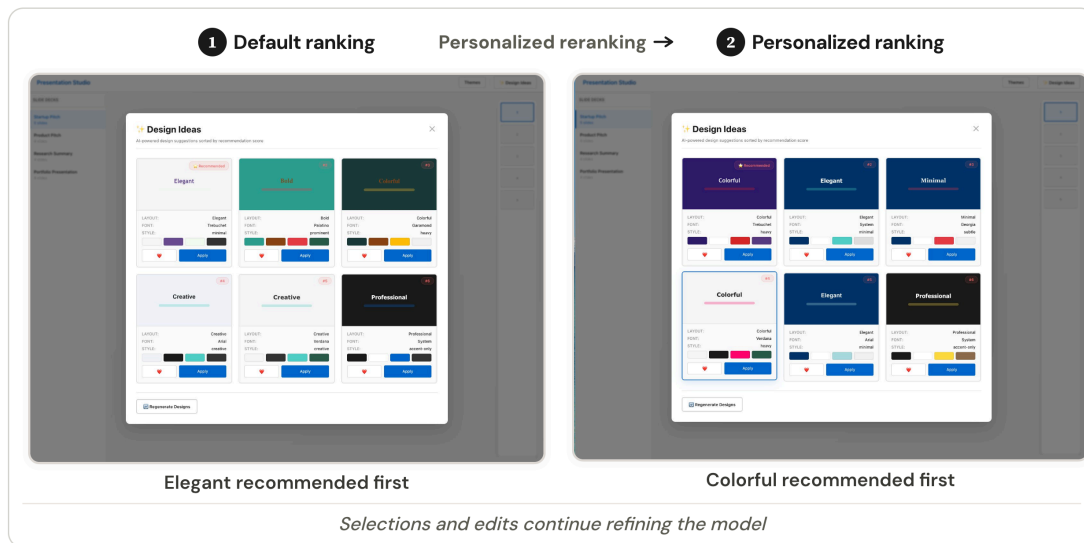


FIGURE 7.7: Preference-based reranking in a presentation editor. A generic gallery places one style first; the personalized gallery promotes the style that better matches the designer's learned taste.

should improve coverage, but future systems also need an open-set mechanism that can recognize when no prior profile fits.

7.6.2 Why Active Elicitation Matters

A preference system should treat questions as an interaction cost. A static onboarding form often asks every user the same items because the system cannot adapt its information needs. EIG instead asks which answer would most change the current belief. The principle connects personalization to the first half of the dissertation. Morae pauses when user input matters for the task. The present method asks when user input matters for the preference model. In both cases, the system should avoid asking merely because uncertainty exists. The question should be timely, relevant, and capable of changing the next action.

The current objective optimizes model uncertainty, not user burden. A pair with high information gain may still be difficult to judge, visually inaccessible, or irrelevant to the user's likely tasks. A deployment policy should combine information gain with estimated cognitive effort, accessibility, and expected downstream value. The system could defer an abstract onboarding question and instead collect a similar signal at a natural decision point during use.

7.6.3 From Explicit Onboarding to Continual Learning

The prototypes suggest a path from eight explicit comparisons to long-term adaptation. Candidate selection, manual overrides, edits, undo actions, and rejected suggestions all provide implicit comparisons. Such signals are abundant but ambiguous. A skipped suggestion may reflect dislike, lack of time, or irrelevance to the current task. A manual edit may correct an error rather than express a stable preference. Continual learning therefore needs confidence-weighted updates and a record of the context in which the signal arose.

Long-term use also introduces preference drift. A person's taste can change with task, audience, role, or design trend. A single persistent profile may overgeneralize across contexts. The model should distinguish stable preferences, such as a need for high contrast, from situational preferences, such as a desire for a dense dashboard during analysis but a sparse screen during presentation. A hierarchical profile could maintain global tendencies alongside task-specific and project-specific components.

7.6.4 Uncertainty Within a Person

DesignPref measured substantial disagreement across people, but a person's own choice can also vary. Some comparisons require long deliberation. A person may focus on layout during one judgment and color during another. Preference strength, response time, repeated choices, and similarity among queried pairs can all provide evidence about confidence.

The current pipeline treats every onboarding answer as equally reliable. A richer Bayesian model could represent both uncertainty about the user's latent profile and uncertainty in each observation. A hesitant slight preference should shift the posterior less than a quick strong preference. The system could also present two candidates when posterior uncertainty remains high rather than force a single personalized result. Personalization should expose ambiguity instead of converting sparse evidence into false certainty.

7.6.5 Reranking Versus Personalized Generation

Reranking offers strong practical advantages. The method works with closed models, separates generation quality from preference alignment, and supports controlled comparison over a shared pool. Reranking also limits the reachable output to what the generator samples. If none of the candidates express the user's preferred trade-off, a perfect selector cannot recover the desired interface.

A future generator could condition directly on the learned preference representation, then retain reranking as a verification layer. Direct conditioning may improve sample efficiency by moving the candidate distribution toward the user. A separate selector remains useful because a personalized prompt does not guarantee that every sample follows the preference or satisfies shared quality constraints. The most robust pipeline may combine preference-conditioned generation, shared correctness checks, personal reranking, and a user-facing mechanism for correction.

7.7 Limitations and Future Work

The prior data bank includes 20 professional designers. Dense repeated judgments make the corpus valuable for personalization research, but the population remains small and occupationally narrow. A larger corpus could reveal preference clusters, improve open-set coverage, and support adaptation for end users without design training. The current method also assumes that the same visual preference model transfers across screen types. More data could support hierarchical priors that distinguish general taste from domain-specific preference.

The online study asks for eight explicit comparisons. The study shows that the budget is useful, not that eight is universally optimal. Different users and tasks may justify different stopping points. A calibrated system could stop when the expected gain from another question falls below its interaction cost, and resume elicitation later when a natural choice arises.

The method personalizes single-page generated interfaces. Full applications introduce navigation, interaction flow, backend behavior, privacy, and persistent state. A visually preferred screen may perform poorly over a multi-step task. Future preference models should include dynamic behavior and long-horizon workflow quality, along with representations that connect a later correction to the earlier generated choice that caused it.

Finally, the pipeline relies on post-hoc selection from a shared candidate pool. As open models and adaptation interfaces improve, preference learning could guide the generator itself. The deployment objective should still remain layered: shared checks for functionality, accessibility, and task fit; personalized guidance for legitimate trade-offs; and user control when the system lacks enough evidence.

7.8 Conclusion

This chapter turns the disagreement measured in DesignPref into an end-to-end personalization method. A new user is represented as a distribution over prior designers. Expected information gain selects a small set of comparisons, Bayesian updates refine the user distribution, and personalized retrieval transfers the evidence to newly generated candidates. The method outperforms a pretrained UI evaluator and a larger multimodal judge in offline prediction. In an online study, eight adaptive comparisons guide generation more effectively than a free-text profile alone.

The result changes the role of population preference data. Global labels should not define one universal design target. They provide a prior that supports efficient individual adaptation. Behavioral choices then identify how a person resolves the trade-offs that remain after shared quality requirements are satisfied. In this form, delegation can preserve agency without requiring constant confirmation: the agent acts for the person, while the learned objective remains grounded in the person's own choices and open to later correction. The final chapter places the result beside the dissertation's work on accessible control and outlines a broader research agenda for collaborative agents that expose plans, communicate state, learn preferences, and remain correctable over time.

Chapter 8

Discussion and Future Directions

Agency is reconsidered whenever the state, stakes, or delegation changes.

Through an accessibility lens, the dissertation defines two responsibilities for collaborative agents: they should surface choices when user intent is unclear and choose well after users delegate decisions. The discussion turns the results into a decision policy and focuses future work on longer collaboration, continual personalization, and inspectable shared artifacts.

An agent can complete a task and still leave the person with less control over the outcome. Morae exposed the problem during execution: an agent could satisfy a shopping request while silently choosing among products that a BLV user did not know existed. Diffscriber exposed a related problem after work had been delegated: a blind author could receive a finished presentation without enough access to the visual decisions that shaped it. DesignPref and the final personalization method showed the same tension in generative interfaces. A model could select a reasonable screen while choosing according to an average preference that fit no particular person well.

The dissertation developed two responses. When the user's intent no longer determines the next action, the system should make the relevant state accessible and surface the choice to the person. When a person delegates a recurring subjective decision, the system should learn from personal evidence so it can choose well and keep the result open to inspection, correction, and override. DreamStruct, GUIDE, and TEMPURA address the representations needed to recognize the first situation. UIClip, DesignPref, and the final personalization method address the shared and personal judgments needed for the second.

The thesis claim is:

Through an accessibility lens, I build collaborative AI agents that surface choices when user intent is unclear and choose well when users delegate decisions.

Human agency is the goal. Decision authority is the narrower interaction question of who decides at one moment. A person can hand decision authority to an agent and retain agency when the delegation is informed, aligned, and reversible. A person can also appear to retain authority while lacking agency if the relevant state remains inaccessible or the system hides the available alternatives.

8.1 Agency as a Decision-by-Decision Policy

The results suggest four factors that agents should consider while reaching a consequential choice.

Decision condition	Risk to agency	Response
Low uncertainty, little preference sensitivity, and an easily reversed action	Excessive confirmation adds friction without improving the result	Act, report progress, and preserve an undo path.
Several valid outcomes depend on the person	The agent substitutes its default for the user's judgment	Surface the alternatives, explain the difference, and ask at the point where the answer changes the action.
The agent can inspect state that the person cannot	The person cannot monitor, verify, or correct execution	Translate the relevant state into an accessible summary linked to the underlying object or action.
The same subjective trade-off appears repeatedly	Repeated questions become burdensome, while a generic choice remains misaligned	Learn from explicit and implicit feedback, represent uncertainty, and keep the choice overridable.
A mistake is costly or difficult to detect	A small model error becomes a hard-to-recover failure	Expose evidence, create a checkpoint, and require confirmation before commitment.

TABLE 8.1: A decision policy for preserving human agency. The response depends on what the agent knows, what the person can inspect, whether the choice is personal, and how costly a mistake would be.

Uncertainty about the task or environment. The agent may not know which action satisfies the request, whether the interface state is complete, or whether the current plan remains valid.

Sensitivity to personal judgment. Several actions may all satisfy the task while producing outcomes that differ in flavor, visual style, information density, privacy, or degree of automation.

Access to the evidence. The agent may see screenshots, code, metadata, or histories that the person cannot inspect efficiently. The gap is acute for BLV users, but the same problem appears whenever an agent acts quickly across remote tools or generated artifacts.

Consequence and reversibility. An easily undone formatting change can tolerate more autonomy than a purchase, deletion, publication, or design decision that becomes the basis for later work. The cost includes the effort required to notice the error, not only the effort required to reverse it.

Table 8.1 maps common combinations to an interaction response. The rows describe patterns rather than rigid rules. A high-consequence choice may require both an explanation and explicit approval. A repeated subjective choice may justify preference learning while preserving a visible override.

The policy separates two kinds of missing information. *Task uncertainty* concerns the environment: the agent may not know which button, plan, or outcome is correct. Better perception and reasoning can reduce the uncertainty. *Preference uncertainty* concerns the person: the agent may know every available option but not know which valid option the person wants. More model scale cannot reveal a preference that the person has never expressed. The system has to ask, learn, or deliberately defer. The policy also changes what counts as an accessible agent. Accessibility does not end when the agent can operate a UI that a BLV person cannot. The person needs

access to the agent's consequential state: what the agent plans, which alternatives appeared, what changed, what remains uncertain, and how to intervene. Task success should therefore be measured alongside awareness of choices, undetected failure, correction cost, and the effort required to verify the outcome. The completed work leaves three directions that matter most for the next generation of collaborative agents.

8.2 Collaboration Across Longer Workflows

Morae returns one choice at the moment when the interface makes the choice concrete. Real work often lasts much longer. A person may revise the goal, explore unfamiliar features, repeat an operation, or discover that an earlier decision created a later problem. A chat transcript and a stream of clicks provide a record, but they rarely give the person an editable account of the shared work.

A stronger interface would make the plan a persistent object. The plan should record what success means, which steps remain provisional, who currently owns each decision, and what evidence supports the next action. Systems such as Cocoa and CowPilot show the value of co-planning and shared control rather than a single handoff from person to agent [39, 41]. The dissertation adds an accessibility requirement: every plan step needs a stable identifier, a concise state, and a link to the relevant interface object. A screen-reader user should be able to ask for the next step that needs input, the last step that changed, or the evidence behind a planned action without reading the entire trace.

Longer workflows also require a graded intervention policy. GUIDE shows that current models struggle to distinguish productive exploration from frustration or debugging [3]. TEMPURA shows that event continuity and temporal boundaries need explicit supervision [4]. Future agents should combine the two. A repeated failed strategy may justify a direct question. A weak signal may justify a quiet suggestion. A routine action may justify no interruption at all. The system should update the policy as the workflow develops rather than classify the user once and keep the label fixed.

Local repair is equally important. When a person corrects one assumption, the agent should not discard every completed step. The plan should preserve accepted work, identify which later steps depend on the correction, and explain the resulting change. Diffscriber already applies the idea to visual artifacts through before-and-after descriptions. The same approach can support agent plans: describe the delta, retain provenance, and let the person repair one decision without restarting the task.

8.3 Continual Personalization Across Tasks and Contexts

The final personalization method shows that eight carefully chosen comparisons can improve generative UI selection. Eight answers provide a useful start, not a finished model of the person. Preferences can remain stable across years, vary by application, follow a project or collaborator, or shift with the current task. A person may prefer dense dashboards at work and sparse reading interfaces at night. A single frozen profile cannot represent the difference.

A continual model should separate several timescales. Stable access needs and interaction constraints should change cautiously. Broad visual preferences can transfer across applications but still evolve. Project conventions may apply only to one team or artifact. The current goal may override all three for one task. A hierarchical model could update the local context strongly and the global profile only when repeated evidence supports the change.

Natural use also produces evidence that is less clean than an onboarding comparison. Selection, editing, undo, repeated generation, hesitation, and abandonment may all reveal a preference. The same behavior can also reflect model error, curiosity, or a changed goal. A useful system should attach confidence and context to each observation rather than treating every click as a stable statement of taste. When an uncertain observation would cause a large profile update, the agent can ask a focused follow-up question.

The person should also be able to inspect and edit the learned profile. A preference history can show representative examples, inferred rules, confidence, scope, and the last update. The user should be able to delete an inference, mark it as task-specific, or replace it with an explicit statement. Population data should remain a prior that helps a new user begin; the population average should never become an invisible rule that the person cannot contest. Continual personalization also creates a transfer question. The dissertation learns design taste in generated UIs and design, but a useful agent will encounter slides, documents, widgets, websites, and other visual artifacts. Some preferences may transfer at the level of hierarchy, density, or visual tone. Others depend on the artifact and audience. Future work should test which parts of a learned preference model can move across domains and which parts need new evidence.

8.4 Accessible, Verifiable, and Shared Artifacts

As agents generate more interfaces and visual artifacts, the artifact itself becomes part of the collaboration. A plausible screenshot does not show whether the code works, whether the semantics are accessible, which user preference shaped the design, or what changed from the previous version. A dependable system needs to connect appearance, implementation, and provenance.

The evaluation order should be explicit. The artifact must first satisfy the requested function and required accessibility constraints. The system can then assess broad design quality and personal fit. A schematic score makes the ordering clear:

$$\text{score}(I, u) = \begin{cases} -\infty, & \text{if a required constraint fails,} \\ S_{\text{shared}}(I) + S_{\text{personal}}(I, u), & \text{otherwise.} \end{cases} \quad (8.1)$$

The separation prevents a preference model from trading away keyboard access, readable text, required content, or correct interaction for a style that the user often likes. Generated code also needs checks that operate on behavior rather than on the screenshot alone. Work on AI-assisted accessibility repair shows both the value of model support and the need for explicit verification in the development process [279].

Every revision should remain inspectable. DreamStruct connects pixels to structure. Diff-scriber connects one version of an artifact to the next. A future design agent can combine the

two: preserve stable element identities, describe the change, identify which requirement or preference motivated it, and link the description to the editable object. The workflow then becomes collaborative editing rather than a sequence of replacements that erase earlier decisions.

Shared artifacts also raise a multi-party question. A BLV author, a sighted designer, an organization, an audience, and an AI agent may each have different claims on the result. GenAssist shows how accessible generative tools can help BLV creators inspect and direct visual generation rather than accept an opaque image [68]. Future systems should record whose preference governs each decision and whether the decision was shared, delegated, or disputed. When preferences conflict, the agent should surface a concrete trade-off instead of averaging the labels. One layout may fit the author's preference for concise text, while another may fit a collaborator's preference for visible evidence. The system can support negotiation by preserving both options and explaining the difference.

The same requirements apply to generated interaction controls. Morae's choice widgets show how a dynamic interface can expose missing information and constrain the next action. A personalized system can adapt the visual form, but roles, labels, focus behavior, keyboard operation, and state announcements should remain stable. Personalization should operate inside an accessibility contract rather than regenerate access from scratch on every turn.

8.5 Conclusion

This dissertation began with a simple failure. An agent completed a request while taking a choice away from a person who could not see that the choice existed. The studies that followed found the same pattern in visual collaboration and generative design, where an agent can produce a plausible result while hiding the decisions that shaped it or choosing according to the wrong person's preferences.

To preserve the person's agency, the agent must surface choices when the person's intent is unclear. The agent must learn the person's preferences so it can choose well after a decision is delegated. The shared work must remain accessible, inspectable, and revisable after the action is complete. Decision authority can move from step to step, but the person should never lose the practical ability to understand and shape the outcome because the agent treated one initial request as permission to decide everything. Accessibility provides a challenging yet realistic test because inaccessible state makes every silent assumption more costly. Personalization provides a complementary test because no universal objective can settle every valid choice.

AI agents will keep gaining capability, and the question this dissertation opened with grows with them, because every new ability creates new decisions that have to sit with someone. What makes an agent collaborative is not what it can do for a person but what it leaves in the person's hands — which choices come back, whose judgment shapes the ones that do not, and how much of the result stays open to revision. An agent that completes the task is automation. An agent that completes the task while the person keeps the choices that matter is collaboration.

References

- [1] Y.-H. Peng, D. Li, J. P. Bigham, and A. Pavel. “Morae: Proactively Pausing UI Agents for User Choices”. In: *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 2025, pp. 1–14. DOI: 10.1145/3746059.3747797.
- [2] Y.-H. Peng, F. Huq, Y. Jiang, J. Wu, X. Y. Li, J. P. Bigham, and A. Pavel. “DreamStruct: Understanding Slides and User Interfaces via Synthetic Data Generation”. In: *European Conference on Computer Vision*. Springer. 2024, pp. 466–485. DOI: 10.1007/978-3-031-72691-0_26.
- [3] S. Yang, J. Yu, Y.-H. Peng, K. Q. Lin, J. W. Cho, Y. Song, and J. Kim. “GUIDE: A Benchmark for Understanding and Assisting Users in Open-Ended GUI Tasks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2026, pp. 13017–13027.
- [4] J.-H. Cheng, Y.-H. Peng, V. Wang, H. Wang, H. Zhou, H.-I. Liu, H.-W. Huang, K.-M. Chen, C.-Y. Yang, W. Chai, Y.-L. Chen, V. Vineet, Q. Cai, and J.-N. Hwang. “TEMPURA: Temporal Event Masked Prediction and Understanding for Reasoning in Action”. In: *arXiv preprint arXiv:2505.01583* (2025).
- [5] Y.-H. Peng, J. Wu, J. Bigham, and A. Pavel. “Diffscriber: Describing Visual Design Changes to Support Mixed-Ability Collaborative Presentation Authoring”. In: *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 2022, pp. 1–13. DOI: 10.1145/3526113.3545637.
- [6] J. Wu, Y.-H. Peng, X. Y. A. Li, A. Swearngin, J. P. Bigham, and J. Nichols. “UIClip: A Data-driven Model for Assessing User Interface Design”. In: *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 2024, pp. 1–16. DOI: 10.1145/3654777.3676408.
- [7] Y.-H. Peng, J. P. Bigham, and J. Wu. “DesignPref: Capturing Personal Preferences in Visual Design Generation”. In: *arXiv preprint arXiv:2511.20513* (2025).
- [8] Y.-H. Peng, S. Das, J. P. Bigham, and J. Wu. “Efficient Personalization of Generative User Interfaces”. In: *arXiv preprint arXiv:2604.09876* (2026). URL: <https://arxiv.org/abs/2604.09876>.
- [9] J. P. Bigham, A. C. Cavender, J. T. Brudvik, J. O. Wobbrock, and R. E. Ladner. “WebinSitu: a comparative analysis of blind and sighted browsing behavior”. In: *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility*. 2007, pp. 51–58.

- [10] J. P. Bigham, I. Lin, and S. Savage. “The Effects of “Not Knowing What You Don’t Know” on Web Accessibility for Blind Web Users”. In: *Proceedings of the 19th international ACM SIGACCESS conference on computers and accessibility*. 2017, pp. 101–109.
- [11] C. L. Bennett, E. Brady, and S. M. Branham. “Interdependence as a frame for assistive technology research and design”. In: *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. 2018, pp. 161–173.
- [12] J. O. Wobbrock, S. K. Kane, K. Z. Gajos, S. Harada, and J. Froehlich. “Ability-based design: Concept, principles and examples”. In: *ACM Transactions on Accessible Computing (TACCESS)* 3.3 (2011), pp. 1–27.
- [13] D. Bennett, O. Metatla, A. Roudaut, and E. D. Mekler. “How Does HCI Understand Human Agency and Autonomy?” In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2023. DOI: 10.1145/3544548.3580651.
- [14] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. “A Model for Types and Levels of Human Interaction with Automation”. In: *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans* 30.3 (2000), pp. 286–297. DOI: 10.1109/3468.844354.
- [15] K. J. K. Feng, D. W. McDonald, and A. X. Zhang. “Levels of Autonomy for AI Agents”. In: *Knight First Amendment Institute – AI and Democratic Freedoms Essay Series* (2025). URL: <https://arxiv.org/abs/2506.12469>.
- [16] E. Horvitz. “Principles of Mixed-Initiative User Interfaces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1999, pp. 159–166. DOI: 10.1145/302979.303030.
- [17] M. A. Hearst. “Mixed-Initiative Interaction”. In: *IEEE Intelligent Systems* 14.5 (1999), pp. 14–23.
- [18] B. Shneiderman. “Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy”. In: *International Journal of Human–Computer Interaction* 36.6 (2020), pp. 495–504. DOI: 10.1080/10447318.2020.1741118.
- [19] J. A. Fails and D. R. Olsen Jr. “Interactive Machine Learning”. In: *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI ’03)*. ACM, 2003, pp. 39–45.
- [20] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. “Power to the People: The Role of Humans in Interactive Machine Learning”. In: *AI Magazine* 35.4 (2014), pp. 105–120.
- [21] K. Gajos and D. S. Weld. “Preference elicitation for interface optimization”. In: *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 2005, pp. 173–182.
- [22] K. Z. Gajos, J. O. Wobbrock, and D. S. Weld. “Automatically generating user interfaces adapted to users’ motor and vision capabilities”. In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. 2007, pp. 231–240. DOI: 10.1145/1294211.1294253.

- [23] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. “Automatically generating personalized user interfaces with Supple”. In: *Artificial Intelligence* 174.12–13 (2010), pp. 910–950. DOI: 10.1016/j.artint.2010.05.005.
- [24] A. Bunt, C. Conati, and J. McGrenere. “Supporting interface customization using a mixed-initiative approach”. In: *Proceedings of the 12th International Conference on Intelligent User Interfaces*. 2007, pp. 92–101. DOI: 10.1145/1216295.1216317.
- [25] A. Bunt, J. McGrenere, and C. Conati. “Understanding the Utility of Rationale in a Mixed-Initiative System for GUI Customization”. In: *User Modeling 2007*. Vol. 4511. Lecture Notes in Computer Science. Springer, 2007, pp. 147–156. DOI: 10.1007/978-3-540-73078-1_18.
- [26] A. Bunt, C. Conati, and J. McGrenere. “Mixed-Initiative Interface Personalization as a Case Study in Usable AI”. In: *AI Magazine* 30.4 (2009), pp. 58–64. DOI: 10.1609/aimag.v30i4.2264.
- [27] L. Findlater and K. Z. Gajos. “Design space and evaluation challenges of adaptive graphical user interfaces”. In: *AI Magazine* 30.4 (2009), pp. 68–73. DOI: 10.1609/aimag.v30i4.2268.
- [28] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. T. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz. “Guidelines for Human-AI Interaction”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI ’19)*. ACM, 2019, pp. 1–13.
- [29] E. Kamar. “Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence”. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI ’16)*. IJCAI, 2016, pp. 4070–4073.
- [30] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. “Mind2Web: Towards a Generalist Agent for the Web”. In: *Advances in Neural Information Processing Systems* 36 (2023).
- [31] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, Y. Bisk, D. Fried, U. Alon, et al. “WebArena: A Realistic Web Environment for Building Autonomous Agents”. In: *arXiv preprint arXiv:2307.13854* (2023). URL: <https://webarena.dev>.
- [32] J. Y. Koh, R. Lo, L. Jang, V. Duvvur, M. Lim, P.-Y. Huang, G. Neubig, S. Zhou, R. Salakhutdinov, and D. Fried. “VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 881–905. DOI: 10.18653/v1/2024.acl-long.50. URL: <https://aclanthology.org/2024.acl-long.50/>.
- [33] H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, and D. Yu. “WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 6864–6890. DOI: 10.18653/v1/2024.acl-long.371. URL: <https://aclanthology.org/2024.acl-long.371/>.

- [34] C. Zhang, Z. Yang, J. Liu, Y. Han, X. Chen, Z. Huang, B. Fu, and G. Yu. “AppAgent: Multimodal Agents as Smartphone Users”. In: *arXiv preprint arXiv:2312.13771* (2023). URL: <https://arxiv.org/abs/2312.13771>.
- [35] D. Gao, L. Ji, Z. Bai, M. Ouyang, P. Li, D. Mao, Q. Wu, W. Zhang, P. Wang, X. Guo, H. Wang, L. Zhou, and M. Z. Shou. “AssistGUI: Task-Oriented PC Graphical User Interface Automation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, USA: IEEE, 2024, pp. 13289–13298. DOI: 10.1109/CVPR52733.2024.01262. URL: <https://ieeexplore.ieee.org/document/10504394>.
- [36] S. Nayak, X. Jian, K. Q. Lin, J. A. Rodriguez, M. Kalsi, N. Chapados, M. T. Özsu, A. Agrawal, D. Vazquez, C. Pal, P. Taslakian, S. Gella, and S. Rajeswar. “UI-Vision: A Desktop-centric GUI Benchmark for Visual Perception and Interaction”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu. Vol. 267. Proceedings of Machine Learning Research. PMLR, July 2025, pp. 45817–45851. URL: <https://proceedings.mlr.press/v267/nayak25a.html>.
- [37] K. Q. Lin, L. Li, D. Gao, W. Qinchen, M. Yan, Z. Yang, L. Wang, and M. Z. Shou. “VideoGUI: A Benchmark for GUI Automation from Instructional Videos”. In: *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- [38] A. Khurana, X. Su, A. Y. Wang, and P. K. Chilana. “Do It For Me vs. Do It With Me: Investigating User Perceptions of Different Paradigms of Automation in Copilots for Feature-Rich Software”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. CHI ’25. Association for Computing Machinery, 2025. DOI: 10.1145/3706598.3713431. URL: <https://doi.org/10.1145/3706598.3713431>.
- [39] F. Huq, Z. Z. Wang, F. F. Xu, T. Ou, S. Zhou, J. P. Bigham, and G. Neubig. “CowPilot: A Framework for Autonomous and Human-Agent Collaborative Web Navigation”. In: *arXiv preprint arXiv:2501.16609* (2025).
- [40] H. Mozannar, G. Bansal, C. Tan, A. Fournery, V. Dibia, J. Chen, J. Gerrits, T. Payne, M. K. Maldaner, M. Grunde-McLaughlin, E. Zhu, G. Bassman, J. Alber, P. Chang, R. Loynd, F. Niedtner, E. Kamar, M. Murad, R. Hosn, and S. Amershi. “Magentic-UI: Towards Human-in-the-loop Agentic Systems”. In: *arXiv preprint arXiv:2507.22358* (2025).
- [41] K. J. K. Feng, K. Pu, M. Latzke, T. August, P. Siangliulue, J. Bragg, D. S. Weld, A. X. Zhang, and J. C. Chang. “Cocoa: Co-Planning and Co-Execution with AI Agents”. In: *arXiv preprint arXiv:2412.10999* (2024). URL: <https://arxiv.org/abs/2412.10999>.
- [42] M. Venkatesan, S. Wen, J. Guo, J. P. Ono, L. Ren, and D. Liu. “Plover: Steering GUI Agents through Plan-Centric Interaction”. In: *arXiv preprint arXiv:2607.15193* (2026). URL: <https://arxiv.org/abs/2607.15193>.
- [43] S. Andolina, V. Orso, H. Schneider, K. Klouche, T. Ruotsalo, L. Gamberini, and G. Jacucci. “SearchBot: Supporting voice conversations with proactive search”. In: *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*. 2018, pp. 9–12.

- [44] V. Chen, A. Zhu, S. Zhao, H. Mozannar, D. Sontag, and A. Talwalkar. “Need Help? Designing Proactive AI Assistants for Programming”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. CHI ’25. Association for Computing Machinery, 2025. DOI: 10.1145/3706598.3714002. URL: <https://doi.org/10.1145/3706598.3714002>.
- [45] K. Pu, D. Lazaro, I. Arawjo, H. Xia, Z. Xiao, T. Grossman, and Y. Chen. “Assistance or Disruption? Exploring and Evaluating the Design and Trade-offs of Proactive AI Programming Support”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. CHI ’25. Association for Computing Machinery, 2025. DOI: 10.1145/3706598.3713357. URL: <https://doi.org/10.1145/3706598.3713357>.
- [46] K. Pu, T. Zhang, N. Sendhilnathan, S. Freitag, R. Sodhi, and T. R. Jonker. “ProMemAssist: Exploring Timely Proactive Assistance Through Working Memory Modeling in Multi-Modal Wearable Devices”. In: *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. UIST ’25. Association for Computing Machinery, 2025. DOI: 10.1145/3746059.3747770. URL: <https://doi.org/10.1145/3746059.3747770>.
- [47] S. Wu, M. Galley, B. Peng, H. Cheng, G. Li, Y. Dou, W. Cai, J. Zou, J. Leskovec, and J. Gao. “CollabLLM: From Passive Responders to Active Collaborators”. In: *arXiv preprint arXiv:2502.00640* (2025).
- [48] Y. Lu, S. Yang, C. Qian, G. Chen, Q. Luo, Y. Wu, H. Wang, X. Cong, Z. Zhang, Y. Lin, W. Liu, Y. Wang, Z. Liu, F. Liu, and M. Sun. “Proactive Agent: Shifting LLM Agents from Reactive Responses to Active Assistance”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=sRIU6k2TcU>.
- [49] B. Yang, L. Xu, L. Zeng, K. Liu, S. Jiang, W. Lu, H. Chen, X. Jiang, G. Xing, and Z. Yan. *ContextAgent: Context-Aware Proactive LLM Agents with Open-World Sensory Perceptions*. 2025. arXiv: 2505.14668 [cs.AI]. URL: <https://arxiv.org/abs/2505.14668>.
- [50] Y. Tang, H. Tang, T. Cao, L. Nguyen, A. Zhang, X. Cao, C. Liu, W. Ding, and Y. Li. “ProAgentBench: Evaluating LLM Agents for Proactive Assistance with Real-World Data”. In: *arXiv preprint arXiv:2602.04482* (2026). URL: <https://arxiv.org/abs/2602.04482>.
- [51] T. Chen, Z. Lu, Z. Xu, G. Shao, S. Zhao, F. Tang, Y. Du, K. Song, Y. Liu, Y. Yan, W. Zhang, X. Tan, W. Lu, J. Xiao, Y. Zhuang, and Y. Shen. “KnowU-Bench: Towards Interactive, Proactive, and Personalized Mobile Agent Evaluation”. In: *arXiv preprint arXiv:2604.08455* (2026). URL: <https://arxiv.org/abs/2604.08455>.
- [52] C. Power, A. Freire, H. Petrie, and D. Swallow. “Guidelines are only half of the story: accessibility problems encountered by blind users on the web”. In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2012, pp. 433–442.
- [53] V. Ashok, Y. Borodin, Y. Puzis, and I. Ramakrishnan. “Capti-speak: a speech-enabled web screen reader”. In: *Proceedings of the 12th International Web for All Conference*. 2015, pp. 1–10.
- [54] Y. Zhong, T. Raman, C. Burkhardt, F. Biadsky, and J. P. Bigham. “JustSpeak: enabling universal voice control on Android”. In: *Proceedings of the 11th Web for All Conference*. 2014, pp. 1–4.

- [55] M. Phutane, C. Jung, N. Chen, and S. Azenkot. "Speaking with My Screen Reader: Using Audio Fictions to Explore Conversational Access to Interfaces". In: *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*. 2023, pp. 1–18.
- [56] S. R. Kodandaram, U. Uckun, X. Bi, I. V. Ramakrishnan, and V. Ashok. "Enabling Uniform Computer Interaction Experience for Blind Users through Large Language Models". In: *arXiv preprint arXiv:2407.19537* (2024).
- [57] A. Gubbi Mohanbabu, R. Natalie, B. Kim, A. Guo, and A. Pavel. "A11y-CUA Dataset: Characterizing the Accessibility Gap in Computer Use Agents". In: *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. 2026, 225:1–225:26. DOI: 10.1145/3772318.3791896.
- [58] M. Perera, S. Ananthanarayan, C. Goncu, and K. Marriott. "The Sky is the Limit: Understanding How Generative AI can Enhance Screen Reader Users' Experience with Productivity Applications". In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, 2025. DOI: 10.1145/3706598.3713634.
- [59] X. Tang, A. Abdolrahmani, D. Gergle, and A. M. Piper. "Everyday Uncertainty: How Blind People Use GenAI Tools for Information Access". In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, 2025.
- [60] J. Hu, C. Yan, Y. Zheng, Z. Wang, and J. Zhang. "Position: Assistive Agents Need Accessibility Alignment". In: *arXiv preprint arXiv:2605.13579* (2026). URL: <https://arxiv.org/abs/2605.13579>.
- [61] M. Das, D. Gergle, and A. M. Piper. "'It Doesn't Win You Friends': Understanding Accessibility in Collaborative Writing for People with Vision Impairments". In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW (2019), pp. 1–26.
- [62] M. Das, T. B. McHugh, A. M. Piper, and D. Gergle. "Co11ab: Augmenting accessibility in synchronous collaborative writing for people with vision impairments". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–18.
- [63] M. Das, A. M. Piper, and D. Gergle. "Design and Evaluation of Accessible Collaborative Writing Techniques for People with Vision Impairments". In: *ACM Transactions on Computer-Human Interaction* 29.2 (2022), pp. 1–42.
- [64] J. Tam and S. Greenberg. "A framework for asynchronous change awareness in collaborative documents and workspaces". In: *International Journal of Human-Computer Studies* 64.7 (2006), pp. 583–598.
- [65] A. Schaadhardt, A. Hiniker, and J. O. Wobbrock. "Understanding blind screen-reader users' experiences of digital artboards". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–19.
- [66] V. Potluri, L. He, C. Chen, J. E. Froehlich, and J. Mankoff. "A multi-modal approach for blind and visually impaired developers to edit webpage designs". In: *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*. 2019, pp. 612–614.

- [67] Z. Zhang and J. O. Wobbrock. “A11yBoard: Making Digital Artboards Accessible to Blind and Low-Vision Users”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023.
- [68] M. Huh, Y.-H. Peng, and A. Pavel. “GenAssist: Making image generation accessible”. In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023, pp. 1–17.
- [69] B. Deka, Z. Huang, C. Franzen, J. Hibschan, D. Afergan, Y. Li, J. Nichols, and R. Kumar. “Rico: A mobile app dataset for building data-driven design applications”. In: *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 2017, pp. 845–854.
- [70] J. Wu, S. Wang, S. Shen, Y.-H. Peng, J. Nichols, and J. P. Bigham. “Webui: A dataset for enhancing visual ui understanding with web semantics”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–14.
- [71] G. Li and Y. Li. “Spotlight: Mobile ui understanding using vision-language models with a focus”. In: *arXiv preprint arXiv:2209.14927* (2022).
- [72] K. You, H. Zhang, E. Schoop, F. Weers, A. Swearngin, J. Nichols, Y. Yang, and Z. Gan. “Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs”. In: *Computer Vision – ECCV 2024*. Springer, 2024. URL: <https://arxiv.org/abs/2404.05719>.
- [73] W. Hong, W. Wang, Q. Lv, J. Xu, W. Yu, J. Ji, Y. Wang, Z. Wang, Y. Dong, M. Ding, and J. Tang. *CogAgent: A Visual Language Model for GUI Agents*. 2023. arXiv: 2312.08914 [cs.CV].
- [74] Y. Lu, J. Yang, Y. Shen, and A. Awadallah. “Omniparser for pure vision based gui agent”. In: *arXiv preprint arXiv:2408.00203* (2024).
- [75] T. Hayama, H. Nanba, and S. Kunifujii. “Structure extraction from presentation slide information”. In: *Pacific Rim International Conference on Artificial Intelligence*. Springer. 2008, pp. 678–687.
- [76] M. Haurilet, Z. Al-Halah, and R. Stiefelhagen. “Spase-multi-label page segmentation for presentation slides”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 726–734.
- [77] T. Ishihara, H. Takagi, T. Itoh, and C. Asakawa. “Analyzing visual layout for a non-visual presentation-document interface”. In: *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*. 2006, pp. 165–172.
- [78] Y.-H. Peng, J. P. Bigham, and A. Pavel. “Slidecho: Flexible non-visual exploration of presentation videos”. In: *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 2021, pp. 1–12.
- [79] Y.-H. Peng, J. Jang, J. P. Bigham, and A. Pavel. “Say it all: Feedback for improving non-visual presentation accessibility”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–12.

- [80] S. Azizi, S. Kornblith, C. Saharia, M. Norouzi, and D. J. Fleet. “Synthetic data from diffusion models improves imagenet classification”. In: *arXiv preprint arXiv:2304.08466* (2023).
- [81] Y. Tian, L. Fan, P. Isola, H. Chang, and D. Krishnan. *StableRep: Synthetic Images from Text-to-Image Models Make Strong Visual Representation Learners*. 2023. arXiv: 2306.00984 [cs.CV].
- [82] Q. Wu, D. Gao, Q. Lin, Z. Wu, and M. Z. Shou. “GUI-Narrator: Detecting and Captioning Computer GUI Actions”. In: *Proceedings of the 33rd ACM International Conference on Multimedia*. MM ’25. Dublin, Ireland: Association for Computing Machinery, 2025, pp. 3683–3692. DOI: 10.1145/3746027.3755150. URL: <https://doi.org/10.1145/3746027.3755150>.
- [83] D. Lu, Y. Xu, J. Wang, H. Wu, X. Wang, Z. Wang, J. Yang, H. Su, J. Chen, J. Chen, Y. Mao, J. Zhou, J. Lin, B. Hui, and T. Yu. *VideoAgentTrek: Computer Use Pretraining from Unlabeled Videos*. 2025. arXiv: 2510.19488 [cs.CL]. URL: <https://arxiv.org/abs/2510.19488>.
- [84] A. Feizi, S. Nayak, X. Jian, K. Q. Lin, K. Li, R. Awal, X. H. Lù, J. Obando-Ceron, J. A. Rodriguez, N. Chapados, D. Vazquez, A. Romero-Soriano, R. Rabbany, P. Taslakian, C. Pal, S. Gella, and S. Rajeswar. *Grounding Computer Use Agents on Human Demonstrations*. 2025. arXiv: 2511.07332 [cs.LG]. URL: <https://arxiv.org/abs/2511.07332>.
- [85] L. Jang, Y. Li, C. Ding, J. Lin, P. P. Liang, D. Zhao, R. Bonatti, and K. Koishida. “VideoWebArena: Evaluating Long Context Multimodal Agents with Video Understanding Web Tasks”. In: *arXiv preprint arXiv:2410.19100* (2024).
- [86] M. Cai, R. Tan, J. Zhang, B. Zou, K. Zhang, F. Yao, F. Zhu, J. Gu, Y. Zhong, Y. Shang, Y. Dou, J. Park, J. Gao, Y. J. Lee, and J. Yang. “TemporalBench: Towards Fine-grained Temporal Understanding for Multimodal Video Models”. In: *arXiv preprint arXiv:2410.10818* (2024).
- [87] L. Qian, J. Li, Y. Wu, Y. Ye, H. Fei, T.-S. Chua, Y. Zhuang, and S. Tang. “Momentor: advancing video large language model with fine-grained temporal reasoning”. In: *Proceedings of the 41st International Conference on Machine Learning*. ICML’24. Vienna, Austria: JMLR.org, 2024.
- [88] D.-A. Huang, S. Liao, S. Radhakrishnan, H. Yin, P. Molchanov, Z. Yu, and J. Kautz. “LITA: Language Instructed Temporal-Localization Assistant”. In: *ECCV*. 2024.
- [89] Z. Cheng, S. Leng, H. Zhang, Y. Xin, X. Li, G. Chen, Y. Zhu, W. Zhang, Z. Luo, D. Zhao, et al. “VideoLLaMA 2: Advancing Spatial-Temporal Modeling and Audio Understanding in Video-LLMs”. In: *arXiv preprint arXiv:2406.07476* (2024).
- [90] Y. Guo, J. Liu, M. Li, X. Tang, Q. Liu, and X. Chen. “Trace: Temporal grounding video llm via causal event modeling”. In: *arXiv preprint arXiv:2410.05643* (2024).
- [91] J. Nielsen. “Finding usability problems through heuristic evaluation”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1992, pp. 373–380.
- [92] A. Miniukovich and A. De Angeli. “Computation of interface aesthetics”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pp. 1163–1172. DOI: 10.1145/2702123.2702575.

- [93] A. Miniukovich, S. Sulpizio, and A. De Angeli. “Visual complexity of graphical user interfaces”. In: *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*. 2018, pp. 1–9. DOI: 10.1145/3206505.3206549.
- [94] A. Oulasvirta, S. De Pascale, J. Koch, T. Langerak, J. Jokinen, K. Todi, M. Laine, M. Krsthombuge, Y. Zhu, A. Miniukovich, G. Palmas, and T. Weinkauff. “Aalto Interface Metrics (AIM): A service and codebase for computational GUI evaluation”. In: *Adjunct Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 2018, pp. 16–19. DOI: 10.1145/3266037.3266087.
- [95] K. Moran, B. Li, C. Bernal-Cárdenas, D. Jelf, and D. Poshyvanyk. “Automated reporting of GUI design violations for mobile apps”. In: *Proceedings of the 40th International Conference on Software Engineering*. 2018, pp. 165–175.
- [96] J. M. Evangelista Belo, M. N. Lystbæk, A. M. Feit, K. Pfeuffer, P. Kán, A. Oulasvirta, and K. Grønbæk. “Auit—the adaptive user interfaces toolkit for designing xr applications”. In: *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 2022, pp. 1–16.
- [97] K. Todi, D. Weir, and A. Oulasvirta. “Sketchplore: Sketch and explore with a layout optimiser”. In: *Proceedings of the 2016 ACM conference on designing interactive systems*. 2016, pp. 543–555.
- [98] A. Swearngin, C. Wang, A. Oleson, J. Fogarty, and A. J. Ko. “Scout: Rapid exploration of interface layout alternatives through high-level design constraints”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–13.
- [99] C. Lee, S. Kim, D. Han, H. Yang, Y.-W. Park, B. C. Kwon, and S. Ko. “GUIComp: A GUI design assistant with real-time, multi-faceted feedback”. In: *Proceedings of the 2020 CHI conference on human factors in computing systems*. 2020, pp. 1–13.
- [100] S. Bunian, K. Li, C. Jemmali, C. Hartevelde, Y. Fu, and M. Seif El-Nasr. “VINS: Visual Search for Mobile User Interface Design”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–14. DOI: 10.1145/3411764.3445762.
- [101] P. Duan, J. Warner, Y. Li, and B. Hartmann. “Generating automatic feedback on ui mockups with large language models”. In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–20.
- [102] P. Duan, C.-Y. Cheng, G. Li, B. Hartmann, and Y. Li. “Uicrit: Enhancing automated design evaluation with a ui critique dataset”. In: *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 2024, pp. 1–17.
- [103] J. Wu, A. Swearngin, A. K. Vajjala, A. Leung, J. Nichols, and T. Barik. “Improving user interface generation models from designer feedback”. In: *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. 2026. DOI: 10.1145/3772318.3791567.
- [104] T. Beltramelli. “pix2code: Generating code from a graphical user interface screenshot”. In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 2018, pp. 1–6.

- [105] J. A. Rodriguez, S. Agarwal, I. H. Laradji, P. Rodriguez, D. Vazquez, C. Pal, and M. Pedersoli. *StarVector: Generating Scalable Vector Graphics Code from Images*. 2023. arXiv: 2312.11556 [cs.CV].
- [106] C. Si, Y. Zhang, Z. Yang, R. Liu, and D. Yang. “Design2Code: How Far Are We From Automating Front-End Engineering?” In: *arXiv preprint arXiv:2403.03163* (2024).
- [107] J. Ge, Z. Z. Wang, X. Zhou, Y.-H. Peng, S. Subramanian, Q. Tan, M. Sap, A. Suhr, D. Fried, G. Neubig, et al. “Autopresent: Designing structured visuals from scratch”. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 2902–2911.
- [108] Y. Cao, P. Jiang, and H. Xia. “Generative and malleable user interfaces with generative and evolving task-driven data model”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 2025, 686:1–686:20. DOI: 10.1145/3706598.3713285.
- [109] B. Min, A. Chen, Y. Cao, and H. Xia. “Malleable overview-detail interfaces”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 2025, 688:1–688:25. DOI: 10.1145/3706598.3714164.
- [110] J. Chen, Y. Zhang, Y. Zhang, Y. Shao, and D. Yang. “Generative interfaces for language models”. In: *arXiv preprint arXiv:2508.19227* (2025).
- [111] P. Okopnyi, O. E. Nordberg, and F. Guribye. “Against generative UI”. In: *Proceedings of the Halfway to the Future Symposium*. 2024, pp. 1–4. DOI: 10.1145/3686169.3686184.
- [112] B. Min, P. Jiang, Z. Huang, and H. Xia. “Gradual Generation of User Interfaces as a Design Method for Malleable Software”. In: *arXiv preprint arXiv:2601.17975* (2026). URL: <https://arxiv.org/abs/2601.17975>.
- [113] Y. Liu, M. Sra, and C. Xiao. “AlignUI: A Method for Designing LLM-Generated UIs Aligned with User Preferences”. In: *arXiv preprint arXiv:2601.17614* (2026). URL: <https://arxiv.org/abs/2601.17614>.
- [114] K. Imteyaz, K. Imteyaz, N. Rajpal, K. Shaikh, M. Muller, and S. Savage. “Design Theater: A Benchmark for Generative UI”. In: *arXiv preprint arXiv:2607.22928* (2026). URL: <https://arxiv.org/abs/2607.22928>.
- [115] J. Ren, X. Shen, Z. Lin, R. Mech, and D. J. Foran. “Personalized Image Aesthetics”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 638–647. DOI: 10.1109/ICCV.2017.76.
- [116] Y. Yang, L. Xu, L. Li, N. Qie, Y. Li, P. Zhang, and Y. Guo. “Personalized Image Aesthetics Assessment with Rich Attributes”. In: *arXiv preprint arXiv:2203.16754* (2022). URL: <https://arxiv.org/abs/2203.16754>.
- [117] J. Yun and J. Choo. “Scaling Up Personalized Image Aesthetic Assessment via Task Vector Customization”. In: *arXiv preprint arXiv:2407.07176* (2024). URL: <https://arxiv.org/abs/2407.07176>.
- [118] A.-S. Maerten, L.-W. Chen, S. De Winter, C. Bossens, and J. Wagemans. “LAPIS: A novel dataset for personalized image aesthetic assessment”. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 6302–6311.

- [119] S. Goree, W. Khoo, and D. J. Crandall. “Correct for whom? subjectivity and the evaluation of personalized image aesthetics assessment models”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 10. 2023, pp. 11818–11827.
- [120] K. Li, L.-M. Po, H. Yang, X. Xu, K. Liu, and Y. Zhao. “AesBiasBench: Evaluating Bias and Alignment in Multimodal Language Models for Personalized Image Aesthetic Assessment”. In: *arXiv preprint arXiv:2509.11620* (2025).
- [121] L. L. Thurstone. “A Law of Comparative Judgment”. In: *Psychological Review* 34.4 (1927), pp. 273–286. DOI: 10.1037/h0070288.
- [122] R. A. Bradley and M. E. Terry. “Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons”. In: *Biometrika* 39.3-4 (1952), pp. 324–345. DOI: 10.1093/biomet/39.3-4.324.
- [123] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. “Deep Reinforcement Learning from Human Preferences”. In: *arXiv:1706.03741* (2017).
- [124] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. “Training Language Models to Follow Instructions with Human Feedback”. In: *NeurIPS*. 2022.
- [125] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”. In: *arXiv:2305.18290* (2023).
- [126] A. P. Dawid and A. M. Skene. “Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28.1 (1979), pp. 20–28.
- [127] F. Caron and A. Doucet. “Efficient Bayesian Inference for Generalized Bradley–Terry Models”. In: *Journal of Computational and Graphical Statistics* 21.1 (2012), pp. 174–196. DOI: 10.1080/10618600.2012.638220.
- [128] S. Poddar, Y. Wan, H. Ivison, A. Gupta, and N. Jaques. “Personalizing reinforcement learning from human feedback with variational preference learning”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 52516–52544.
- [129] X. Li, R. Zhou, Z. C. Lipton, and L. Leqi. “Personalized language modeling from personalized human feedback”. In: *arXiv preprint arXiv:2402.05133* (2024).
- [130] Y. Choi, S. Cho, M. Lee, M. Park, Y. Ko, J. Ok, and D. Kim. “CoPL: Collaborative preference learning for personalizing LLMs”. In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 2025, pp. 12875–12893. DOI: 10.18653/v1/2025.emnlp-main.650.
- [131] J.-N. Li, J. Guan, S. Wu, W. Wu, and R. Yan. “From 1,000,000 users to every user: Scaling up personalized preference for user-level alignment”. In: *arXiv preprint arXiv:2503.15463* (2025). URL: <https://arxiv.org/abs/2503.15463>.

- [132] A. Siththaranjan, C. Laidlaw, and D. Hadfield-Menell. “Distributional preference learning: Understanding and accounting for hidden context in rlhf”. In: *arXiv preprint arXiv:2312.08358* (2023).
- [133] V.-A. Padurean, P. Kamalaruban, N. Kotalwar, A. Gotovos, and A. Singla. “Inference-time personalized alignment with a few user preference queries”. In: *arXiv preprint arXiv:2511.02966* (2025). URL: <https://arxiv.org/abs/2511.02966>.
- [134] M. Oh, S. Lee, and J. Ok. “Active preference-based learning for multi-dimensional personalization”. In: *arXiv preprint arXiv:2411.00524* (2024). URL: <https://arxiv.org/abs/2411.00524>.
- [135] A. Sepiarskaia, J. Kiseleva, F. Radlinski, and M. de Rijke. “Preference elicitation as an optimization problem”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, pp. 172–180. DOI: 10.1145/3240323.3240352.
- [136] S. Kalloori, F. Ricci, and R. Gennari. “Eliciting pairwise preferences in recommender systems”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, pp. 329–337. DOI: 10.1145/3240323.3240364.
- [137] L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and A. Nowe. “Ordered preference elicitation strategies for supporting multi-objective decision making”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 1478–1486.
- [138] S. Alves, C. Duarte, K. Montague, and T. Guerreiro. “Exploring the Role of Interaction Data to Empower End-User Decision-Making in UI Personalization”. In: *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. CHI ’26. 2026. DOI: 10.1145/3772318.3791022.
- [139] J. P. Bigham, T. Lau, and J. Nichols. “Trailblazer: enabling blind users to blaze trails through the web”. In: *Proceedings of the 14th international conference on Intelligent user interfaces*. 2009, pp. 177–186.
- [140] G. Vanderheiden and C. Y. Marte. “Will AI allow us to dispense with all or most accessibility regulations?” In: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–9.
- [141] E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang. “Reinforcement learning on web interfaces using workflow-guided exploration”. In: *arXiv preprint arXiv:1802.08802* (2018).
- [142] G. Kim, P. Baldi, and S. McAleer. “Language models can solve computer tasks”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [143] H. Lai, X. Liu, I. L. Iong, S. Yao, Y. Chen, P. Shen, H. Yu, H. Zhang, X. Zhang, Y. Dong, et al. “AutoWebGLM: Bootstrap And Reinforce A Large Language Model-based Web Navigating Agent”. In: *arXiv preprint arXiv:2404.03648* (2024).
- [144] H. Bai, Y. Zhou, M. Cemri, J. Pan, A. Suhr, S. Levine, and A. Kumar. “Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning”. In: *arXiv preprint arXiv:2406.11896* (2024).

- [145] Target. <https://www.target.com/>. Accessed: 2025-04-09.
- [146] A. Rodrigues. “Breaking barriers with assistive macros”. In: *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. 2015, pp. 351–352.
- [147] E. Pucci, I. Possaghi, C. M. Cutrupi, M. Baez, C. Cappiello, and M. Matera. “Defining Patterns for a Conversational Web”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–17.
- [148] T. Winograd. *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. Tech. rep. MAC-TR-84. Massachusetts Institute of Technology, 1971.
- [149] T. J.-J. Li, A. Azaria, and B. A. Myers. “SUGILITE: creating multimodal smartphone automation by demonstration”. In: *Proceedings of the 2017 CHI conference on human factors in computing systems*. 2017, pp. 6038–6049.
- [150] B. Wang, G. Li, and Y. Li. “Enabling conversational interaction with mobile ui using large language models”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–17.
- [151] A. Srinivasan, M. Dontcheva, E. Adar, and S. Walker. “Discovering natural language commands in multimodal interfaces”. In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 2019, pp. 661–672.
- [152] W. Chen, X. Liu, J. Zhang, I. I. Lam, Z. Huang, R. Dong, X. Wang, and T. Zhang. “MIWA: Mixed-Initiative Web Automation for Better User Control and Confidence”. In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023, pp. 1–15.
- [153] F. Huang, G. Li, T. Li, and Y. Li. “Automatic Macro Mining from Interaction Traces at Scale”. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–16.
- [154] G. P. Laput, M. Dontcheva, G. Wilensky, W. Chang, A. Agarwala, J. Linder, and E. Adar. “Pixeltone: A multimodal interface for image editing”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pp. 2185–2194.
- [155] S. Murty, C. Manning, P. Shaw, M. Joshi, and K. Lee. “BAGEL: Bootstrapping Agents by Guiding Exploration with Language”. In: *arXiv preprint arXiv:2403.08140* (2024).
- [156] A. Patel, M. Hofmarcher, C. Leoveanu-Condrei, M.-C. Dinu, C. Callison-Burch, and S. Hochreiter. “Large Language Models Can Self-Improve At Web Agent Tasks”. In: *arXiv preprint arXiv:2405.20309* (2024).
- [157] TaxyAI. *TaxyAI Browser Extension*. <https://github.com/TaxyAI/browser-extension>. Accessed: 2024-09-10. 2024.
- [158] M. Müller and G. Žunić. *Browser Use: Enable AI to control your browser*. 2024. URL: <https://github.com/browser-use/browser-use>.

- [159] A. Burns, D. Arsan, S. Agrawal, R. Kumar, K. Saenko, and B. A. Plummer. “A dataset for interactive vision-language navigation with unknown command feasibility”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 312–328.
- [160] L. Sun, X. Chen, L. Chen, T. Dai, Z. Zhu, and K. Yu. “Meta-gui: Towards multi-modal conversational agents on mobile gui”. In: *arXiv preprint arXiv:2205.11029* (2022).
- [161] X. H. Lù, Z. Kasner, and S. Reddy. “Weblinx: Real-world website navigation with multi-turn dialogue”. In: *arXiv preprint arXiv:2402.05930* (2024).
- [162] Z. Peng, Y. Kwon, J. Lu, Z. Wu, and X. Ma. “Design and evaluation of service robot’s proactivity in decision-making support process”. In: *proceedings of the 2019 CHI conference on human factors in computing systems*. 2019, pp. 1–13.
- [163] M. Marge, C. Espy-Wilson, N. G. Ward, A. Alwan, Y. Artzi, M. Bansal, G. Blankenship, J. Chai, H. Daumé III, D. Dey, et al. “Spoken language interaction with robots: Recommendations for future research”. In: *Computer Speech & Language* 71 (2022), p. 101255.
- [164] D. Fried, J. Andreas, and D. Klein. “Unified pragmatic models for generating and following instructions”. In: *arXiv preprint arXiv:1711.04987* (2017).
- [165] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, J. Hart, P. Stone, and R. J. Mooney. “Improving grounded natural language understanding through human-robot dialog”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6934–6941.
- [166] F. Zhu, Y. Zhu, X. Chang, and X. Liang. “Vision-language navigation with self-supervised auxiliary reasoning tasks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10012–10022.
- [167] A. Padmakumar, J. Thomason, A. Shrivastava, P. Lange, A. Narayan-Chen, S. Gella, R. Piramuthu, G. Tur, and D. Hakkani-Tur. “Teach: Task-driven embodied agents that chat”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2. 2022, pp. 2017–2025.
- [168] A. Narayan-Chen, P. Jayannavar, and J. Hockenmaier. “Collaborative dialogue in Minecraft”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 5405–5415.
- [169] C.-P. Bara, S. CH-Wang, and J. Chai. “MindCraft: Theory of mind modeling for situated dialogue in collaborative tasks”. In: *arXiv preprint arXiv:2109.06275* (2021).
- [170] M. Marge and A. I. Rudnický. “Miscommunication detection and recovery in situated human-robot dialogue”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 9.1 (2019), pp. 1–40.
- [171] K. Nguyen and H. Daumé III. “Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning”. In: *arXiv preprint arXiv:1909.01871* (2019).
- [172] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. “Vision-and-dialog navigation”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 394–406.

- [173] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. “Asking for Help Using Inverse Semantics”. In: *Robotics: Science and Systems X*. 2014.
- [174] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. “Speaker-follower models for vision-and-language navigation”. In: *Advances in neural information processing systems* 31 (2018).
- [175] M. Hahn, W. Zeng, N. Kannen, R. Galt, K. Badola, B. Kim, and Z. Wang. “Proactive Agents for Multi-Turn Text-to-Image Generation Under Uncertainty”. In: *arXiv preprint arXiv:2412.06771* (2024).
- [176] S. Vijayvargiya, X. Zhou, A. Yerukola, M. Sap, and G. Neubig. “Interactive Agents to Overcome Ambiguity in Software Engineering”. In: *arXiv preprint arXiv:2502.13069* (2025).
- [177] *Introducing Operator*. <https://openai.com/index/introducing-operator/>. Accessed: 2025-04-09.
- [178] *Project Mariner*. <https://deepmind.google/technologies/project-mariner/>. Accessed: 2025-04-07.
- [179] wunderwuzzi23. *System Prompts*. Accessed: April 10, 2025. 2024. URL: https://github.com/wunderwuzzi23/scratch/tree/master/system_prompts.
- [180] Y. Pan, D. Kong, S. Zhou, C. Cui, Y. Leng, B. Jiang, H. Liu, Y. Shang, S. Zhou, T. Wu, et al. “Webcanvas: Benchmarking web agents in online environments”. In: *arXiv preprint arXiv:2406.12373* (2024).
- [181] T. Xue, W. Qi, T. Shi, C. H. Song, B. Gou, D. Song, H. Sun, and Y. Su. “An Illusion of Progress? Assessing the Current State of Web Agents”. In: *arXiv preprint arXiv:2504.01382* (2025).
- [182] *Google Calendar*. <http://calendar.google.com/>. Accessed: 2025-04-09.
- [183] *Google Docs*. <https://docs.google.com/>. Accessed: 2025-04-09.
- [184] Y. Fan, H. Zhao, R. Zhang, Y. Shen, X. E. Wang, and G. Wu. “GUI-Bee: Align GUI Action Grounding to Novel Environments via Autonomous Exploration”. In: *arXiv preprint arXiv:2501.13896* (2025).
- [185] Y. Song, D. Yin, X. Yue, J. Huang, S. Li, and B. Y. Lin. “Trial and error: Exploration-based trajectory optimization for llm agents”. In: *arXiv preprint arXiv:2403.02502* (2024).
- [186] A. Jacovi, S. Swayamdipta, S. Ravfogel, Y. Elazar, Y. Choi, and Y. Goldberg. “Contrastive explanations for model interpretability”. In: *arXiv preprint arXiv:2103.01378* (2021).
- [187] T. Wang, R. Sridhar, D. Yang, and X. Wang. “Identifying and mitigating spurious correlations for improving robustness in nlp models”. In: *arXiv preprint arXiv:2110.07736* (2021).
- [188] H. Shen, T. Kneare, R. Ghosh, K. Alkiek, K. Krishna, Y. Liu, Z. Ma, S. Petridis, Y.-H. Peng, L. Qiwei, et al. “Towards bidirectional human-ai alignment: A systematic review for clarifications, framework, and future directions”. In: *arXiv preprint arXiv:2406.09264* (2024).

- [189] X. Zhang, L. De Greef, A. Swearngin, S. White, K. Murray, L. Yu, Q. Shan, J. Nichols, J. Wu, C. Fleizach, et al. "Screen recognition: Creating accessibility metadata for mobile applications from pixels". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–15.
- [190] Microsoft Learn. *UI Automation Overview*. <https://learn.microsoft.com/en-us/dotnet/framework/ui-automation/ui-automation-overview>. 2024.
- [191] Apple. *Mac Automation Scripting Guide: Automating the User Interface*. 2024. URL: <https://developer.apple.com/library/archive/documentation/LanguagesUtilities/Conceptual/MacAutomationScriptingGuide/AutomatetheUserInterface.html>.
- [192] Y.-H. Peng, M.-T. Lin, Y. Chen, T. Chen, P. S. Ku, P. Taele, C. G. Lim, and M. Y. Chen. "Personal touch: Improving touchscreen usability by personalizing accessibility settings based on individual user's touchscreen interaction". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–11.
- [193] M. Huh, S. Yang, Y.-H. Peng, X. Chen, Y.-H. Kim, and A. Pavel. "AVscript: Accessible Video Editing with Audio-Visual Scripts". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–17.
- [194] R. D. Jacobson. "Navigating maps with little or no sight: An audio-tactile approach". In: *Content Visualization and Intermedia Representations (CVIR'98)*. 1998.
- [195] L. Holloway, K. Marriott, and M. Butler. "Accessible maps for the blind: Comparing 3D printed models with tactile graphics". In: *Proceedings of the 2018 chi conference on human factors in computing systems*. 2018, pp. 1–13.
- [196] T. Götzelmann. "LucentMaps: 3D printed audiovisual tactile maps for blind and visually impaired people". In: *Proceedings of the 18th international ACM Sigaccess conference on computers and accessibility*. 2016, pp. 81–90.
- [197] M. Taeb, A. Swearngin, E. Schoop, R. Cheng, Y. Jiang, and J. Nichols. "Axnav: Replaying accessibility tests from natural language". In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–16.
- [198] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. "Show and tell: A neural image caption generator". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3156–3164.
- [199] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2048–2057.
- [200] V.-Q. Nguyen, M. Suganuma, and T. Okatani. "Grit: Faster and better image captioning transformer using dual visual features". In: *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*. Springer. 2022, pp. 167–184.

- [201] J. Li, D. Li, S. Savarese, and S. Hoi. “Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models”. In: *arXiv preprint arXiv:2301.12597* (2023).
- [202] J. Kim, Y. Choi, M. Kahng, and J. Kim. “FitVid: Responsive and flexible video content adaptation”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–16.
- [203] R. Krosnick, S. W. Lee, W. S. Laseck, and S. Onev. “Espresso: Building responsive interfaces with keyframes”. In: *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2018, pp. 39–47.
- [204] K. Gajos and D. S. Weld. “SUPPLE: Automatically generating user interfaces”. In: *Proceedings of the 9th International Conference on Intelligent User Interfaces*. 2004, pp. 93–100. DOI: 10.1145/964442.964461.
- [205] J. Wu, T. Barik, X. Zhang, C. Lea, J. Nichols, and J. P. Bigham. “Reflow: Automatically Improving Touch Interactions in Mobile Applications through Pixel-based Refinements”. In: *arXiv preprint arXiv:2207.07712* (2022).
- [206] Y.-H. Peng, M.-W. Hsi, P. Taele, T.-Y. Lin, P.-E. Lai, L. Hsu, T.-c. Chen, T.-Y. Wu, Y.-A. Chen, H.-H. Tang, et al. “Speechbubbles: Enhancing captioning experiences for deaf and hard-of-hearing people in group conversations”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–10.
- [207] J. Wu, R. Krosnick, E. Schoop, A. Swearngin, J. P. Bigham, and J. Nichols. “Never-ending Learning of User Interfaces”. In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023, pp. 1–13.
- [208] H. Liu, C. Li, Q. Wu, and Y. J. Lee. “Visual Instruction Tuning”. In: *NeurIPS*. 2023.
- [209] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. “Qwen technical report”. In: *arXiv preprint arXiv:2309.16609* (2023).
- [210] Y.-H. Peng, P. Chi, A. Kannan, M. R. Morris, and I. Essa. “Slide gestalt: Automatic structure extraction in slide decks for non-visual access”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–14.
- [211] M. Haurilet, A. Roitberg, M. Martinez, and R. Stiefelhagen. “Wise—slide segmentation in the wild”. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2019, pp. 343–348.
- [212] A. Araujo, J. Chaves, H. Lakshman, R. Angst, and B. Girod. “Large-scale query-by-image video retrieval using bloom filters”. In: *arXiv preprint arXiv:1604.07939* (2016).
- [213] R. Tanaka, K. Nishida, K. Nishida, T. Hasegawa, I. Saito, and K. Saito. “SlideVQA: A Dataset for Document Visual Question Answering on Multiple Images”. In: *arXiv preprint arXiv:2301.04883* (2023).
- [214] K. Jobin, A. Mishra, and C. Jawahar. “Semantic Labels-Aware Transformer Model for Searching Over a Large Collection of Lecture-Slides”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 6016–6025.

- [215] D. W. Lee, C. Ahuja, P. P. Liang, S. Natu, and L.-P. Morency. “Lecture Presentations Multimodal Dataset: Towards Understanding Multimodality in Educational Videos”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 20087–20098.
- [216] R. Kumar, A. Satyanarayan, C. Torres, M. Lim, S. Ahmad, S. R. Klemmer, and J. O. Talton. “Webzeitgeist: design mining the web”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pp. 3083–3092.
- [217] C. Chen, T. Su, G. Meng, Z. Xing, and Y. Liu. “From UI Design Image to GUI Skeleton: A Neural Machine Translator to Bootstrap Mobile GUI Implementation”. In: *Proceedings of the 40th International Conference on Software Engineering*. ICSE ’18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 665–676. DOI: 10.1145/3180155.3180240. URL: <https://doi.org/10.1145/3180155.3180240>.
- [218] J. Chen, M. Xie, Z. Xing, C. Chen, X. Xu, L. Zhu, and G. Li. “Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?” In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1202–1214. DOI: 10.1145/3368089.3409691. URL: <https://doi.org/10.1145/3368089.3409691>.
- [219] S. Chen, L. Fan, T. Su, L. Ma, Y. Liu, and L. Xu. “Automated Cross-Platform GUI Code Generation for Mobile Apps”. In: *2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)*. 2019, pp. 13–16. DOI: 10.1109/AI4Mobile.2019.8672718.
- [220] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk. “Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps”. In: *IEEE Transactions on Software Engineering* 46.2 (2020), pp. 196–221. DOI: 10.1109/TSE.2018.2844788.
- [221] J. Wu, X. Zhang, J. Nichols, and J. P. Bigham. “Screen parsing: Towards reverse engineering of ui models from screenshots”. In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. 2021, pp. 470–483.
- [222] A. Swearngin and Y. Li. “Modeling mobile interface tappability using crowdsourcing and deep learning”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–11.
- [223] E. Schoop, X. Zhou, G. Li, Z. Chen, B. Hartmann, and Y. Li. “Predicting and explaining mobile ui tappability with vision modeling and saliency analysis”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–21.
- [224] Y. Jiang, E. Schoop, A. Swearngin, and J. Nichols. *ILuvUI: Instruction-tuned LangUage-Vision modeling of UIs from Machine Conversations*. 2023. arXiv: 2310.04869 [cs.HC].
- [225] B. Wang, G. Li, X. Zhou, Z. Chen, T. Grossman, and Y. Li. “Screen2words: Automatic mobile ui summarization with multimodal learning”. In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. 2021, pp. 498–510.

- [226] L. A. Leiva, A. Hota, and A. Oulasvirta. “Enrico: A dataset for topic modeling of mobile UI designs”. In: *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2020, pp. 1–4.
- [227] Y. Chen, W. Li, X. Chen, and L. Van Gool. “Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1841–1850.
- [228] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3234–3243.
- [229] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting. “Large language models in medicine”. In: *Nature medicine* 29.8 (2023), pp. 1930–1940.
- [230] A. R. Khan, S. Khan, M. Harouni, R. Abbasi, S. Iqbal, and Z. Mehmood. “Brain tumor segmentation using K-means clustering and deep learning with synthetic data augmentation for classification”. In: *Microscopy Research and Technique* 84.7 (2021), pp. 1389–1399.
- [231] V. Viswanathan, C. Zhao, A. Bertsch, T. Wu, and G. Neubig. *Prompt2Model: Generating Deployable Models from Natural Language Instructions*. 2023. arXiv: 2308.12261 [cs.CL].
- [232] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci. “Language models are realistic tabular data generators”. In: *arXiv preprint arXiv:2210.06280* (2022).
- [233] V. Jeronymo, L. Bonifacio, H. Abonizio, M. Fadaee, R. Lotufo, J. Zavrel, and R. Nogueira. “InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval”. In: *arXiv preprint arXiv:2301.01820* (2023).
- [234] R. Wang, W. Zhou, and M. Sachan. *Let’s Synthesize Step by Step: Iterative Dataset Synthesis with Large Language Models by Extrapolating Errors from Small Models*. 2023. arXiv: 2310.13671 [cs.CL].
- [235] P. Jandaghi, X. Sheng, X. Bai, J. Pujara, and H. Sidahmed. *Faithful Persona-based Conversational Dataset Generation with Large Language Models*. 2023. arXiv: 2312.10007 [cs.CL].
- [236] S. Fu, N. Tamir, S. Sundaram, L. Chai, R. Zhang, T. Dekel, and P. Isola. *DreamSim: Learning New Dimensions of Human Visual Similarity using Synthetic Data*. 2023. arXiv: 2306.09344 [cs.CV].
- [237] HuggingFaceM4. *WebSight (v0.1)*. Accessed: 2024-01-16. 2024. URL: <https://huggingface.co/datasets/HuggingFaceM4/WebSight>.
- [238] P. Sharma, T. R. Shaham, M. Baradad, S. Fu, A. Rodriguez-Munoz, S. Duggal, P. Isola, and A. Torralba. “A Vision Check-up for Language Models”. In: *arXiv preprint arXiv:2401.01862* (2024).
- [239] Y. Han, C. Zhang, X. Chen, X. Yang, Z. Wang, G. Yu, B. Fu, and H. Zhang. *ChartLlama: A Multimodal LLM for Chart Understanding and Generation*. 2023. arXiv: 2311.16483 [cs.CV].

- [240] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. "Chartgpt: Leveraging llms to generate charts from abstract natural language". In: *IEEE Transactions on Visualization and Computer Graphics* (2024).
- [241] T. Tseng, R. Cheng, and J. Nichols. "Keyframer: Empowering Animation Design using Large Language Models". In: *arXiv preprint arXiv:2402.06071* (2024).
- [242] J. Wu, J. B. Tenenbaum, and P. Kohli. "Neural scene de-rendering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 699–707.
- [243] W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang. "Layoutgpt: Compositional visual planning and generation with large language models". In: *Advances in Neural Information Processing Systems* 36 (2024).
- [244] Z. Tan, A. Beigi, S. Wang, R. Guo, A. Bhattacharjee, B. Jiang, M. Karami, J. Li, L. Cheng, and H. Liu. *Large Language Models for Data Annotation: A Survey*. 2024. arXiv: 2402.13446 [cs.CL].
- [245] *Human Interface Guidelines*. <https://developer.apple.com/design/human-interface-guidelines/>. Accessed: 2024-03-07. 2024.
- [246] Microsoft. *The 10-20-30 Rule of PowerPoint*. 2023. URL: <https://www.microsoft.com/en-us/microsoft-365-life-hacks/presentations/10-20-30-rule-of-powerpoint>.
- [247] M. Strizic. *App Screens Design*. 2024. URL: <https://decode.agency/article/app-screens-design/>.
- [248] Microsoft Support. *What is a slide layout?* 2024. URL: <https://support.microsoft.com/en-us/office/what-is-a-slide-layout-99da5716-92ee-4b6a-a0b5-beea45150f3a>.
- [249] PresentationLoad GmbH. *10 Types of PowerPoint Slides*. 2023. URL: <https://www.presentationload.com/blog/10-types-of-powerpoint-slides/>.
- [250] H. Y. Hsu, X. He, Y. Peng, H. Kong, and Q. Zhang. "Posterlayout: A new benchmark and approach for content-aware visual-textual presentation layout". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 6018–6026.
- [251] C. Gleason, A. Pavel, E. McCamey, C. Low, P. Carrington, K. M. Kitani, and J. P. Bigham. "Twitter A11y: A browser extension to make Twitter images accessible". In: *Proceedings of the 2020 chi conference on human factors in computing systems*. 2020, pp. 1–12.
- [252] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. "Rethinking the role of demonstrations: What makes in-context learning work?" In: *arXiv preprint arXiv:2202.12837* (2022).
- [253] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. "Bertscore: Evaluating text generation with bert". In: *arXiv preprint arXiv:1904.09675* (2019).
- [254] *chartjs*. <https://www.chartjs.org/>. Accessed: 2024-03-07.
- [255] *chartistjs*. <https://gionkunz.github.io/chartist-js/>. Accessed: 2024-03-07.
- [256] *chartcss*. <https://chartscss.org/>. Accessed: 2024-03-07.

- [257] J. Wu, E. Schoop, A. Leung, T. Barik, J. P. Bigham, and J. Nichols. “Uicoder: Finetuning large language models to generate user interface code through automated feedback”. In: *arXiv preprint arXiv:2406.07739* (2024).
- [258] X. Zhou, V. Koltun, and P. Krähenbühl. “Probabilistic two-stage detection”. In: *arXiv preprint arXiv:2103.07461* (2021).
- [259] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. “Deformable detr: Deformable transformers for end-to-end object detection”. In: *arXiv preprint arXiv:2010.04159* (2020).
- [260] M. Tan, R. Pang, and Q. V. Le. “Efficientdet: Scalable and efficient object detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10781–10790.
- [261] J. Lee, Y.-H. Peng, J. Herskovitz, and A. Guo. “Image Explorer: Multi-Layered Touch Exploration to Make Images Accessible”. In: *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 2021, pp. 1–4.
- [262] C. Gleason, A. Pavel, H. Gururaj, K. Kitani, and J. Bigham. “Making GIFs Accessible”. In: *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility*. 2020, pp. 1–10.
- [263] K. Lee, M. Joshi, I. R. Turc, H. Hu, F. Liu, J. M. Eisenschlos, U. Khandelwal, P. Shaw, M.-W. Chang, and K. Toutanova. “Pix2struct: Screenshot parsing as pretraining for visual language understanding”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 18893–18912.
- [264] X. Che, H. Yang, and C. Meinel. “Lecture video segmentation by automatically analyzing the synchronized slides”. In: *Proceedings of the 21st ACM international conference on Multimedia*. 2013, pp. 345–348.
- [265] M. Lin, M. Chau, J. Cao, and J. F. Nunamaker Jr. “Automated video segmentation for lecture videos: A linguistics-based approach”. In: *International Journal of Technology and Human Interaction (IJTHI)* 1.2 (2005), pp. 27–45.
- [266] P. P. Liang, Y. Lyu, X. Fan, Z. Wu, Y. Cheng, J. Wu, L. Chen, P. Wu, M. A. Lee, Y. Zhu, et al. “Multibench: Multiscale benchmarks for multimodal representation learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 10938–10950.
- [267] H. Hu, K. C. Chan, Y.-C. Su, W. Chen, Y. Li, K. Sohn, Y. Zhao, X. Ben, B. Gong, W. Cohen, et al. “Instruct-Imagen: Image generation with multi-modal instruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 4754–4763.
- [268] M. Westerlund. “The emergence of deepfake technology: A review”. In: *Technology innovation management review* 9.11 (2019).
- [269] D. Sato, H. Takagi, and C. Asakawa. “Accessibility evaluation based on machine learning technique”. In: *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*. 2006, pp. 253–254.
- [270] M. Kurze. “TDraw: a computer-based tactile drawing tool for blind people”. In: *Proceedings of the second Annual ACM Conference on Assistive technologies*. 1996, pp. 131–138.

- [271] L. Shi, Y. Zhao, R. Gonzalez Penuela, E. Kupferstein, and S. Azenkot. “Molder: an accessible design tool for tactile maps”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–14.
- [272] J. Li, S. Kim, J. A. Miele, M. Agrawala, and S. Follmer. “Editing spatial layouts through tactile templates for people with visual impairments”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–11.
- [273] W. Seo and H. Jung. “Understanding the community of blind or visually impaired vloggers on YouTube”. In: *Universal Access in the Information Society 20.1* (2021), pp. 31–44.
- [274] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.
- [275] M. J. Sabet, P. Dufter, F. Yvon, and H. Schütze. “SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings”. In: *arXiv preprint arXiv:2004.08728* (2020).
- [276] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, X. Han, A. Mensch, A. Berg, T. Berg, and H. Daumé III. “Midge: Generating image descriptions from computer vision detections”. In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. 2012, pp. 747–756.
- [277] F. J. Och and H. Ney. “A systematic comparison of various statistical alignment models”. In: *Computational Linguistics 29.1* (2003), pp. 19–51.
- [278] N. Reimers and I. Gurevych. “Sentence-bert: Sentence embeddings using siamese bert-networks”. In: *arXiv preprint arXiv:1908.10084* (2019).
- [279] P. Mowar, Y.-H. Peng, J. Wu, A. Steinfeld, and J. P. Bigham. “CodeA11y: Making AI Coding Assistants Useful for Accessible Web Development”. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 2025, pp. 1–15. DOI: 10.1145/3706598.3713335.